2 TRAFFIC DEMAND ESTIMATION IN REAL-TIME APPLICATIONS

2.1 INTRODUCTION

2.1.1 MOBILITY PATTERNS. ORIGIN-DESTINATION MATRICES

Mobility patterns are essential factors in transport planning, traffic control and traffic management systems. Mobility patterns are usually represented in terms of a trip or mobility matrix *T* between origin and destination, over a fixed time period. The t_{ij} elements of the matrix represent the number of trips made for one specific purpose (such as working, shopping, leisure, etc.) for which the origin zone is *i*-th and the destination zone is *j*-th in the geographical area that is the object of the transport study.

The classic method in transport studies is to begin by defining a set of zones in the geographical area that is the object of the transport study, that is to say, the area must be divided into different zones, and the characteristics of each zone must be homogenous, in terms of social and economic variables.

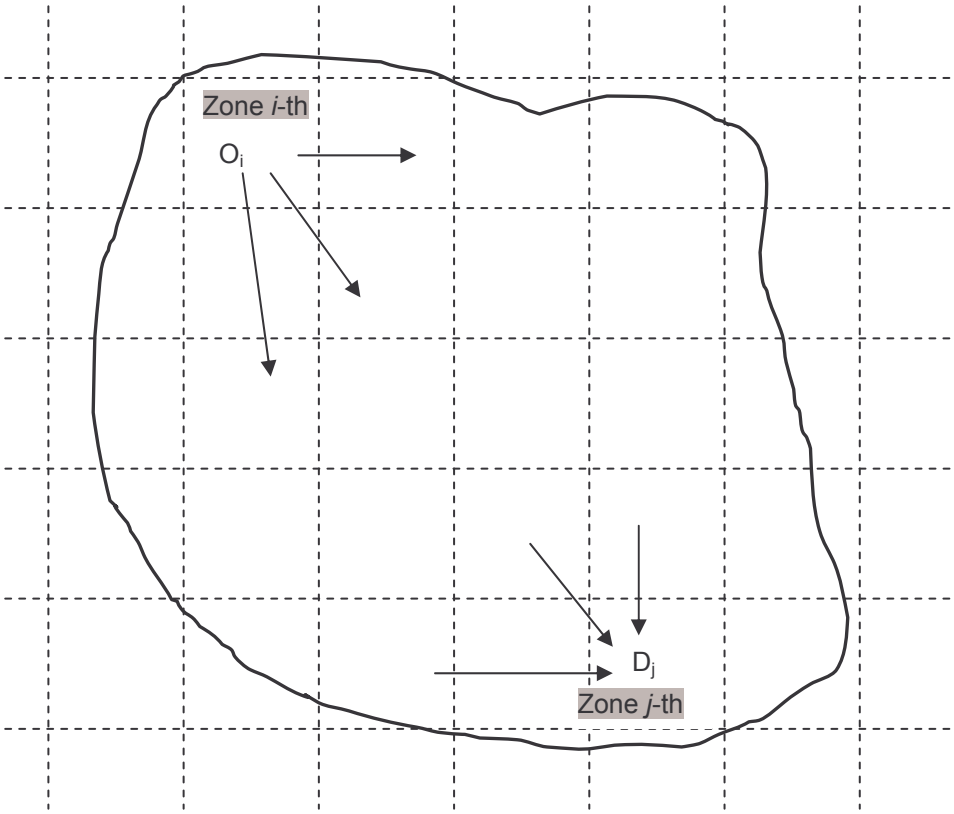


Figure 2.1. Scheme of generation and attraction model

The first step is to find an explanation for the number of trips that are generated in or attracted to one zone according to its social and economic characteristics (housing, cars, income levels, etc). The step is based on a model called a *generation-attraction- model.* The schema of a generation-attraction model is as follows:

$O_i = f_i(S_i^1, S_i^2,...., S_i^k,..., S_i^m)$

$D_j = f_j(S_j^1, S_j^2,...., S_j^k,..., S_j^m)$

where S_i^k represents the social or economic variable *k*-th in zone *i*-th; *Oi* represents the number of trips generated in zone *i*-th; and *Dj* represents the number of trips attracted to zone *j*-th.

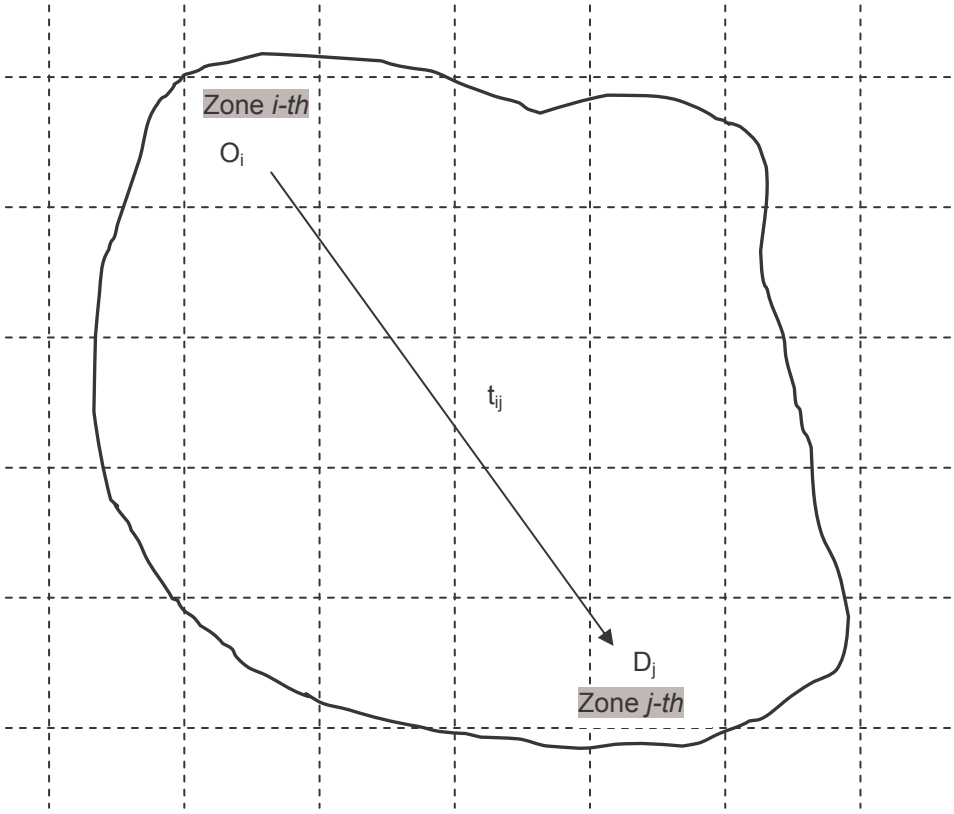


Figure 2.2. Scheme of the distribution model

The generation-attraction model is not sufficient, because as well as knowing the number of trips generated or attracted in each zone, it is essential to know the number of trips generated in a specific zone that has another zone as the destination. This model is known as a *distribution model (*Figure 2.2 depicts the schema of a generation-attraction model). This model is represented by the following restrictions:

(E 2.1)

$$\sum_{j \in J} t_{ij} = O_i \quad \forall i \in I$$

$$\sum_{i \in I} t_{ij} = Dj \quad \forall j \in J$$

$$t_{ij} \geq 0 \quad \forall i \in I, \forall j \in J$$

where I = {*i*/generation zone *i*-th}; J = {*j*/attraction zone *j*-th}; and t_{ij} is the number of trips from zone *i*-th to destination *j*-th.

This distribution model gives a mobility or OD matrix *T*, such that

$$T = \lfloor t_{ij} \rfloor$$

From a generation-attraction model, a matrix *T* that fulfils restrictions (E 2.1 must be found. A *gravity model* that may be used to find the matrix *T* from the generation-model, such that:

$$t_{ij} = k\, O_i\, D_j\, f(u_{ij}) \qquad \text{(E 2.2)}$$

where $f(u_{ij})$ represents a dissuasion function expressed in terms of travel time u_{ij}.

A further model is the maximum entropy model expressed by Wilson (1970):

$$H(T) = MAX \sum_i \sum_j t_{ij} \, (\log t_{ij} - 1)$$

$$s.a.$$

$$\sum_j t_{ij} = O_i \quad , \quad \forall i \in I$$

$$\sum_i t_{ij} = Dj \quad , \quad \forall j \in J$$

$$\sum_i \sum_j u_{ij}\, t_{ij} = \hat{T} \qquad \text{(E 2.3)}$$

where

$$\hat{T} = \text{Total travel time in the network}$$

$$u_{ij} = \text{Travel time from } i \text{ to } j$$

Taking the OD or mobility matrix, the traffic flow behaviour is characterised by applying mathematical programming models.

2.1.2 TRANSPORT PLANNING STUDIES

Transport problems clearly constitute a domain in which mathematical programming models and techniques based on nonlinear optimisation may be applied (Barceló, 1997a). Mathematical models of traffic flow behaviour are a prerequisite in transport planning studies.

The goal of a transport planning study might be to estimate the flow distribution generated by demand in a road network according to the mobility pattern, which represents traffic demand, and the conditions of use of the network, which represent the capacity offered by the network. The models used are called traffic assignment models.

The conditions of use of the network are modelled using behavioural route choice models, congestion models and the dependency of the travel time with the volume. The route choice is modelled considering Wardrop's principles (Wardrop, 1952), which formalise the concept of network equilibrium. Wardrop's first principle states that *"The journey times in all routes actually used are equal and less than those which would be experienced by a single vehicle on any unused route".* Each user non-cooperatively seeks to minimise the cost of his or her trip. The traffic flows that satisfy this principle are usually referred to as "user equilibrium" flows, since each user chooses the route that is best. Specifically, a "user-optimised" equilibrium is reached when no user can lower the cost of his or her trip through unilateral action. Wardrop's second principle, "*At equilibrium the average journey time is minimum",* implies that each user behaves cooperatively in choosing his own route to ensure the most efficient use of the whole system. Traffic flows satisfying Wardrop's second principle are generally deemed "system optimal".

The road network is modelled using a graph, where nodes $n \in N$ represent origins, destinations and intersections between road segments, and arcs $a \in A$ represent the infrastructure. The volume in arc a is represented by v_a, and $s_a(v)$ represents the cost of use arc a (where v is the vector of the volumes of all the arcs in the network). One hypothesis of modelling is that $s_a(v)$ is monotone, that is, that

$$[s(v') - s(v'')]T\,(v' - v'') \geq 0\ ,\ \forall \text{ feasible } v', v''$$

is continuous and differentiable. This hypothesis is necessary considering the properties of the mathematical model, which is, on the other hand, consistent with the experience.

To simplify the notation, the i-th represents the origin-destination pair (r, s) and g_i represents the demand or number of trips from zone r to zone s ($g_i = t_{rs}$).

The demand from one origin to one destination g_i, $i \in I$, where I is the set of all origin-destination pairs, can use the paths or route k, $k \in K_i$, where K_i is the set of all feasible routes of OD i-th. The flow in path k, h_k satisfies the following flow conservation and non-negativity conditions:

$$\sum_{k \in K_i} h_k = g_i, \quad \forall i \in I$$
$$h_k \geq 0, \quad \forall k \in K_i, \ \forall i \in I \qquad \text{(E 2.4)}$$

The flow in arc *a* is determined by

$$v_a = \sum_{i \in I} \sum_{k \in K_i} \delta_{ak} h_k, \quad \forall a \in A$$
where (E 2.5)
$$\delta_{ak} = \begin{cases} 1 \text{ if arc a belongs to path k} \\ 0 \text{ otherwise} \end{cases}$$

The cost of each path s_k is the sum of all costs of all arcs that belongs to path *k*:

$$s_k = \sum_{a \in A} \delta_{ak} s_a(v), \quad \forall k \in K_i \ , \forall i \in I \qquad \text{(E 2.6)}$$

and u_i is the minimum cost of OD pair *i*-th:

$$u_i = Min \{ s_k, \quad \forall k \in K_i \}, \forall i \in I \qquad \text{(E 2.7)}$$

Then, the formalisation of the user equilibrium concept implied in Wardrop's first principle is determined by the pattern of flows (*v, u*) that satisfies the following constraints:

$$h_k [s_k - u_i] \quad , \quad \forall k \in K_i \forall i \in I$$
$$s_k - u_i \geq 0 \quad , \quad \forall k \in K_i \forall i \in I \qquad \text{(E 2.8)}$$

The network equilibrium model is formulated by supposing that for every O/D pair Wardrop's user optimal principle is satisfied, or in other words, that all the used directed paths are of equal cost, that is:

$$s_k^* - u_i^* = \begin{cases} = 0 & if \ h_k^* \geq 0 \\ \geq 0 & if \ h_k^* = 0 \end{cases} \quad k \in K_i, i \in I \qquad \text{(E 2.9)}$$

Over the feasible set (E 2.4) and (E 2.5) . This network equilibrium model may be restated in the form of a variational inequality:

$$(s_k^* - u_i^*)(h_k - h_k^*) \geq 0 \quad k \in K_i, i \in I \qquad \text{(E 2.10)}$$

where h_k is any feasible path flow. If $h^*_k>0$, then $s^*_k=u^*_i$ since h_k may be smaller than h^*_k, if $h^*_k=0$ then the inequality is satisfied when $s^*_k-u^*_i \geq 0$.

By summing over $k \in K_i$, and $i \in I$, and taking into account constraints (E 2.4) and (E 2.5) when the demand g_i is constant, model (E 2.10) can be reformulated as follows (Fisk and Boyce, 1983; Magnanti, 1984; Dafermos, 1980):

$$s\left(v^*\right)^T\left(v-v^*\right)\geq 0 \qquad \text{(E 2.11)}$$

Which is the variational inequality formulation derived by Smith (1979).

When the user cost functions are separable, that is, they depend only on the flow in the link: $s_a(v)=s_a(v_a)\ a\in A$, and demands g_i are considered constant, independent of travel costs, the variational inequality formulation has the following equivalent convex optimisation problem (Patriksson, 1994; Florian and Hearn, 1995):

$$\begin{aligned} &Min \sum_{a\in A}\int_0^{v_a} s_a(x)dx \\ &s.t.\ \sum_{k\in K_i} h_k = g_i, \forall i\in I \\ &\quad h_k \geq 0, k\in K_i, i\in I \end{aligned} \qquad \text{(E 2.12)}$$

and the definitional constraint of v_a (E 2.5).

Although the traffic assignment problem is a special case of non-linear multi-commodity network flow problem, and may be solved by any of the methods used for the solution of this problem, more efficient algorithms for solving this problem, based on an adaptation of the linear approximation method of Frank and Wolfe (1956) have been developed in the past years (Leblanc *at al*., 1975 and Florian, 1976). Other efficient algorithms based on the restricted simplicial approach have been developed more recently by Hearn et al. (1987) or on an adaptation of the parallel tangents method (PARTAN) (Florian et al., 1987).

Assignment models in transport planning are macroscopic models in which traffic flow is considered to be an aggregation; the models do not take into account the individual parts, that is, the vehicles.

Within macroscopic models, transport planning analyses use a static view of mobility patterns. In this static view, mobility patterns are calculated as an average over a relatively long period, because the aim of the analysis is to evaluate the expected use of the road infrastructure over a long-term period. Figure 2.3 depicts this static view of demand, in which the number of trips for each OD pair is distributed in a homogeneous way in each time interval. This view corresponds to the objectives of an analysis of the use of a road infrastructure over a long-term period.

