Journal of Data Science **4**(2006), 39-65

Using Hybrid Clustering to Approximate Fastest Paths on Urban Networks

Anjali Awasthi[1], Yves Lechevallier[1], Michel Parent[1]
and Jean-Marie Proth[2]
[1]*INRIA Rocquencourt, Project IMARA and AXIS*
and [2]*INRIA Nancy, Project-SAGEP*

Abstract: Estimating fastest paths on large networks is a crucial problem for dynamic route guidance systems. The present paper proposes a statistical approach for approximating fastest paths on urban networks. The traffic data used for conducting the statistical analysis is generated using a macroscopic traffic simulation software developed by us. The traffic data consists of the input flows, the arc states or the number of cars in the arcs and the paths joining the various origins and the destinations of the network.

To find out the relationship between the input flows, arc states and the fastest paths of the network, we subject the traffic data to hybrid clustering. The hybrid clustering uses two methods namely k-means and Ward's hierarchical agglomerative clustering. The strength of the relationship among the traffic variables was measured using canonical correlation analysis. The results of hybrid clustering are decision rules that provide fastest paths as a function of arc states and input flows. These decision rules are stored in a database for performing predictive route guidance. Whenever a driver arrives at the entry point of the network, the current arc states and input flows are matched against the database parameters. If agreement is found, then the database provides the fastest path to the driver using the corresponding decision rule. In case of disagreement, the database recommends the driver to choose the shortest path as the fastest path in order to reach the destination.

Key words: Canonical correlation analysis, fastest paths, hybrid clustering, traffic simulation.

1. Introduction

The problem of finding fastest path between two points on a network is same as finding the shortest path on a network provided the state of the network or the number of vehicles present in various arcs of the network does not evolves. The solution lies in simply replacing the arc travel times by the arc lengths and

using any shortest path algorithm (based on PERT). Unfortunately, the state of the network does not always remains constant due to the randomly varying input flows and arc states and this leads to continuously changing fastest paths inside the network. This is the reason why a standard shortest path computation algorithm cannot be used for finding fastest paths on urban networks.

A number of methodologies for finding out the shortest paths have been proposed by researchers over recent years. Zhan (1997) presents a set of three shortest path algorithms that run fastest on real road networks. These are the graph growth algorithm implemented with two queues, the Dijkstra algorithm implemented with approximate buckets and the Dijkstra algorithm implemented with double buckets. Buckets are sets arranged in a sorted fashion. The search of an element in a bucket data structure is less time consuming than a non-ordered list. Smith *et al.* (1998) introduced a hierarchical approach for solving shortest paths in a large scale network. Their approach consists of two steps. The first step consists of decomposing the network into several smaller (low level) sub-networks and the second step consists of:

- Finding the sequence of the sub-networks to visit for computing the shortest path.
- Optimization of travel time inside each sub-network .

Data analysis techniques (Cutting *et al.* 1992, Marcotorchino *et al.* 1985, Diday *et al.* 1976) have been applied in the domain of transportation by the following researchers:

- Kohonen Self Organizing Feature Maps (SOFM) (Murtagh 1995, Kohonen 1998, Ambroise *et al. 2000*) have been used with advanced neural networks for freeway travel time prediction. Park and Rillett (1998), Dougherty (1995) use SOFM to transform an incoming signal pattern of multiple dimensions into a one- or two- dimensional discrete map and to perform this transformation adaptively in a topologically ordered fashion. Kisgyorgy and Rillett (2002) use Kohonen SOFM to classify the input vectors into different clusters where the vectors associated with each cluster have similar features.
- Jain *et al.* (1999) present an overview of pattern clustering methods from a statistical pattern recognition perspective.

Our approach uses hybrid clustering followed by canonical correlation analysis for prediction of fastest paths on dynamically changing urban networks. Hybrid clustering is conducted in four steps. The first steps consists of subjecting the traffic data to multiple correspondence analysis. This stage identifies the principal

factors containing maximum amount of information. In the second phase, the dimension space of the data points is reduced. This gives us the most significant data points in the reduced dimension space. The third phase consists of classifying the data points obtained from phase 2 through k-means clustering (Everitt 1974, Jain *et al.* 1999, Meila *et al.* 1998, Diday *et al.* (1978,1983), Tomassone 1988, Roux 1986, Hansen 1997, Klein *et al.* 1991). In the fourth phase, we group the classes obtained from phase 3 through a hierarchical agglomerative clustering method called Ward's method (Ward, 1963). The elements of the classes obtained at the end of phase 4 are subjected to canonical correlation analysis in order to find out the relationship among input flows, arc states and paths, which is the objective of our study.

2. Dynamic Route Guidance Method

The route guidance method developed by us is hierarchical in nature. Our approach for predictive route guidance on large scale urban networks consists of the following steps:

1. Decomposition of the urban networks into sub-networks of reasonable size that would be as much as possible independent from each other. In other words, we are trying to minimize the number of boundary nodes of subnetworks.

2. Finding the sequence of sub-networks to visit in order to reach the destination.

3. Optimization of travel time inside the sub-networks.

Assuming that the sub-networks have already been obtained, we will concentrate mainly on the second and the third steps. The second step involves dynamic programming for finding out the sequence of subnetworks to visit in order to reach a destination. The dynamic programming will provide us a global path that will be used for determining the input and output nodes of the subnetworks to be visited. The third step involves local path computation. This path will be computed using the decision rules obtained from hybrid clustering of traffic data. This paper mainly focuses on local path computation approach.

Note that before subjecting the traffic data to data analysis, we classify the traffic data into ranks. The classification is done in such a way that traffic elements having the same rank will lead us to approximately the same results. For example, two arcs having same arc state ranks will lead to identical travel times for the paths (of which they are a part) between a given origin and the exit node. Likewise, we also assign ranks to input flows and paths.

Note that the global path computation assumes that the characteristics of the state of the system remain constant throughout the journey. This is the only characteristic that differentiates the global approach from the local approach for fastest path computation which assumes that characteristics of the network continuously vary with time. The local approach computes the travel time inside the current sub-network and the next subnetworks depending upon the local traffic conditions prevailing at that instant. Thus, our hierarchical guidance approach involves both global and local path computation. The objective of global path computation is to find out the sequence of the sub-networks to be visited in order to reach the destination and the local path computation is done in order to find out the fastest path between the entry and exit point of each sub-network taking into account the real time state of the system. Figure 1 schematises this approach.