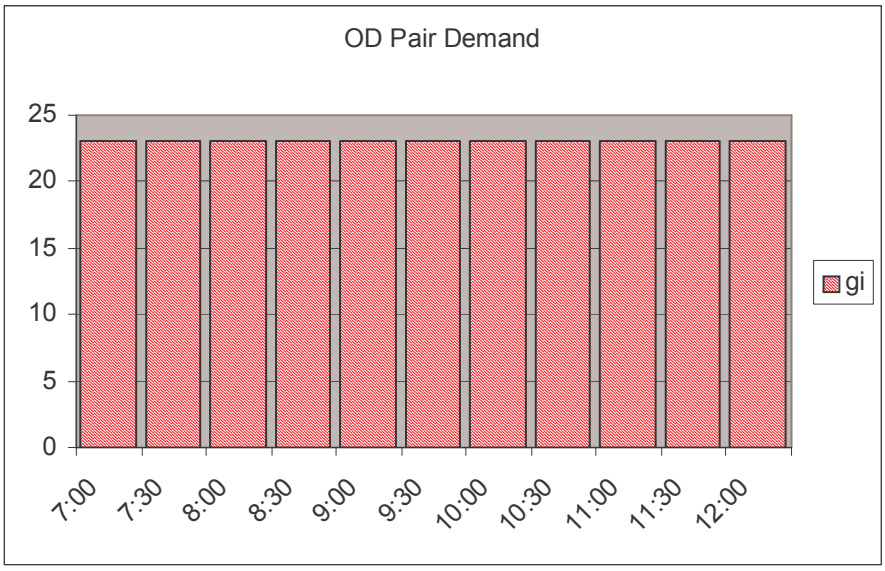


Figure 2.3. Static Demand

2.1.3 SCHEMES OF TRAFFIC MANAGEMENT AND INFORMATION SYSTEMS IN REAL-TIME

Traffic management and traffic information systems in real time have to deal with short time intervals in which neither the demand nor the flow are constant or homogenous. Demand and flow behave dynamically, that is, they are time-dependent. The definition of traffic management given by Barceló (1991) is broader than the classic conception of traffic management, which takes action over time, including control over space, for instance, the redistribution of flow by means of rerouting actions in which alternatives routes are proposed. Therefore, traffic management applications require dynamic modelling that represents flow variation over time.

Advanced traffic management systems (ATMS) and advanced traffic information systems (ATIS) have a dynamic view of traffic and make use of the infrastructures present in road networks that enable information to be gathered on-line, such as detector measures. This dynamic view of traffic involves a dynamic view of mobility matrices, that is, the mobility matrix is time-dependent.

The number of trips g_i is not a static but a dynamic (time-dependent) value as depicted in Figure 2.4.

All proposals for advanced traffic management and control systems based on telematic technologies highlight the importance of the short-term prediction of traffic flow evolution, which is equivalent to the short-term prediction of the network state for correct decision-making in traffic management, information dissemination to users, etc. Several system architectures of those proposed have been evaluated as part of European projects in recent years, including ARTIS (ARTIS, 1994), PETRI (PETRI, 1996) and CAPITALS (CAPITALS, 1998).

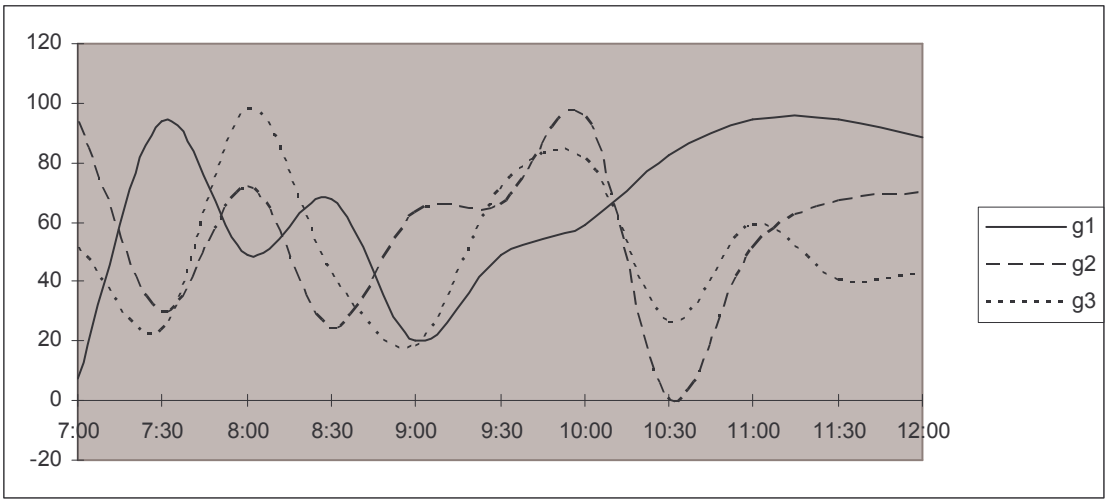


Figure 2.4. Dynamic demand

The European PETRI project may be used as an example that shows how the short-term prediction of the network state is required. This project involved the design and implementation of a traffic management and information dissemination system in real time, using advanced telematic technologies. Figure 2.5 depicts the conceptual scheme for the project.

For this traffic management and information system scheme, it is assumed that there is a layout of detectors that provide detector measures in real time available in the road network. These real-time detector measures are the input of the coordination module, which carries out the following two basic functions:

Using the network state estimation model, it estimates and identifies the current network state and receives as input the detector measurements in real-time.

Using the current network state, it predicts the short- or medium-term network state. The network state forecast may be interpreted as the normal or natural evolution of the network state without the application of further control or management policies.

The output of the coordination module is the input of the traffic pattern recognition module, which identifies and characterises bottlenecks by identifying the main traffic patterns, assuming there are no new preventive traffic control and management actions. The traffic pattern recognition module characterises the bottleneck, gives its location and start time, and sends the information to the diagnosis module.

The diagnosis module proposes a set of traffic control and management actions designed to either prevent the negative effects of bottlenecks or find solutions to alleviate the effects of an existing bottleneck. The actions are proposed to the operator of the traffic management and control centre, and it is then the operator who decides whether to accept them.

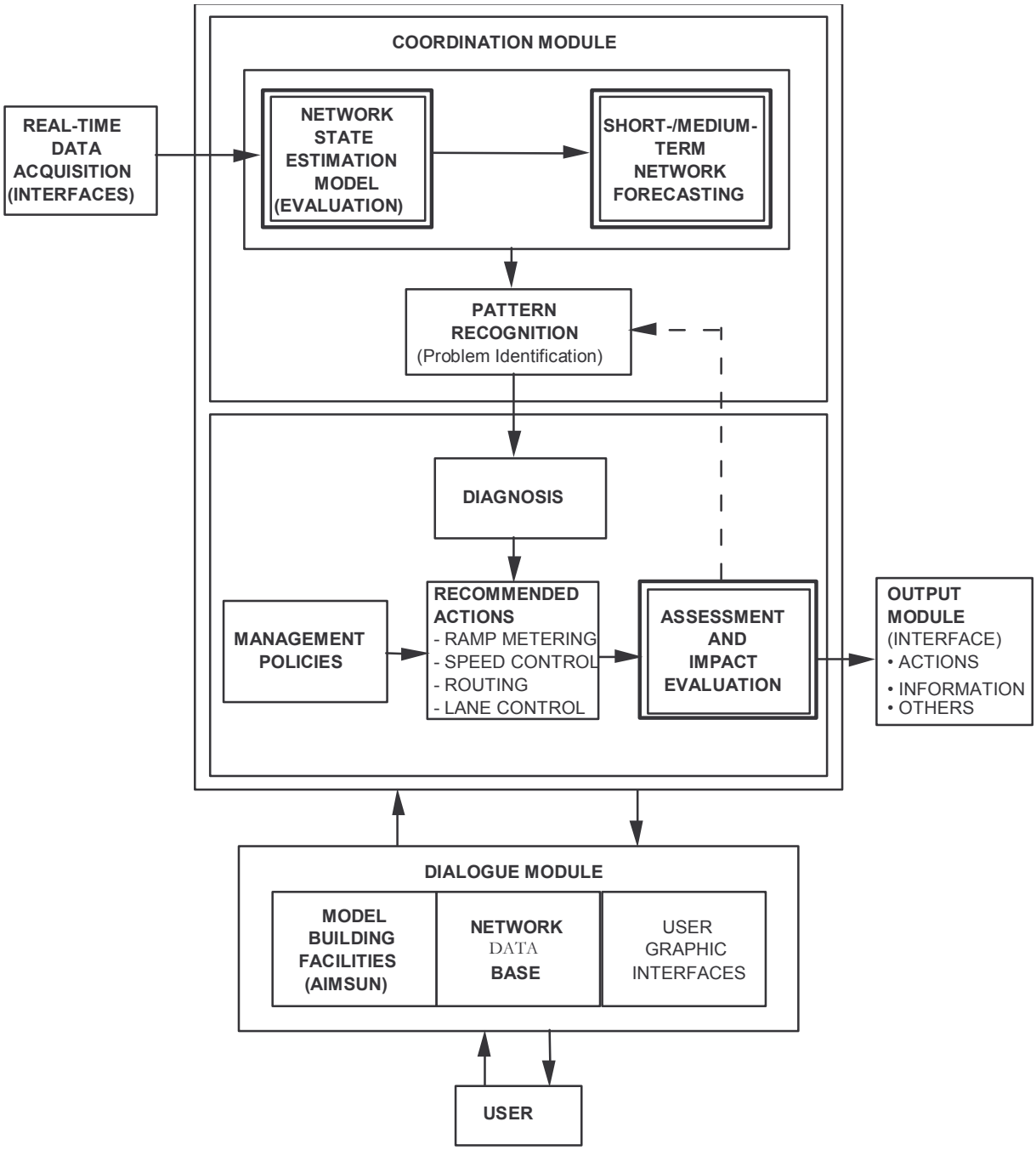


Figure 2.5. PETRI conceptual scheme

Figure 2.6 depicts a scheme that represents the functional cycle of the system. The detector measurements, which are aggregated every Δt units of time, are the input of the system. To clarify the functional cycle, the sequence of activities can be grouped into three series or chains of events. The first sequence of events represents the current situation, whilst sequences (2) and (3) represent the situation forecast. Sequence (1), which represents the current network state, the detector measurements and the origin-destination matrix, which is estimated in the previous time interval and represents traffic demand in the current time interval, are the input of the dynamic network loading module, which forecasts traffic flow in the whole network that is supervised by the system. These traffic flow forecasts are the input of the microscopic simulator

that, running the simulation, produces the current traffic conditions or state. The problem identification module of analyse this current state and present it to the operator using the dialogue module.

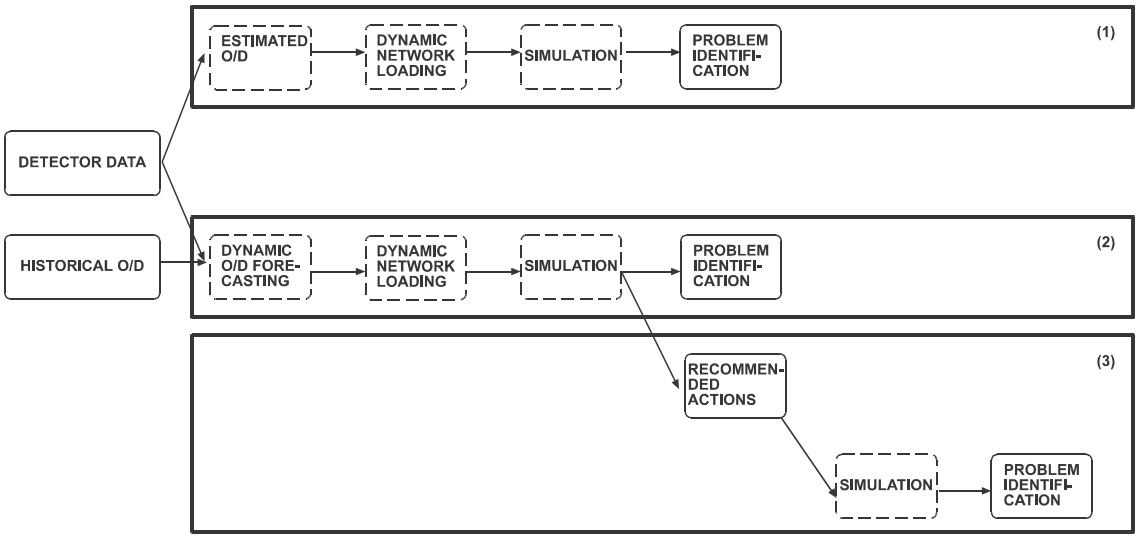


Figure 2.6. Schema of activities or processes in the PETRI

The sequence of events (2) concerns to forecasted traffic state for the next time intervals. The current detector measurements and the set of historical origin-destination matrices are the input for this chain. All this data is the input for the dynamic OD forecasting that produces, as output, the origin-destination matrix forecast for subsequent intervals. From this point the next events are the same as sequence (1), but instead to use the current situation, sequence (2) uses the traffic data forecasted for the next time interval. That is, sequence (1) represents the current situation, whereas sequence (2) is an attempt to predict the evolution of the current situation for subsequent time intervals, assuming that the current conditions will continue without any new traffic control and management action. When the problems are detected and identified, the recommended actions module proposes a set of control and management policies or actions and simulates them to determine the evolution of the traffic conditions if they were to be applied. This belongs to sequence of events (3), and then the operator, comparing the evolution of network state with (obtained at the end of sequence (3)) and without (obtained at the end of sequence (2)) new policies, can reject or accept the proposed actions.

Another scheme, where the mechanism of traffic prediction is necessary, is developed in the CAPITALS project. For historical and technical reasons, large cities usually have several traffic control and management centres. In most cases, these control and management systems operate locally, that is, they take into account only the traffic conditions of their own area, regardless of the traffic conditions in adjacent areas. The absence of communication between different control and management centres gives rise to conflicts when there is a dynamic change in demand that generates variations in traffic flow interchanges between adjacent zones.

CAPITALS attempts to avoid, or alleviate, the conflicts originated by the lack of coordination between control and management centres. To achieve this aim, it designs and implements a supervisor system with the following functions:

- It enables dialogue and the exchange information between traffic control centres.
- It coordinates the different zones' strategies based on the information available for each zone.
- It recommends changes and modifications in the control and management strategies of each zone, based on information on the variation in traffic demand and traffic flow forecast.

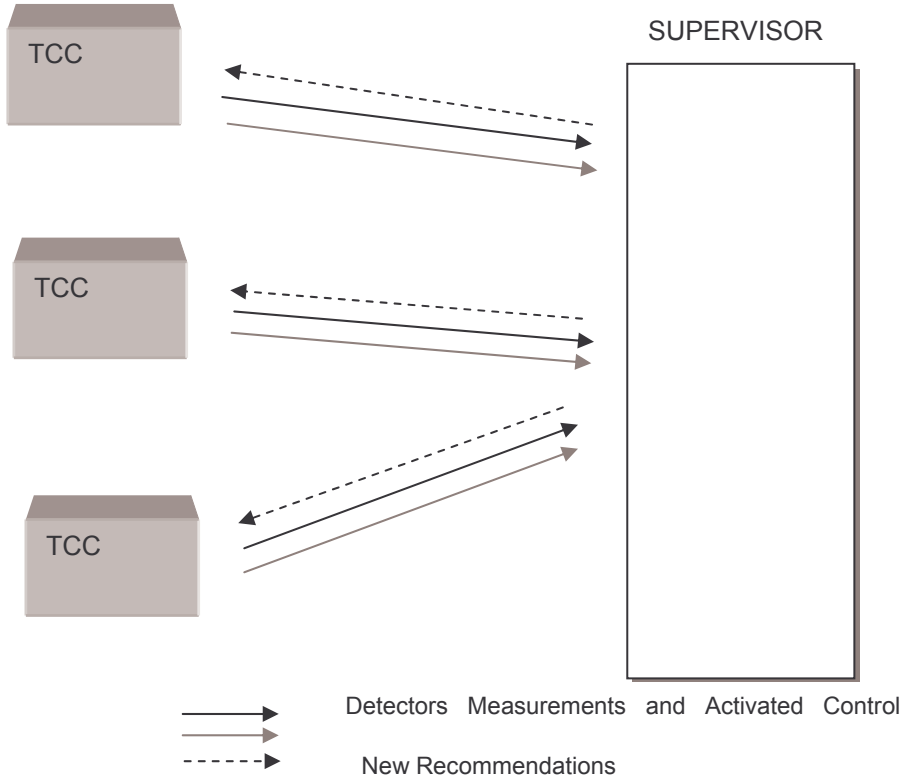


Figure 2.7. Generic view of the system's components

The functional architecture of the supervisor system is shown in Figure 2.7. Local traffic control centres (TCCs) send detection data that is aggregated every Δt time unit (one of the system's design parameters, which is defined by the operator). The supervisor uses this information to estimate the network state in each zone. The detector measurements for each zone, in addition to the historical information for the current time interval, comprise the input in the short-term prediction module that is used to evaluate the short-term traffic state in each zone. On the basis of a comparison between the current and forecast traffic state in each zone, recommendations are made to the TCCs, which then adapt local traffic control plans to the expected forthcoming variation in traffic demand.

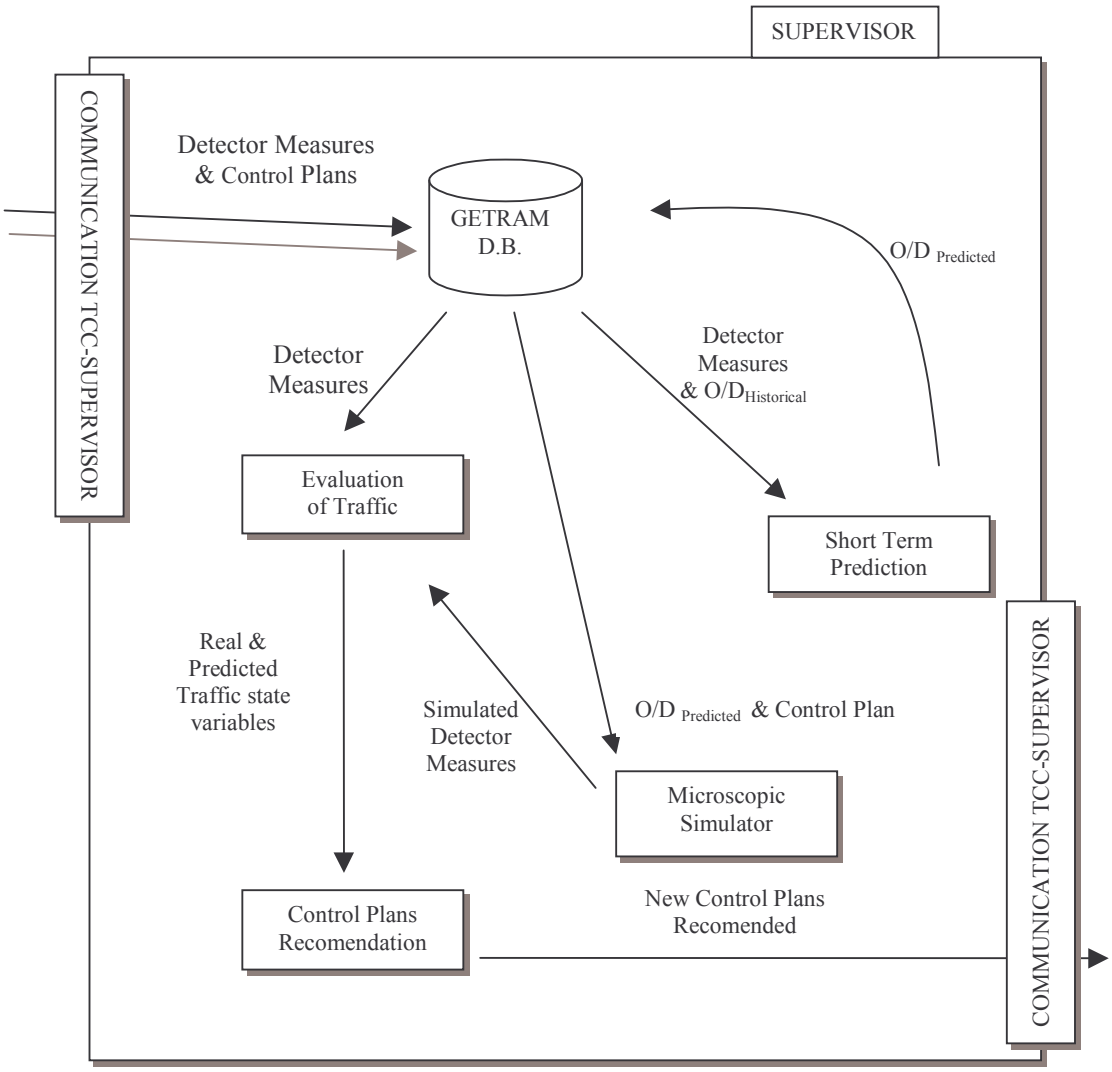


Figure 2.8. Functional design of the supervisor in CAPITALS

The supervisor is composed of the following modules, as depicted in Figure 2.8:

- The GETRAM database is the traffic network database (Grau et al., 1993). It was selected because it contains the model for the microscopic AIMSUN simulator (Barceló, 1994 and 1995; Ferrer, 1993) that completes this architecture. This database contains the information required to model the scenario considered by the system, namely, the topology of the network and traffic, and the information equipment, such as, for instance, detectors, traffic control plans, turning movements, intersections, on and off ramps, variable message signs and mobility patterns or traffic demand).

- The traffic evaluation module contains the models and algorithms that are needed to estimate traffic conditions in the network by combining the real detector measurements and the microscopic simulator's detector emulation.

- The short-term prediction module contains the module and algorithms needed to predict the mobility pattern for the subsequent time interval, taking into account the historical data and the current detector measurements. The mobility pattern, expressed as an origin-destination matrix, is used to estimate the traffic flow interchange between adjacent zones, as depicted in Figure 2.9.

- The microscopic simulator is used to evaluate the evolution of the forecast traffic state, assuming that no new control and management policies are applied.

- The control plan recommendation module is based on the evolution of the forecast traffic state, a decision-support model relative to the traffic state and a set of strategies. It recommends changes to the strategies, so that these are better suited to the changes in the traffic state that are expected.

- The communication module implements the interfaces between the supervisor and the TCCs, and it manages the exchange of information between them. Every Δt, a TCC sends the supervisor information on the detector's traffic measurements and on the current active control plan. In the other direction, the supervisor sends the TCC the recommended strategies.

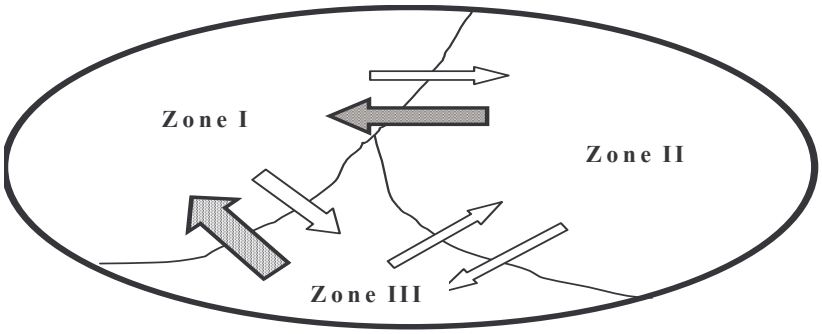


Figure 2.9. Flow interchange between adjacent zones

The diagram in Figure 2.10 illustrates the cyclic processes that are performed by the supervisor at each time interval.

a) At the end of time intervals t-1, t, t+1,…, the supervisor receives the aggregated detection data and the desegregated control data for the time period.

b) The detector data for time interval t-1 and the historical OD matrix for this time period are the input used by the prediction module to estimate the OD matrix forecast for time period t.

c) The OD matrix forecast for time period *t* and the control data are the input for the simulation model of the supervisor, which estimates the current network state at the beginning of time period *t*.

d) The flows estimated by the simulator for the current network state are compared to the flows measured, to determine whether the estimation of the current state is acceptable or not.

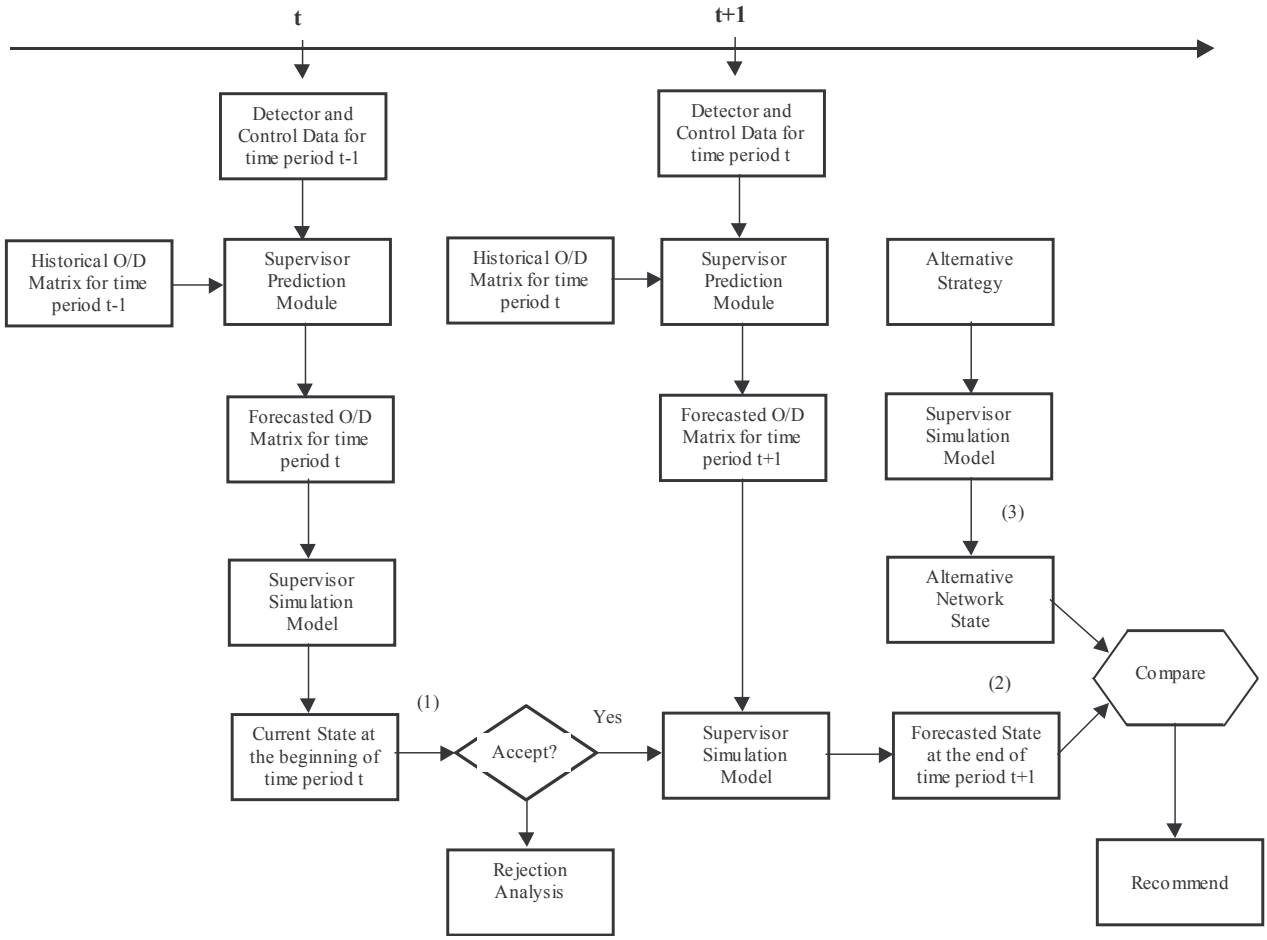


Figure 2.10. Cyclic processes performed by the supervisor in CAPITALS

e) To estimate the OD matrix forecast for time period *t*+1, steps a and b are repeated with the data and historical OD matrix for time period *t*.

f) Starting from the estimated current state at the beginning of time period *t*, a simulation is performed to estimate the state forecast for the end of time period *t*. The simulator communicates with two traffic control systems (MOT and SAINCO), which exploit the aforementioned feature of the AIMSUN simulator to ensure a one-to-one correspondence with the real system.

g) By following the same process but repeating the simulation with the OD matrix forecast for time period *t* and the alternative strategy, one obtains the alternative state.

h) The comparison between the alternative state and the forecast state determines the type of recommendation.

The Phase I and interphase process comprises tuning the system, refining the system parameters, analysing the rejections, determining how to overcome the rejection causes identified, etc. These tasks are carried out by the supervisor's logging feature, which enables the situation leading up to the rejection to be reproduced.

2.2 SCHEMES FOR THE SHORT-TERM PREDICTION OF TRAFFIC DEMAND

All proposals for advanced traffic management systems based on telematic technologies agree on the importance of a short-term forecast of the evolution of traffic flows or, equivalently network state, as a proper basis for management decisions. Consequently a lot of effort has been devoted to researching and developing proper forecasting procedures. Perhaps the most relevant of these efforts is the EU's DYNA project (Gunn, 1994; Ben-Akiva et al., 1994; Inaudi et al., 1994). Although the results of the project cannot be applied or extrapolated to complex urban structures, other models that are more suited to complex networks have been developed, by Cascetta (1993) and Barceló (1997b), for example. Unfortunately, these models do not appear to be appropriate for full dynamic applications, and so we had to look elsewhere in our search for a suitable prediction model. The promising features of neural networks, which makes them suitable for use as predictive tools (Baldi and Hornik, 1995), encouraged us to explore this approach.

2.2.1 SCHEME FOR THE SHORT-TERM PREDICTION OF TRAFFIC DEMAND IN DYNA

The methodology proposed in the EU's DYNA project can be applied to traffic networks whose structure is linear, such as a motorway. The demand prediction is expressed in terms of incoming traffic flows and it has a smoothing effect on the deviations between the observed traffic flows and the historical traffic flows.

The methodology proposed in the DYNA project, which assumes a time interval *h,* is to predict short-term traffic demand at time interval *h+i*, is function the traffic demand predicted at interval *h* and the historical traffic demand at interval *h+i,* such that

$$d^{pred}[h+i] = \Phi[\, d^{est}[h],\, d^{hist}[h+i]] \quad i>0$$

where

$d^{pred}[h]$ is the OD matrix to be predicted at time interval *h;*

$d^{est}[h]$ is the OD matrix estimated at time interval *h*; and

$d^{hist}[h]$ is the historical OD matrix at time interval *h.*

Let us assume that

$d^{pred}[h+i]$ is the OD matrix predicted at time interval $h+i$, $i>0$;

$e^{pred}[h+i]$ are the incoming traffic flows predicted at time interval $h+i$, $i>0$;

$s^{pred}[h+i]$ is the probability matrix that each element (k,j) represents the probability of one vehicle that has centroid k as its origin and enters at interval $h+i$ $(i>0)$ having centroid j as its destination;

$e_d^{hist}[h]$ are the historical incoming flows at interval h of day d; and

$e_d^{obs}[h]$ are the observed incoming flows at interval h of day d.

The process of short-term traffic demand prediction is then defined as follows:

Step 1.

$$e^{pred}[h+i] = \Phi[\, e^{hist}[h],\ e^{hist}[h+i],\ e^{obs}[j],\ j \leq h] \quad i>0$$

Different filters Φ were specified and verified with real data, but the most used filter was

$$\Delta e_t[h] = \alpha_1 (e_d^{hist}[h] - e_d^{obs}[h]) + (1 - \alpha_1)\, \Delta e_t[h-1]$$

$$e_d^{pred}[h+i] = e_{d-1}^{hist}[h+i] - \Delta e_d[h], \quad i > 0$$

where d represents the current day and the historical data is obtained as follows:

$$e_d^{hist}[h] = \alpha_2\, e_{d-1}^{hist}[h] + (1 - \alpha_2)\, e_d^{obs}[h]$$

Parameters α_1 i α_2 must be calibrated and must accomplish

$$0 \leq \alpha_1 \leq 1 \text{ i } 0 \leq \alpha_2 \leq 1$$

Step 2.

Once the incoming traffic flow has been predicted, the probability matrix s is determined using the same mechanism, such that

$$\Delta s_d[h] = \beta_1 (s_d^{hist}[h] - s_d^{est}[h]) + (1 - \beta_1)\, \Delta s_d[h-1] \text{ and}$$

$$s_d^{pred}[h+i] = s_{d-1}^{hist}[h+i] - \Delta s_d[h], \quad i > 0$$

where d represents the current day and the historical data is obtained as follows:

$$s_d^{hist}[h] = \beta_2\, s_{d-1}^{hist}[h] + (1 - \beta_2)\, s_d^{est}[h]$$

Parameters β_1 i β_2 must be calibrated and must accomplish

$$0 \le \beta_1 \le 1 \text{ i } 0 \le \beta_2 \le 1$$

Step 3.

Finally, the result of the prediction is

$$d^{pred}[h+i] = e^{pred}[h+i] * s^{pred}[h+i],\ i>0$$

2.2.2 SCHEME FOR SHORT-TERM PREDICTION USING NEURAL NETWORKS

Ashok and Ben-Akiva (1993) propose a different framework for the real-time estimation/prediction of time-dependent OD matrices. They formulate the problem as a Kalman filter, in which the state vector consists of deviations of OD flows from prior estimates based on historical data. The conceptual framework taken from the reference is depicted in Figure 2.11. In this approach, if x_{rh} are the number of vehicles in the *r*-th OD pair that leave the origin during interval *h*, and x_{rh}^H the corresponding best historical estimate, to formulate the transition equation we assume the hypothesis that the deviations in OD flows from a historical base at period *h* can be related to the deviations in OD flows in previous time periods, written in autoregressive form as

$$x_{r,h+1} - x_{r,h+1}^H = \sum_{p=h-q'}^{h} \sum_{r'=1}^{n_{OD}} f_{rh}^{r'p} \left(x_{r'p} - x_{r'p}^H\right) + w_{rh}$$

where n_{OD} is the number of OD pairs, coefficients $f_{rh}^{r'p}$ describe the effect of the deviation $\left(x_{r'p} - x_{r'p}^H\right)$ on the deviation $\left(x_{r,h+1} - x_{r,h+1}^H\right)$, w_{rh} is a random error and q' is the number of lagged OD flow estimations assumed to affect the OD flow deviation in interval *h*+1. The author suggests computing the matrix $\mathbf{f_h^p}$ by means of linear regression models for each OD pair for each interval, assuming that the autoregressive process remains constant with respect to *h* and therefore the values of the matrix $\mathbf{f_h^p}$ only depend on the difference (*h* - *p*) and not on individual values of *h* and *p*. The problem can be simplified further by assuming a diagonal structure for the matrix, thereby ignoring interdependencies across OD flows.

The relationships between the traffic counts observed on link *l* during interval *h,* y_{lh} and OD flows x_{rp} are defined in terms of the following measurement equation:

$$y_{lh} = \sum_{p=h-p'}^{h} \sum_{r=1}^{n_{OD}} a_{lh}^{rp} x_{rp} + v_{lh}$$

where a_{lh}^{rp} is the fraction of the *r*-th OD flow that departed from its origin during interval *p* and is on link *l* during interval *h*, v_{lh} is the measurement error, and *p'* is the maximum number of time intervals taken to travel between any OD pair in the network.

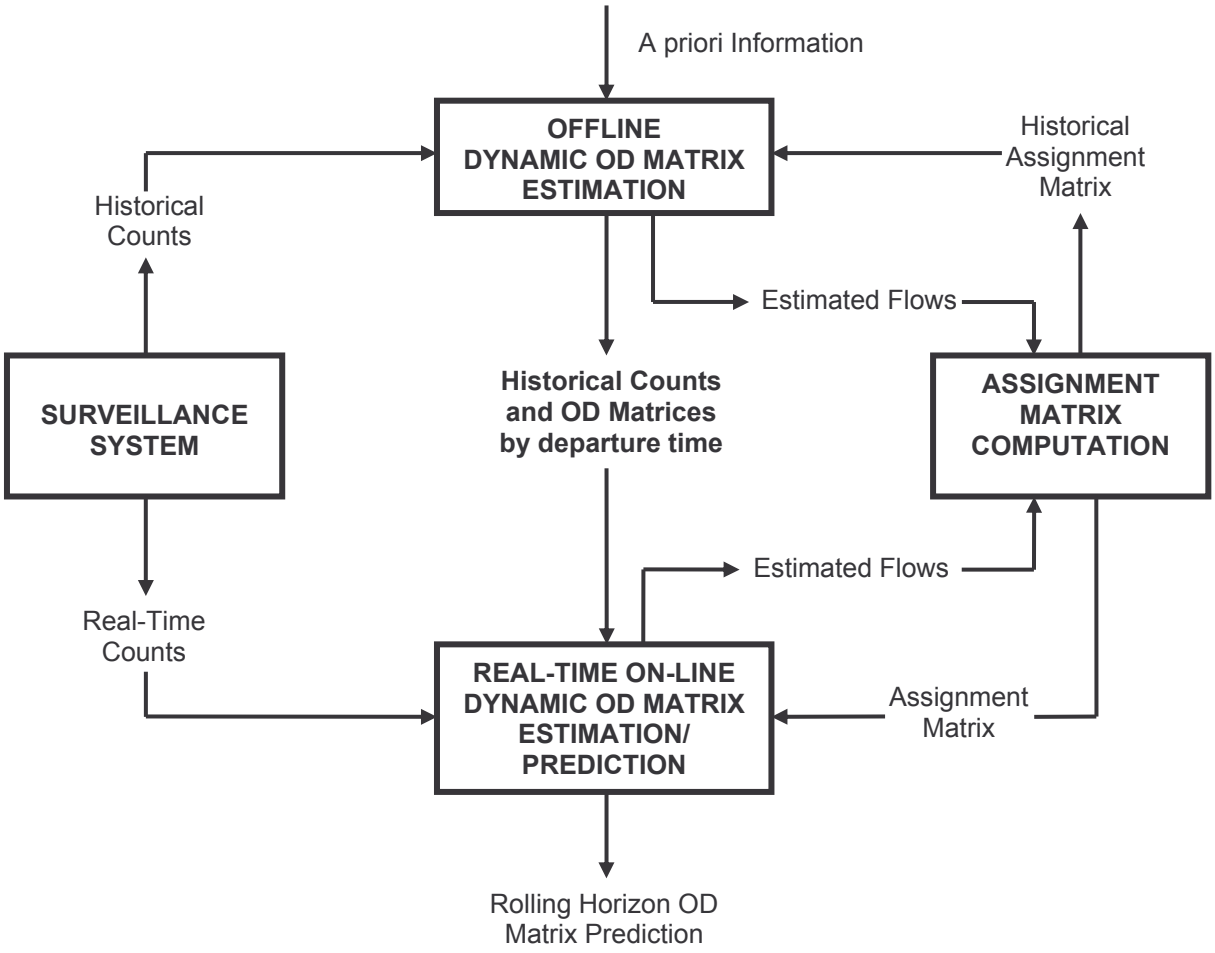


Figure 2.11. Conceptual framework for real-time estimation/prediction of OD matrices

This approach has been further developed and computationally tested by Ashok (2000) and Ben-Akiva (2002), and integrated in a dynamic traffic assignment context by Ben-Akiva et al. (2002). Nevertheless, when work began on this thesis only primary results were available, and the computational performance of the approach proposed remained unclear; therefore, we decided to take a different approach. The promising features of neural networks, which makes them suitable for use as predictive tools (Baldi and Hornik, 1995), encouraged us to explore them further, within a conceptual framework that shares certain concepts with that of Ashok and Ben-Akiva (1993), depicted in Figure 2.11.

To address the problem, we consider origins and destinations as pairs, *I* being the set of all OD pairs in the network, so if origin *r* and destination *s* are the *i*-th OD pair, g_i denotes the

corresponding entry of demand matrix *G*, which represents the total number of trips between origin *r* and destination *s*. Therefore,

$$OD_{(r,s)} = g_i \ , \ \ i = (r,s) \in I$$

The total number of trips between an origin *r* and a destination *s* is not a fixed value over time; it is a dynamic value (i.e. it is time-dependent), as depicted in Figure 2.4. According to this dynamic vision of demand, we can consider each of the OD matrix's components as a time series. Therefore, forecasting an OD matrix consists in performing the forecast for each component in the matrix, that is, in simultaneously forecasting many multivariate time series. Solutions to this problem that are based on classic forecasting methods, such as Box-Jenkins or Kalman filtering, have been proposed by several authors (Davis, 1993; Davis et al., 1994; Van der Ziipp and Hamerslag, 1996). The approaches proposed provide relatively good results for linear infrastructures, such as motorways, although it remains unclear whether they would provide reliable results in the case of more complex road networks. In several of the most promising cases (Davis, 1994), however, the computational task required practically invalidates their use in real-time applications in large-scale networks and makes it advisable to look for other methods.

Recently, when most of the research for this thesis had been completed, an alternative computational approach to the Ashok and Ben-Akiva (1993) approach was proposed by Bierlaire and Crittin (2004), who reformulated the model in terms of least squares. Their computational results seem to show that the drawbacks mentioned have been overcome.

Neural networks appear to be natural candidates for forecasting models, particularly if their easily parallelisable structure is taken into account and high computational speed is required to achieve the system's objectives. Further reasons to consider a neural network approach are the results reported by Chakraborty (1992) for multivariate time series analysis using neural networks and by Weigend (1992) in his evaluation of their predictive capabilities compared to other classic models.

A neural network model is generally characterised by the following three elements:

- The neuron's characteristics
- The topology of the neural network
- The rules of the training or learning process

2.2.2.1 NEURON CHARACTERISTICS

A neuron receives *n* values (X_i, where *i=1...n)* as input and produces one value *Y* as output. A neuron is characterised by *n* weights (W_i where *i=1...n)*, a nonlinear activation function *f(x)* and a bias or offset θ, as shown in Figure 2.12.

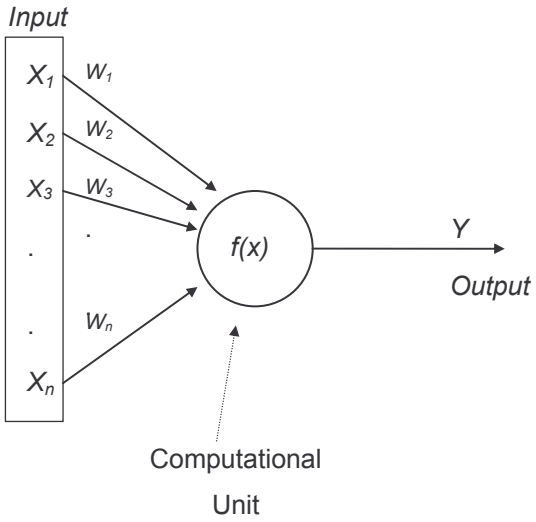
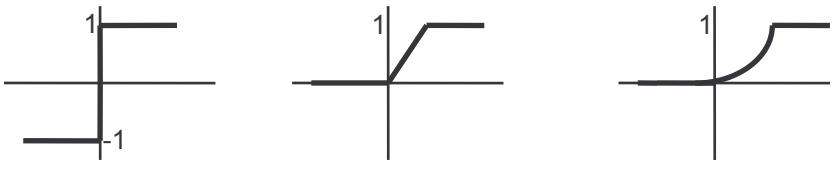


Figure 2.12. Neuron representation

The neuron output is determined by the following expression:

$$Y = f(\sum_{i=1}^{n} W_i \, X_i + \theta)$$

where *f(x)* is the nonlinear activation function of the neuron. Figure 2.13 depicts several generic examples of this activation function.



a) Hard Limiter b) Threshold Logic c) Sigmoid

Figure 2.13. Activation function shapes

In our case, the activation function selected is the sigmoid function $f(x) = \frac{1}{1+e^{-x}}$ (logistic function) or $f(x) = \tanh(x)$. The main difference between them is that the tanh output ranges from [-1,1], whereas the logistic output ranges from [0,1]. It is often said that when predicting a target around a zero mean, tanh is likely to be more suitable.

The sigmoid function, as defined by Hecht-Nielsen (1989), is a bounded differentiable real function that is defined for all real input values. It rapidly approaches asymptotically fixed finite upper and lower limits as its argument gets larger or smaller respectively, and this limited dynamic range effectively implements noise suppression and cut-off, as Másson (1990) shows. The sigmoid function makes use of the sigmoid rule—one of the most frequently used nonlinear activation rules—and as such is particularly suited to our problem, in which there are continuous inputs and outputs.

Although in certain cases there might be a relationship between the physical system modelled in terms of a neural network and the topology of the neural network, in such way that a physical interpretation of the neurons could be provided, in the most typical applications of neural networks this relationship does not exist and no physical interpretation of what a neuron or a layer of neurons means or captures in terms of the physical system can be provided. Our interest in a neural network approach lies in its ability to provide a computational mechanism when no specific structures can be identified, thus providing the basis for a model that represents the system. This is precisely the case in the OD estimation and the reason why we took this approach. In fact, the approach is perhaps similar to a regression approach, in which no specific structural relationship between predictors and response variables is assumed and coefficients only provide a de facto relationship. In fact, our case can be said to exploit the well-known relationships between multilayered neural networks and multiple regression models.

2.2.2.2 NEURAL NETWORK TOPOLOGY

The neural network topology determines the connection between the various computational units. The neural network that we used was a multilayer perceptron (Hecht-Nielsen, 1989), whose topology corresponds to a feed-forward network in which the neurons (i.e. the processing units) are arranged in layers. All the neurons in a layer are connected directly to all the neurons in the next layer, as shown in Figure 2.14. The input layer, denoted by layer 0, contains no real neurons and its purpose is to spread the input to the neurons in the first hidden layer.

The hidden layers are numbered from 1 to L-1, and the output layer is L. In general, the k-th layer contains N_k neurons, and therefore the input layer has N_0 elements and the output layer has N_L neurons. A neuron n in layer k is connected to all the neurons in layer k-1 through several connections (exactly N_{k-1}), each of which is associated with a weight. If we organise this weight in a vector $W^{(n)}(k)$, the neuron has the corresponding bias or offset $b^{(n)}(k)$. In a multilayer perceptron, these elements are static; they determine the topology of the neural network, and there is a dynamic element that determines the state of each layer when the input is propagated through the neural network. If we apply the input, represented by a vector IN with exactly N_0 components, the neural network propagates from input to output through every layer. Each k-th layer takes on a precise state, represented by a vector $S(k)$ with N_k components, which represents the output of each N_k neuron in this layer.

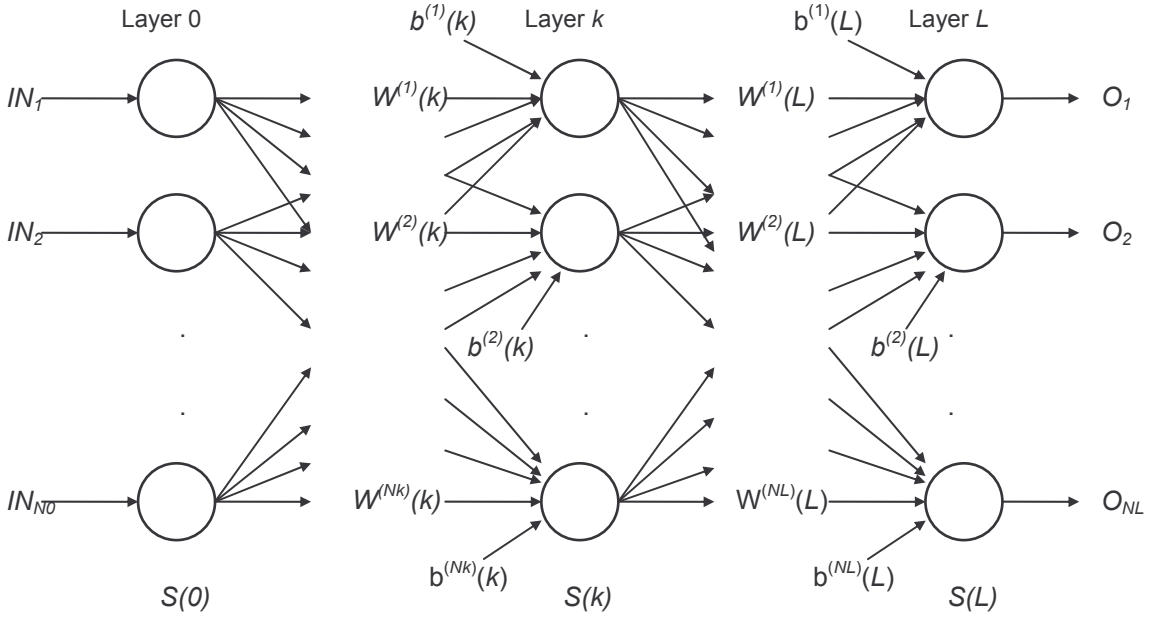


Figure 2.14. A multilayer neural network

As shown above, layer 0 contains no real neurons, its sole function being to spread the input to layer 1, so *S(1)* is exactly *IN* and the output of the network will be *S(L)*, which corresponds to vector *O*. The state of the *n*-th neuron of the *k*-th layer is computed with the feed-forward rule, such that

$S_n(k) = f\,(\, S(k\text{-}1)^T\, W^{(n)}(k) + b^{(n)}(k))$ in vectorial form

or $S_n(k) = f\,(\, \sum(S_i(k\text{-}1)\, W_i^{(n)}(k) + b^{(n)}(k)))$ in scalar form

where *f* is the activation function of the neuron. The procedure to determine the output of the neural network for a given input is called the feed-forward procedure and is described by the following algorithm:

Feed-Forward Procedure
  for each neuron $n = 1 \ldots N_0$
    $S_n(0) = IN_n$ {state of *n*-th neuron in layer *0*}
  end for
  for each layer $k = 1 \ldots L$
    for each neuron $n = 1 \ldots N_k$
      $S_n(k) = f\,(\, S(k\text{-}1)^T\, W^{(n)}(k) + b^{(n)}(k))$ {state of *n*-th neuron in layer *l*}
    end for
  end for
end procedure

2.2.2.3 TRAINING PROCESS

The training algorithm used is an ad hoc version of the back propagation algorithm described by Hecht-Nielsen (1989). It is a supervised learning process, given that the weights of the different neuron connections are iteratively changed with reference to a set of predefined patterns specified as a set of input-output pairs. At each step, the computational error is estimated to be

$$E = \frac{1}{\mathrm{P}\, N_L} \sum_{p=1}^{P} \left\| t^{(p)} - S^{(p)}(L) \right\|^2$$

$$E = \frac{1}{\mathrm{P}\, N_L} \sum_{p=1}^{P} \sum_{n=1}^{N_L} \left(tn^{(p)} - Sn^{(p)}(L) \right)^2$$

where P is the number of desired outputs, $t^{(p)}$ is the p-th desired output and $S^{(p)}(L)$ is the p-th output produced by the neural network. Back propagation tries to minimise the total squared error E using the gradient algorithm shown below.

The gradient is computed as

$$\frac{\partial E}{\partial Wi^{(n)}(L)} = -\sum_{p=1}^{P} \delta n^{(p)}(L) \cdot Si^{(p)}(L-1)$$

$$\text{where:}$$

$$\delta n^{(p)}(L) = \frac{2}{P \cdot N_L} \left(tn^{(p)} - Sn^{(p)}(L) \right) \left(1 - \left[Sn^{(p)}(L) \right]^2 \right)$$

Weights in the output layer are modified according to the delta rule

$$\Delta Wi^{(n)}(L) = \eta \sum_{p=1}^{P} \delta n^{(p)}(L) \cdot Si^{(p)}(L-1)$$

$$\text{where } \eta \text{ is a parameter}$$

For other layers k = 1 ... L-1, the rule is defined as

$$\frac{\partial E}{\partial Wi^{(n)}(k)} = -\sum_{p=1}^{P} \delta n^{(p)}(k) \cdot Si^{(p)}(k-1)$$

$$\text{where:}$$

$$\delta n^{(p)}(k-1) = \left(1 - \left[Sn^{(p)}(k-1) \right]^2 \right) \cdot \sum_{j=1}^{Nk} \left[\delta j^{(p)}(k) \cdot Wn^{(j)}(k) \right]$$

and weights are modified by

$$\Delta W_i^{(n)}(k) = \eta \sum_{p=1}^{P} \delta_n^{(p)}(k) \cdot S_i^{(p)}(k-1)$$

where η is a parameter

The back propagation algorithm is therefore

Back Propagation Procedure

{Feed-Forward Phase}

for each layer *k* = 1 ... *L*

for each neuron *n* = 1 ... N_k

for each input *p* = 1 ... *P*

$$S_n^{(p)}(k) = f\left(S^{(p)}(k\text{-}1)^T \, W^{(n)}(k) + b_{(n)}(k) \right)$$

end for

end for

end for

{Error Computation Phase}

for each neuron *n* = 1 ... N_l

for each input *p* = 1 ... *P*

$$\delta_n^{(p)}(L) = \frac{2}{P \cdot N_L} \left(tn^{(p)} - S_n^{(p)}(L) \right) \left(1 - \left[S_n^{(p)}(L) \right]^2 \right)$$

end for

end for

{Error back-propagation Phase}

for each layer *k* = *L-1* ... 1

for each neuron *n* = 1 ... N_l

for each input *p* = 1 ... *P*

$$\delta_n^{(p)}(k) = \left(1 - \left[S_n^{(p)}(k) \right]^2 \right) \cdot \sum_{j=1}^{Nk} \left[\delta_j^{(p)}(k+1) \cdot W_n^{(j)}(k+1) \right]$$

end for

end for

end for

{Step Phase}

for each layer *k* = 1 ... *L*

for each neuron *n* = 1 ... N_l

$$\Delta b_n(k) = \eta \sum_{p=1}^{P} \delta_n^{(p)}(k)$$

for each weight *i* = 1 ... N_{k-1}

$$\Delta W_i^{(n)}(k) = \eta \sum_{p=1}^{P} \delta_n^{(p)}(k) \cdot S_i^{(p)}(k-1)$$

end for

end for

end for

{Weight-updating Phase}

for each layer *k* = 1 ... *L*

for each neuron *n* = 1 ... N_l

$$b_n(k) = b_n(k) + \Delta b_n(k)$$

for each weight *i* = 1 ... N_{k-1}

$$W_i^{(n)}(k) = W_i^{(n)}(k) + \Delta W_i^{(n)}(k)$$

end for

end for

end for

end procedure

2.2.2.4 PREDICTION

The prediction process forecasts the OD matrix in the next interval from the detector measurements collected and the historical OD matrix, using a multilayer perceptron neural network, as displayed in Figure 2.15. The predictor process receives, as input, the detector measurements V_a (*a* being the set of arcs) of where the detectors are located, together with the historical demand $OD_i^{historical}$, which represents the demand between the *i*-th origin-destination pair. To run the neural network optimally, this input must be normalised, which can be done by following these rules: 1) the detector measurements can be divided by the maximum capacity of each arc, i.e. V_a / V_a^{max} and 2) the different demands of the historical OD matrix can be divided by the maximum demand, i.e. $OD_i^{historical}$ / OD_i^{max}. Consequently, the neural network input will be defined in the interval [0...1]. The output will not give the demand values of each OD matrix component directly, but rather the percentage of variation of each component with respect to the historical OD matrix, i.e. if α_i is the result of the *i*-th forecast OD matrix component, then each component will be:

$$O/D_i^{predicted} = O/D_i^{historical}(1 + \alpha_i)$$
$$\text{where } \alpha_i \in [-1..1]$$

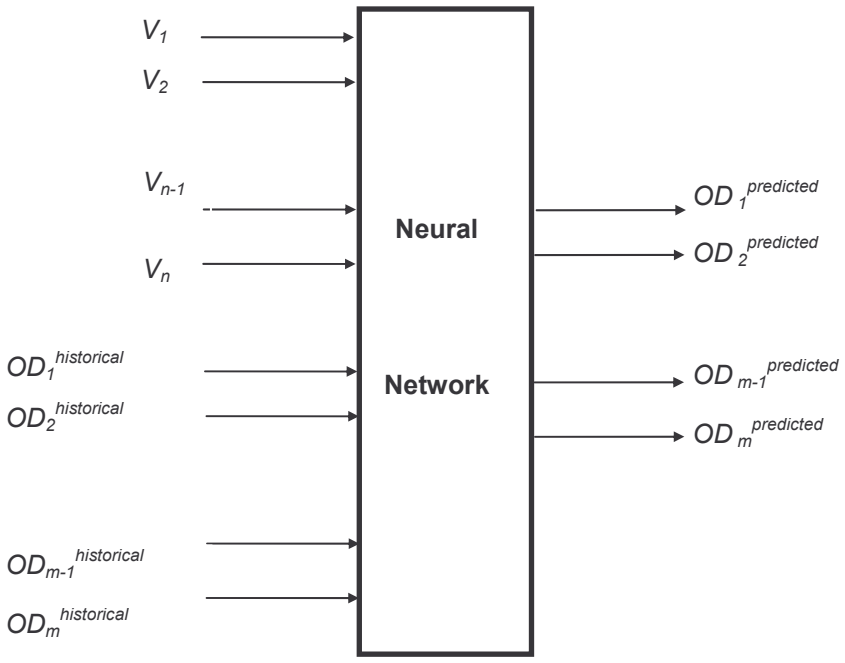


Figure 2.15. A scheme for a predictor module

The expected output of the prediction process for each OD pair is shown in Figure 2.16, as well as the expected prediction for the *i*-th OD pair at times t_1 and t_2 (α_i^1 *and* α_i^2 *respectively).*

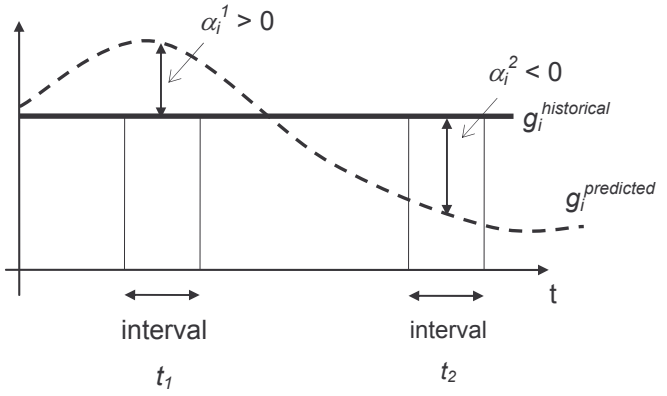


Figure 2.16. Relation of the neural network output to the historical and predicted OD matrices

The training process gauges the neural network, i.e. determines the different weights of the link connections, and this depends on a set of desired input and output pairs. This training or learning process is performed by means of a back propagation procedure using simulated input data. The simulator used to generate these input data was AIMSUN, as described by Barceló and Ferrer (1997b).

The experiment was conducted with the microscopic simulator, which provided, as output, the detector measurements that corresponded to the simulation of traffic flows obtained from an OD matrix. Then, from the historical OD matrix, small perturbations of this matrix expressed as

percentage variations and the detector measurements generated by simulation, the necessary inputs (patterns) for the training module were simulated, as displayed in Figure 2.17.

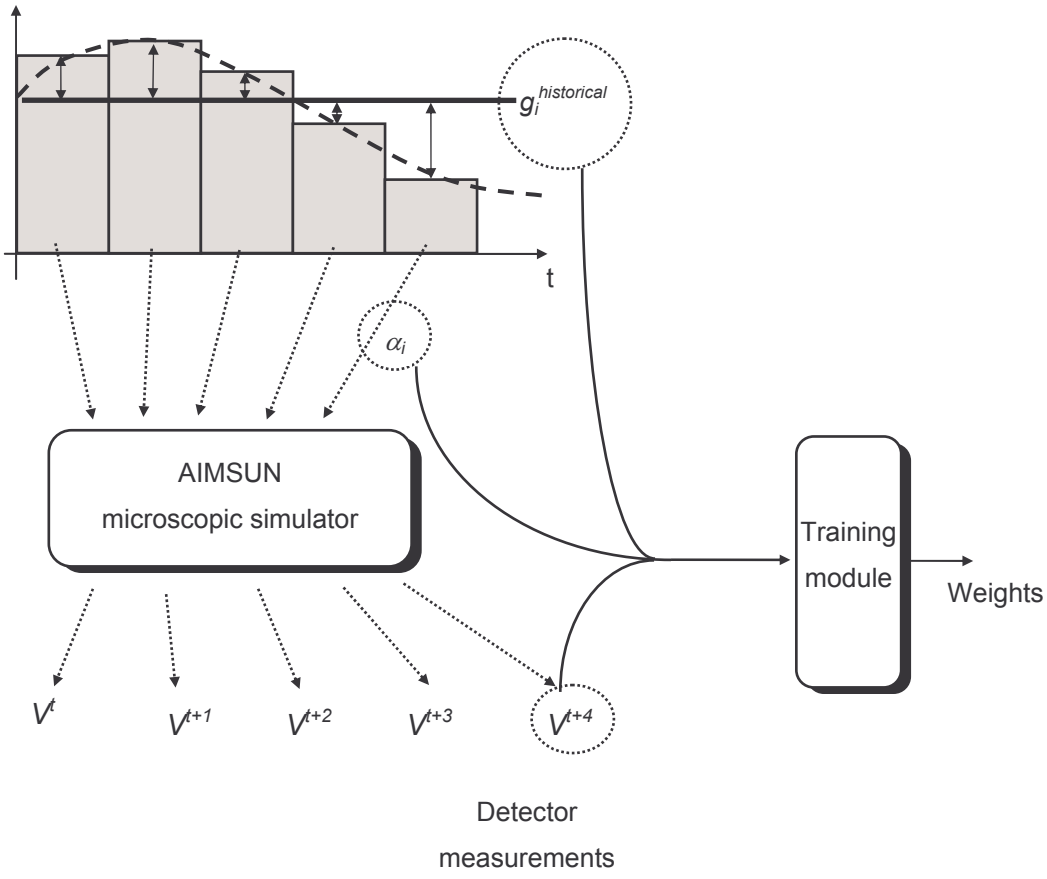


Figure 2.17. Scheme of the experiment

2.2.2.5 TRAINING METHOD: GENERATION OF THE NEURAL NETWORK PATTERNS

The short-term prediction process requires the input of a historical time-sliced OD matrix, as well as the patterns to train the neural network that has to produce the forecasts. Time-sliced OD matrices are currently unavailable and their direct production is difficult and costly, although for several telematic applications the possibility of generating such information in real time has been considered (see, for instance, the report on floating car data in the SOCRATES DRIVE I V1007 project). Our proposal, used in CAPITALS, consists in generating an initial estimate using the information that is available. This initial estimate could later be improved upon and refined with the experience gained during the testing and evaluation of the system. The

generation of the neural network patterns for the training process involved the steps displayed in the logic diagram in Figure 2.18.

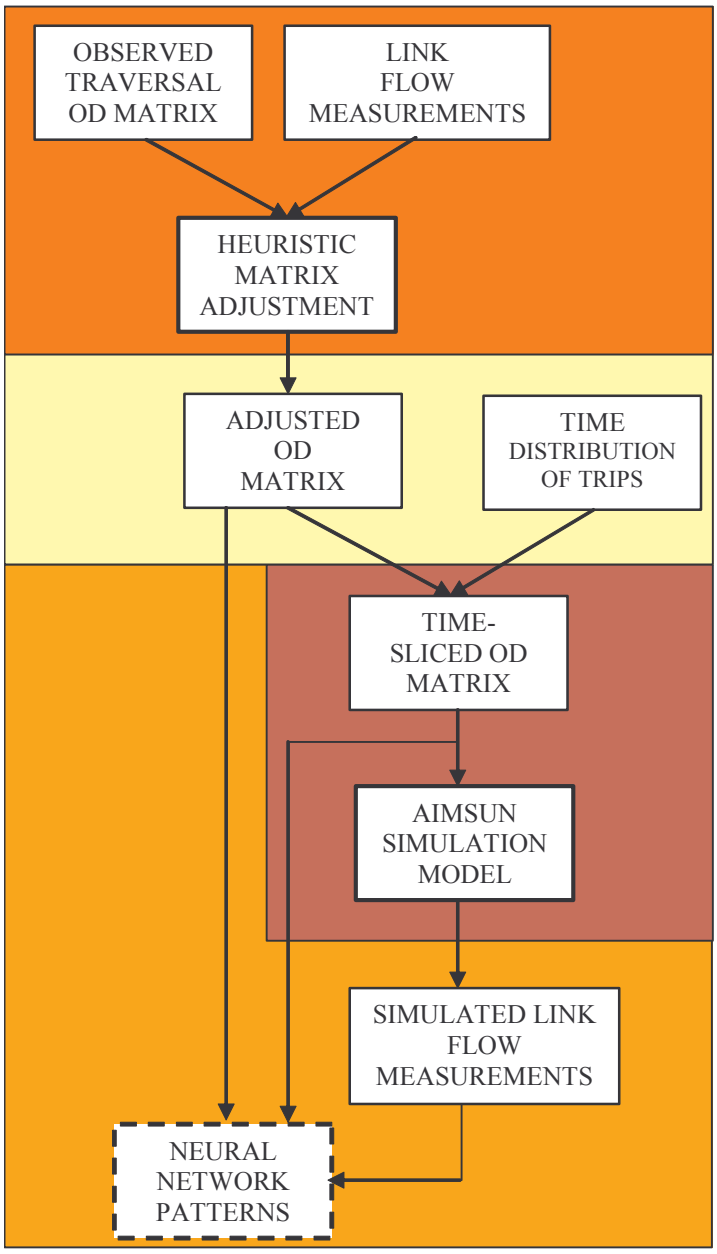


Figure 2.18. Logic diagram of the production of the neural network patterns

1. The traversal matrix for the site and the link flow measurements provided by the data collection were the input to the heuristic matrix adjustment whose output was the adjusted OD matrix. In practice, it was proved that the process is able to adjust trip matrices to flow variations in the time horizon considered; for each time slice, reflecting, in this way, the time variability of the traffic demand is reflected.

We adopted the following bi-level formulation of the matrix adjustment problem as a nonlinear optimisation problem:

$$Min\ F(v(g),\hat{v}) = \frac{1}{2}\sum_{a\in\hat{A}}(v(g)_a - \hat{v}_a)^2$$

$$v(g) = \arg\ \min \sum_{a\in A}\int_0^{v_a} s_a(x)dx$$

$$s.t. \quad \sum_{k\in K_i} h_k = g_i, \quad \forall i \in I$$

$$h_k \geq 0, \quad \forall k \in K_i,\ \forall i \in I$$

$$v_a = \sum_{i\in I}\sum_{k\in K_i}\delta_{ak} h_k, \quad \forall a \in A$$

where $v_a(g)$ is the flow on link a estimated by the lower level traffic assignment problem with the adjusted trip matrix g, h_k is the flow on the k-th path for the i-th OD pair, $\hat{v}_a$ is the flow measured on link a, $\hat{A}$ is the set of with observed flows, I is the set of all origin-destination pairs in the network, and K_i is the set of paths that connect the i-th OD pair. The volume-delay function for link $a \in A$ is $s_a(v_a)$. The algorithm used to solve the problem, which is based on a proposal by Spiess (1990), is heuristic in nature, of the steepest descent type, and does not guarantee that a overall optimum to the problem formulated will be found. The iterative heuristic works as follows:

At iteration m:

- Given a solution g_i^m, an equilibrium assignment was solved, which gave link flows v_a^m and proportions $\left(p_{ia}^m\right)$ that satisfied the relationship

$$v_a^m = \sum_{i\in I} p_{ia}^m g_i^m, \quad \forall a \in A$$

  Note: the target or observed matrix was used in the first iteration (i.e. $g_i^1 = \hat{g}_i, \ \forall i \in I$)

- The gradient of the objective function $F(v(g))$ was computed. The gradient was based on the relative change in the demand, such that

$$g_i^{m+1} = \begin{cases} \hat{g}_i & \text{for } m=0 \\ g_i^m\left(1-\lambda^m\left[\dfrac{\partial F(v(g))}{\partial g_i}\right]_{g_i^m}\right) & \text{for } m=1,2,3,... \end{cases}$$

(A change in the demand is therefore proportional to the demand in the initial matrix, and zeroes are preserved in the process).

- The gradient was approximated by

$$\frac{\partial F(v(g))}{\partial g_i} = \sum_{k \in K_i} p_k \sum_{a \in \hat{A}} \delta_{ak}(v_a - \hat{v}_a), \quad \forall i \in I$$

(where $\hat{A} \subset A$ was the subset of links with flow counts and $p_k = \frac{h_k}{g_i}$).

- The step length was approximated as follows:

$$\lambda^* = \frac{\sum_{a \in \hat{A}} v_a'(\hat{v}_a - v_a)}{\sum_{a \in \hat{A}} v_a'^2}$$

where

$$v_a' = -\sum_{i \in I} g_i \left(\sum_{k \in K_i} p_k \sum_{a \in \hat{A}} \delta_{ak}(v_a - \hat{v}_a) \right) \left(\sum_{k \in K_i} \delta_{ak} p_k \right)$$

To ensure convergence, the step length must satisfy the condition

$$\lambda^* \frac{\partial F(v(g))}{\partial g_i} \leq 1 \qquad \forall i$$

There were two main reasons for selecting this heuristic: the quality of the results it provides, in terms of the regression analysis between the observed and the estimated flows, and the easy implementation using the EMME/2 transport planning package's macro language. This, however, is not the only way of solving the problem; there are alternative heuristics, such as those proposed by Florian and Chen (1995) and Chen (1994), which can also be implemented in EMME/2 by means of the utilities that are available.

2. The adjusted OD matrix was combined with the information collected on the time distribution of the total number of trips on the network, based on observed data, to generate an adjusted time-sliced OD matrix that was consistent with the time variation in the link flows in the time horizon and the time distribution of the total number of trips.

3. The adjusted time-sliced OD matrix was the input to a route-based variant of the AIMSUN microscopic simulation model described by Barceló et al. (1995), in which vehicles follow time-dependent routes from origins to destinations. In this way, a heuristic dynamic assignment was performed. The simulation model emulated the detector measurements

and thus generated a set of link flow measurements similar to those produced by the real detection system.

4. The adjusted time-sliced OD matrix, the simulated link flow measurements and the adjusted OD matrix defined the neural network patterns used in the training process.

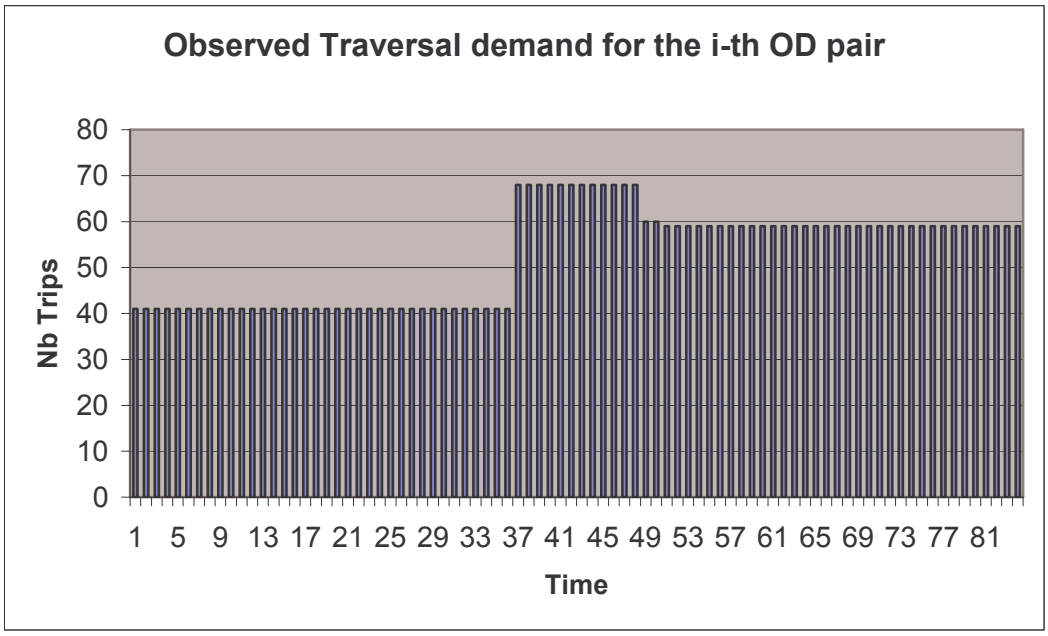


Figure 2.19. Observed traversal demand for the *i*-th OD pair

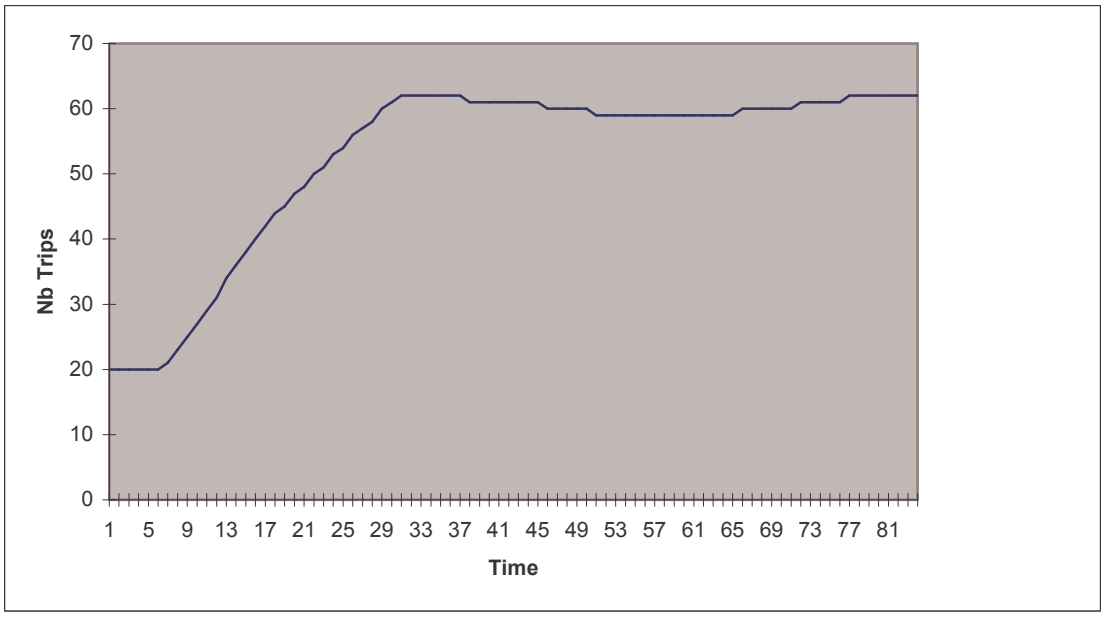


Figure 2.20. Adjusted time-sliced demand for the *i*-th OD pair

Figure 2.19 illustrates the graphic output for the *i*-th entry of the observed traversal OD matrix (historical) corresponding to one origin-destination pair. The values on the x-axis in Figure 2.19 and Figure 2.20 were divided into 84 intervals of 5 minutes each, assuming that the simulated data collection process aggregates the detector measurements every 5 minutes. The y-axis represents the number of trips.

Figure 2.20 shows the same entry for the adjusted time-sliced OD matrix.

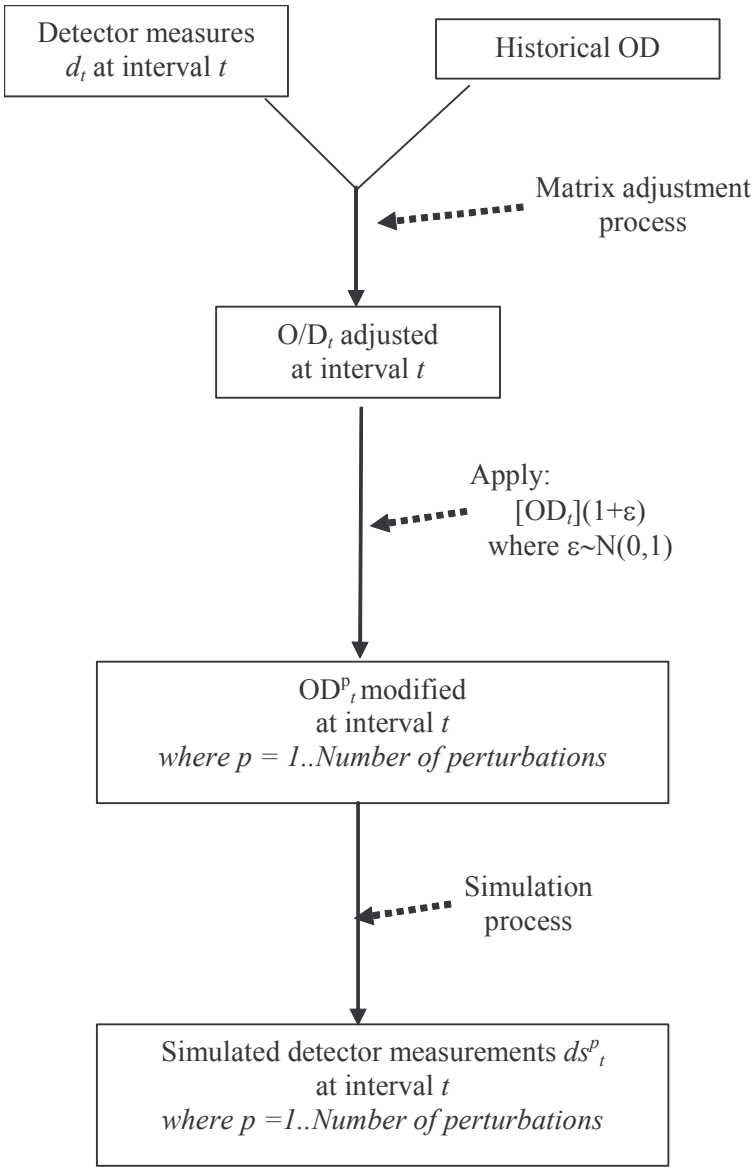


Figure 2.21. Scheme to obtain new patterns

2.2.2.6 RANDOM PERTURBATION OF ADJUSTED OD MATRIX

Training and validating a neural network requires a large number of patterns. To obtain the number of patterns required, we applied a perturbation of the adjusted OD matrices, which were adjusted using the real detector data, by applying a random noise. Figure 2.21 depicts the

scheme used to obtain new patterns from the adjusted OD matrices. Each element *i* of the adjusted OD matrix was multiplied by a factor $(1+\varepsilon_i)$ where ε_i follows a normal distribution ($\varepsilon_i \sim N(0,1)$). Each perturbed OD matrix is the input for the simulation process to obtain simulated detector measurements, and then the new patterns are composed by the perturbed OD matrices and the simulated detector measurements.

2.2.3 ADDRESSING THE PROBLEM OF NEURAL NETWORK SIZE

The dynamic prediction of an OD matrix by means of neural networks has one main drawback: the amount of data required to properly train the neural network. If N is the total number of centroids (origins and/or destinations) in the road network representation, the maximum number of entries in the matrix is N*(N-1), if we take into account the fact that there are no trips from a centroid to itself. Therefore, if the total number of links in the road network with detectors is M, according to the selected topology displayed in Figure 2.14, the total number of entries in the neural network will be N*(N-1)+M, and N*(N-1) will be the number of exits. According to this topology, the total number of parameters that must be estimated during the neural network's training is as follows:

Let l_j be the number of links between layers *j* and *j-1.*

l_1 = (N*N-1)+M)*N*(N-1)

l_2 = (N*(N-1))*(N*(N-1))

Let *l* be the total number of links $l = l_1+l_2$.

Let b_j the number of biases in layer *j.*

b_1 = N*(N-1)

b_2 = N*(N-1)

Let *b* the total number of biases $b = b_1+b_2$.

Then, the total number of parameters to be estimated is ***p = l+b***, which means that, for example,

for N = 25, the total number of parameters to be estimated is p = 721,200,

for N = 50, p = 12,009,900, and

for N = 100, p = 196,039,800.

The large number of parameters to be estimated and the fact that the number of patterns required for proper training must be greater than the number of parameters renders the training

process impossible. **The solution proposed consists in reducing the size of the neural network without undermining its capacity to represent the road network.**

The large number of parameters in neural networks might lead to a significant risk of overfitting; therefore, before applying a model, a careful study of the most appropriate number of parameters is necessary. Nevertheless, in our case, the extraordinary number of parameters involved raised the aforementioned question of the computational impossibility of training the network; therefore, before addressing the risk of overfitting, we had to address the computational issue. Once this issue was addressed, the number of parameters was reduced so drastically that no evidence of a risk of overfitting was found in subsequent computational experiments.

We started at each time period by a preprocessing of the road network, in which we analysed the connectivity of the network and identified the *k* current paths most likely to be used between each OD pair, as suggested by Jayakrishnan et al. (1994), which can be computed using the algorithms proposed by Epstein (1994). The number *k* was fixed empirically (3 or 4 in most cases) and the current travel times estimated by simulation were the cost criterion. Then, if we take into account that volume v_a on link *a* is given by

$$v_a = \sum_{i \in I} \sum_{k \in K_i} h_k \delta_{ak}, \quad \text{where} \quad \delta_{ak} = \begin{cases} 1 \text{ if arc a belongs to path k} \\ 0 \text{ otherwise} \end{cases}$$

and where *I* is the set of all OD pairs, K_i is the set of all paths connecting the *i*-th OD pair and h_k is the flow on the *k*-th path. Then, if we define

$$\bar{I}_a = \{\text{Set of OD pairs using link a in a k shortest path}\}$$
$$\text{where } a \in \hat{A} = \{\text{Subset of links with detectors}\}$$

and we define the auxiliary graph *G=(N,E)*, whose set of nodes *N* and set of links *E* is given by

$$N = \left\{\bar{I}_a : \forall a \in \hat{A}\right\}$$
$$E = \left\{(a,b) : \bar{I}_a \cap \bar{I}_b \neq \varnothing, \forall a,b \in \hat{A}\right\},$$

then the number of different neural networks will be the number of unconnected components of *G*. Consequently, the number *n* of OD pairs to be considered in each neural network (which therefore determines the number of neurons in the input and output layers) is given by

$$n = card\left\{\bar{I}_a \cup \bar{I}_b, \forall a,b : (a,b) \in E\right\}.$$

2.2.3.1 CLUSTER ANALYSIS

In many cases, the partitioning condition can be very strict, which means that, if no significant errors are induced, a certain degree of overlapping should be allowed. In our case, the method

proposed could be replaced by a clustering analysis, because the degree of overlapping could be controlled as a function of the similarity level between clusters.

Cluster analysis requires a preprocessing of the road network, for which the number of vehicles detected at each detector point must be known and grouped according to the origin and destination centroids of each vehicle. In order to do so, a new functionality was added to AIMSUN and, at the end of each simulation, a table was run, in which the rows represented detectors and the columns represented OD pairs. In light of this definition of the rows and columns, the element *(i,j)* denotes the number of vehicles that crossed the *j*-th detector and whose origin-destination pair was *i*-th.

The table used in cluster analysis allows OD pairs to be grouped according to their similarity. The cluster analysis proposed by Dillon (1984) is a process that classifies or partitions groups of items. The process can be divided into the following three steps:

- Step 1. Build a matrix that has a dimension *n* x *p*, where *n* is the number of items and *p* the number of variables.
- Step 2. Obtain a matrix *n* x *n* in which the elements represent a measure of similarity or the distances between the different items.
- Step 3. Obtain the clusters, which can be classified either by exclusive classification, that is, if a partition is being represented, or hierarchical classification.

To carry out this cluster analysis process, the following must be defined:

- the measure of similarity between individuals, and
- the algorithm of clustering using the similarity measure.

The cluster analysis chosen was the Ward method (Dillon, 1984). This process uses error sum of squares (ESS) as a measure of similarity, such that

$$ESS = \sum_{j=1}^{k} \left(\sum_{i=1}^{n_j} X_{ij}^2 - \frac{1}{n_j} \left(\sum_{i=1}^{n_j} X_{ij} \right)^2 \right)$$

where X_{ij} represents the value of the *i*-th item in the *j*-th cluster, *k* is the number of clusters in each algorithm step and n_j represents the total number of items in the *j*-th cluster.

The algorithm, which was developed by Ward (1963), uses a hierarchical process. Hierarchical methods produce “trees”, which are formally referred to as dendograms. Hierarchical methods

fall into two categories: build-up (agglomerative) methods and split-down (divisive) methods. Agglomerative methods generally follow this procedure:

Step 1. Initially, consider each item (customer) to be its own cluster.

Step 2. Join the two items that are closest in terms of a given measure of distance.

Step 3. Join the next two closest objects (individual items or clusters), either by joining two items to form a group or by attaching an item to an existing cluster.

Step 4. Return to step 3 until all the items are clustered.

Agglomerative methods differ in the way in which they join clusters to one another. In Ward's method, clusters are formed on the basis of the change in the error sum of squares (ESS) associated with joining any pair of clusters.

The number of neural networks is determined by the number of clusters and the number of OD pairs to be considered in each neural network is determined by the cardinality of each cluster.

2.3 COMPUTATIONAL RESULTS

The aim of the computational results is to validate the hypothesis of proposing the neural networks as a forecasting mechanism and the methodology used to predict OD matrices. After describing the data preprocessing conducted to generate valid patterns for the training process, two groups of neural network topologies have been computationally tested to determine the one most suited to the OD prediction problem, in terms of R^2 (regression coefficient). These results confirm the initial hypothesis on the use of neural networks as a forecasting mechanism. Finally, the computational results illustrate how the independent OD pairs are determined using a cluster analysis (explained in section 2.2.3.1)

Data from the Madrid site of the EU's CAPITALS project reported in CAPITALS (1998) was used in the computational experiments conducted. The OD matrix for the site area was extracted from the OD matrix for Madrid that had most recently been updated after the 1997 mobility study of Madrid commissioned by the *Consorcio de Transportes de Madrid* (Madrid Transport Consortium). The reference submatrix was obtained as a traversal matrix of the overall matrix using the traversal matrix procedures provided by the EMME/2 transport planning software INRO (1996).

The traversal matrix that is, the local OD matrix for the selected site, was adjusted to a time horizon from 7:00 a.m. till 14:00 p.m. using the traffic counts collected. The employed has been the Spiess (1990) heuristics were employed as the adjustment procedure for a bi-level matrix adjustment model, implemented as a macro of the EMME/2 package.

The result is a historical, adjusted OD matrix sliced into 84 intervals of 5 minutes each.

2.3.1 DATA PREPROCESSING

Patterns are defined by the following:

- Input

  a) *Adjusted OD matrix for the k-*th *time interval*. The number of entries depends on the relationships that are defined in Section 2.2.3. An initial analysis was conducted for each OD pair to determine the most suitable topology and to validate the method. For the sake of completeness, we will present the results for OD pair 5/36. Therefore, in this exercise, there is only one input value.

  b) *Simulated detection for the k-*th *time interval*. The number of entries depends on the number of links with detection on the corresponding subnetwork. There are 23 values for the links with detection on the best routes that connect the OD pair selected.

- Output

  *a) Adjusted time-sliced OD matrix for the (k+1)-*th *time interval.* There are the same number of values as for the input, so one in the case of OD pair 5/36.

In our example, the values available lie in the interval from 7:00 a.m. till 14:00 p.m., which is divided into time intervals of 5 minutes, meaning a total of 84 data sets. Because we want to make the prediction for the next time interval, 83 different patterns are available to us for training the neural network. To avoid the influence of the scale measurements of the various types of input data, the input values in the interval [-1...1] were normalised for all entries. Table 2.1 shows an example of the statistical description of several input data, used as criterion to determine the normalization process.

Statistical description of the input data that corresponds to OD pair 5/36 of historical, adjusted OD matrix

Variable	N	Mean	Median	TrMean	StDev	SEMean
Parell_o	83	52.52	59.00	52.31	10.56	1.16
Variable	Min	Max	Q1	Q3		
Parell_o	41.00	68.00	41.00	59.00		

Statistical description of the input data that corresponds to detector measurements

Variable	N	Mean	Median	TrMean	StDev	SEMean
detect1	83	30.51	0.00	25.28	44.16	4.85
detect2	83	607.7	600.0	586.4	444.6	48.8
detect3	83	249.1	192.0	233.3	182.5	20.0
detect4	83	2.313	0.000	1.120	7.830	0.859
detect5	83	248.8	192.0	234.2	178.1	19.5
detect6	83	165.83	168.00	164.80	54.09	5.94
detect7	83	90.65	72.00	83.04	77.80	8.54
detect8	83	21.11	12.00	19.04	21.09	2.31
detect9	83	201.7	192.0	193.4	154.7	17.0
detect10	83	503.6	528.0	512.5	186.8	20.5
detect11	83	28.34	12.00	23.20	40.39	4.43
detect12	83	3.181	0.000	2.880	5.329	0.585
detect13	83	1205.6	1416.0	1218.9	472.6	51.9

detect14	83	915.6	1008.0	929.6	309.3	33.9
detect15	83	179.3	192.0	178.9	95.7	10.5
detect16	83	47.57	48.00	45.76	34.43	3.78
detect17	83	446.3	492.0	448.8	172.8	19.0
detect18	83	1121.6	1212.0	1132.2	460.9	50.6
detect19	83	584.8	648.0	589.4	203.9	22.4
detect20	83	542.0	564.0	540.6	252.9	27.8
detect21	83	354.4	408.0	360.6	99.4	10.9
detect22	83	2.313	0.000	1.120	7.830	0.859
detect23	83	126.4	96.0	118.9	112.8	12.4

Variable	Min	Max	Q1	Q3
detect1	0.00	192.00	0.00	48.00
detect2	48.0	1608.0	192.0	912.0
detect3	0.0	1020.0	132.0	348.0
detect4	0.000	60.000	0.000	0.000
detect5	0.0	936.0	132.0	360.0
detect6	60.00	288.00	120.00	192.00
detect7	0.00	372.00	36.00	144.00
detect8	0.00	84.00	0.00	36.00
detect9	0.0	576.0	60.0	276.0
detect10	48.0	792.0	396.0	660.0
detect11	0.00	156.00	0.00	36.00
detect12	0.000	12.000	0.000	12.000
detect13	312.0	1860.0	780.0	1584.0
detect14	168.0	1392.0	780.0	1140.0
detect15	0.0	372.0	108.0	240.0
detect16	0.00	180.00	24.00	72.00
detect17	120.0	732.0	300.0	576.0
detect18	216.0	1848.0	732.0	1488.0
detect19	168.0	948.0	468.0	732.0
detect20	120.0	1056.0	324.0	756.0
detect21	120.0	480.0	312.0	420.0
detect22	0.000	60.000	0.000	0.000
detect23	0.0	420.0	36.0	180.0

Table 2.1. Statistical description of input data

2.3.2 NETWORK TOPOLOGY

To illustrate the computational results and the network topology, subsequent sections consider the traffic demand estimation of one OD pair as a subnetwork of the global network to consider all OD pairs. As described in Section 2.2.2.2, the topology used for the neural network was "feed-forward" and the composition of the (5/36) OD pair was the following:

Input Layer

The input layer, which is composed of 24 neurons, performs the transfer of the input value. Therefore, the functions for each neuron are

- Activation function: *identity function*

$$y = x$$

- Output function: *identity function*

$$y = x$$

Hidden Layer

- Activation function:

$$y = \tanh(x)$$

- Output function: *identity function*

$$y = x$$

We tested neural networks that had one or two hidden layers and a variable number of neurons in each layer.

Output Layer

Only one neuron for the case of one OD pair, defined by

- Activation function:

$$y = \tanh(x)$$

- Output function: *identity function*

$$y = x$$

In the results of the experiment, the activation function used is the tanh function. The reasons for this are as follows:

- The use of the tanh as an activation function is due to the output range is defined in the interval -1 to 1 and this interval matches with the output desired in the prediction process (see Section 2.2.2.4)
- The tanh values are centred around 0. There is reason to believe that using tanh will result in faster training (Brown et al., 1993) and (Kalman and Kwasny 1993). For instance, when using a sigmoid, an input value of 0 results in no weight change; however, using a tanh function, an input value of 0, when mapped to -1, results in a weight change. This speeding up of the convergence tends to be even more pronounced the more premapped input values of 0 there are. Suggested theoretical underpinnings for this speeding up have been postulated in the literature. Further discussion of this issue and closely related topics can be found in Gallant (1993) and Stornetta and Huberman (1987).

2.3.3 NEURAL NETWORK TRAINING

The neural network was modelled using the *SNNS* simulator (University of Stuttgart, 1995). The training was conducted with a back propagation algorithm that had the parameters shown below, the values of which were determined empirically.

- The η (learning) parameter determines the step length in the gradient descent direction. The value used in our experiments was η = 0.2.

- The *dmax* parameter determines the tolerance between the output and the input values of the neural network. The value used in our experiments was *dmax* = 0.01

The number of iterations depends on the behaviour of the SSE curve (Sum of Squared Errors). On one hand, the SSE resulting from the training patterns has to be reduced, and on the other, the SSE resulting from testing the validation patterns must also be reduced. The corresponding curves for our training phase are displayed in Figure 2.22, where the y-axis corresponds to the SSE values and the x-axis to the number of iterations.

In a typical neural network training process, an initialisation phase that assigns random weights to the connections takes place first. In our case, we initialised the weights randomly in the interval [-2, 2].

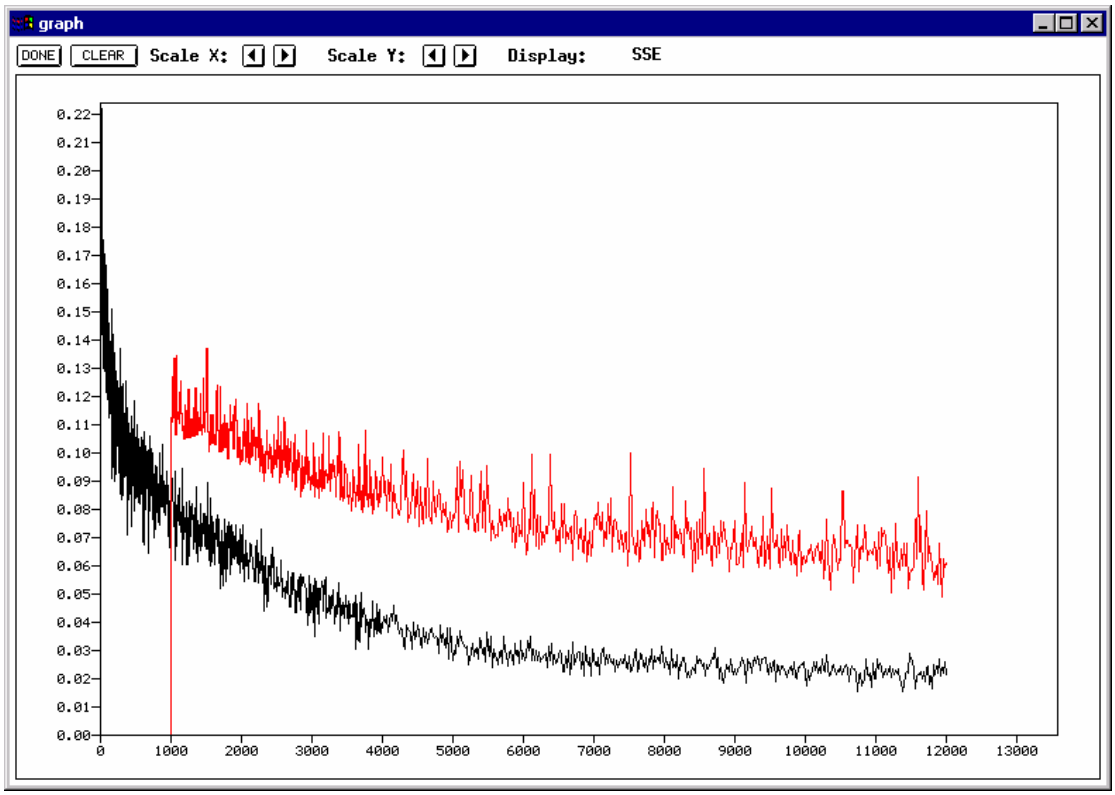


Figure 2.22. Plot of the SSE for the training and validation patterns

2.3.4 RESULTS AND PREDICTION

Two groups of neural network topologies were computationally tested to determine the one most suited to the OD prediction problem. The first group was composed of neural networks that had only one hidden layer and the second group comprised networks that had two hidden layers.

2.3.4.1 NEURAL NETWORKS WITH ONE HIDDEN LAYER

The nomenclature used to describe the neural network topology is *n-m-p*, where the neural network has *n* neurons in the input layer, *m* neurons in the hidden layer and *p* neurons in the output layer.

- Topology 24-4-1

  SSE = 26.71

  R^2 = 0.9983

  SSE_{val} = 101.80

- Topology 24-3-1

  SSE = 48.27

  R2 = 0.9963

  SSEval = 80.14

- Topology 24-2-1

  SSE = 50.74

  R2 = 0.9978

  SSEval = 38.46

By way of example, Figure 2.23 and Figure 2.24 display the results obtained for topology 24-4-1.

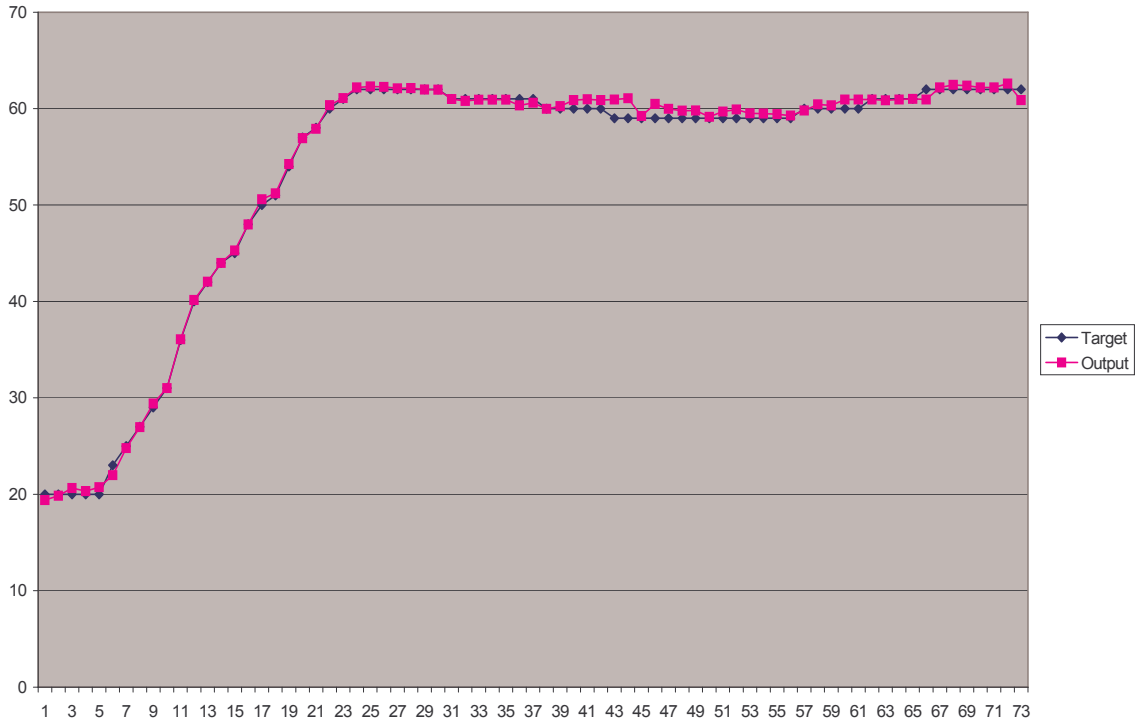


Figure 2.23. Output and target for training pattern 24-4-1

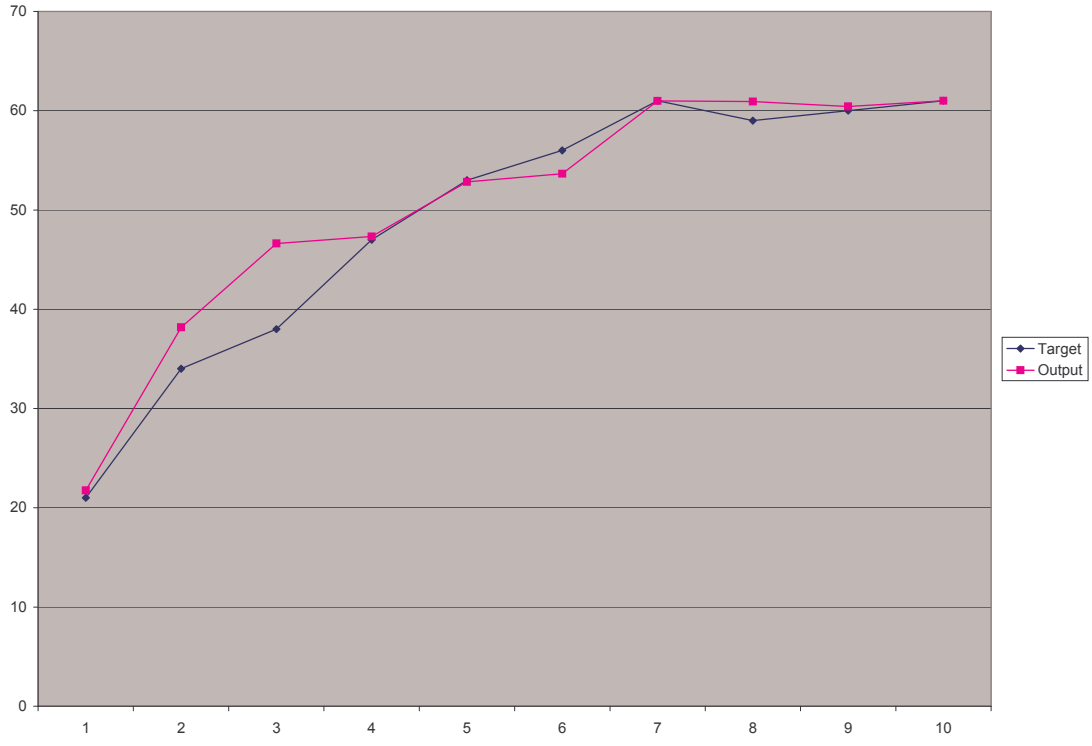


Figure 2.24. Output and target for validation pattern 24-4-1

2.3.4.2 NEURAL NETWORKS WITH TWO HIDDEN LAYERS

The nomenclature used to describe the neural network topology is *n-m-p*, where the neural network has *n* neurons in the input layer, *m* neurons in the hidden layer and *p* neurons in the output layer.

- Topology 24-3-2-1:

  SSE = 6.12

  R2 = 0.9996

  SSEval = 122.07

- Topology 24-2-4-1

  SSE = 12.35

  R2 = 0.9991

  SSEval = 70.94

Figure 2.25 and Figure 2.26 display, in this case, the results obtained for topology 24-3-2-1.

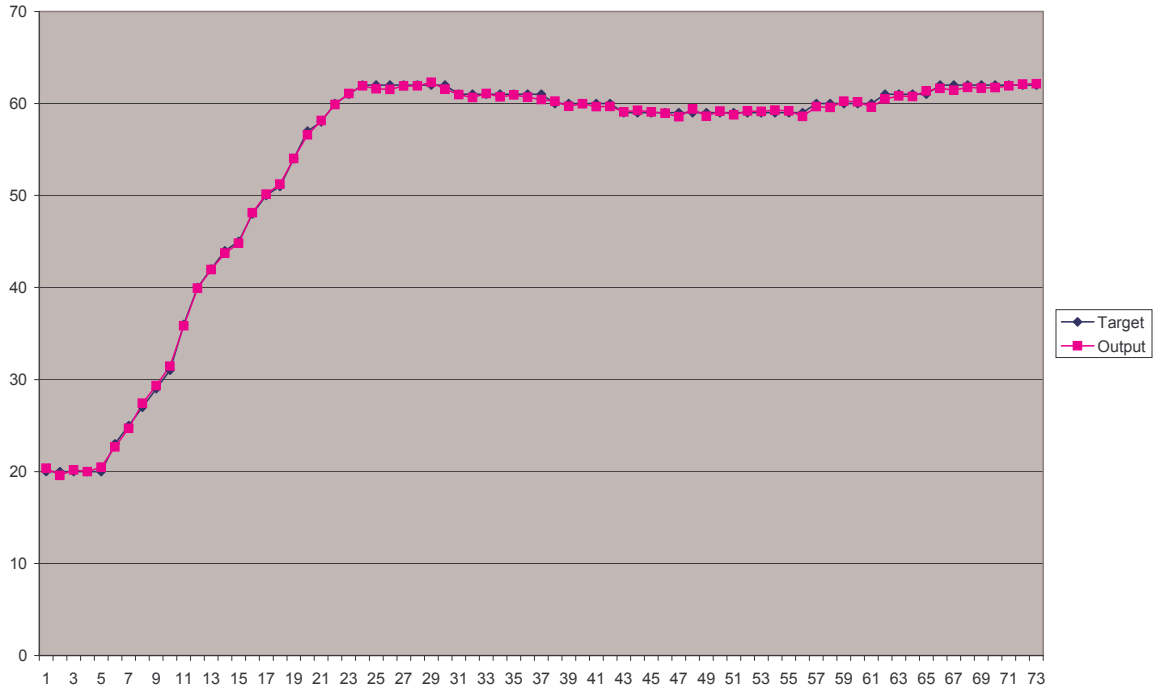


Figure 2.25. Output and target considering the training pattern 24-3-2-1

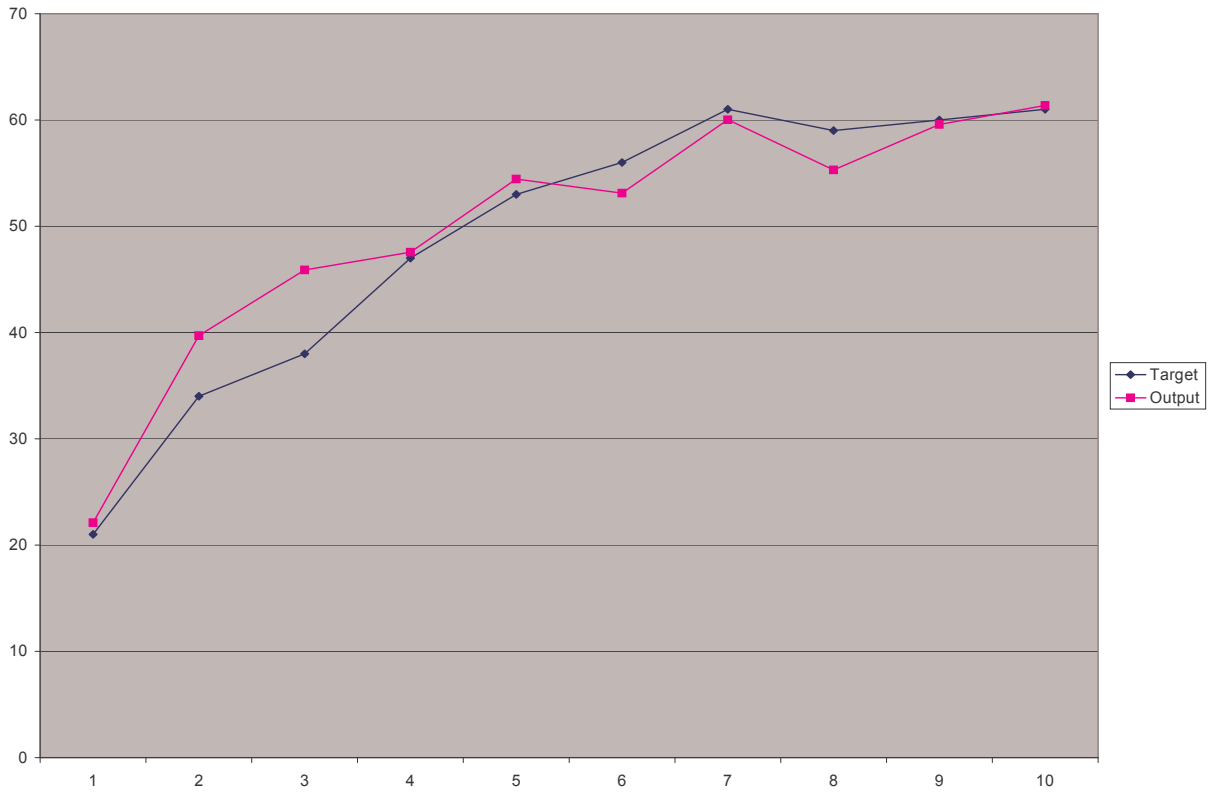


Figure 2.26. Output and target considering the validation pattern 24-3-2-1

The experimental results confirm the validity of both configurations and given the tiny differences between them the decision was to use the simpler one.

2.3.5 NEURAL NETWORK SIZE

The traversal matrix for the CAPITALS site had 98 centroids and thus, in theory, 9506 OD pairs and 377 detectors. This implies a theoretical total of 184,330,846 parameters to be estimated. Having calibrated the AIMSUN model of the site, our analysis revealed that only 1112 OD pairs involved a significant number of trips (at least 5% of the highest entry). If, at most, 10% of overlapping among clusters were accepted, these OD pairs would be clusters according to Ward's method, as implemented in the MINITAB statistical package (1998), which considers the distance between two clusters as the sum of the squared deviations from points to cluster centroids and minimises the within-cluster sum of squares. The distances used were Pearson distances. The method was selected because it tends to produce clusters of a similar number of observations, although it is sensitive to outliers (see details in section 2.2.3.1). The final partition leads to the following 9 clusters:

	Nb of observations	Within cluster sum of squares	Average distance from centroid	Maximum distance from centroid
Cluster1	106	128913.117	33.053	64.682
Cluster2	372	16698.529	5.964	19.210
Cluster3	315	964.750	1.467	5.925
Cluster4	169	38485.558	13.812	40.860
Cluster5	93	47907.066	21.019	49.551
Cluster6	20	82398.261	62.351	95.052
Cluster7	20	4636.483	14.560	23.397
Cluster8	5	5864.195	33.311	47.550
Cluster9	12	25708.829	45.043	68.794

The route-based AIMSUN model enables the paths that are used between origins and destinations to be analysed and the detectors located on the links that make up these paths to be identified. A neural network can be associated with each cluster. Its parameters are determined by the number of OD pairs and the number of detectors on the links of the paths that connect these OD pairs. In this case, the number of parameters to be estimated for the largest neural network, the one associated with cluster number 2, is as follows:

Number of OD pairs = N*(N-1) = 372

Number of detectors = M = 237

Total number of parameters to be estimated = 365,676

For the neural network associated with cluster number 6, N*(N-1) = 20, M = 51 and the number of parameters is 1860, which are more reasonable numbers.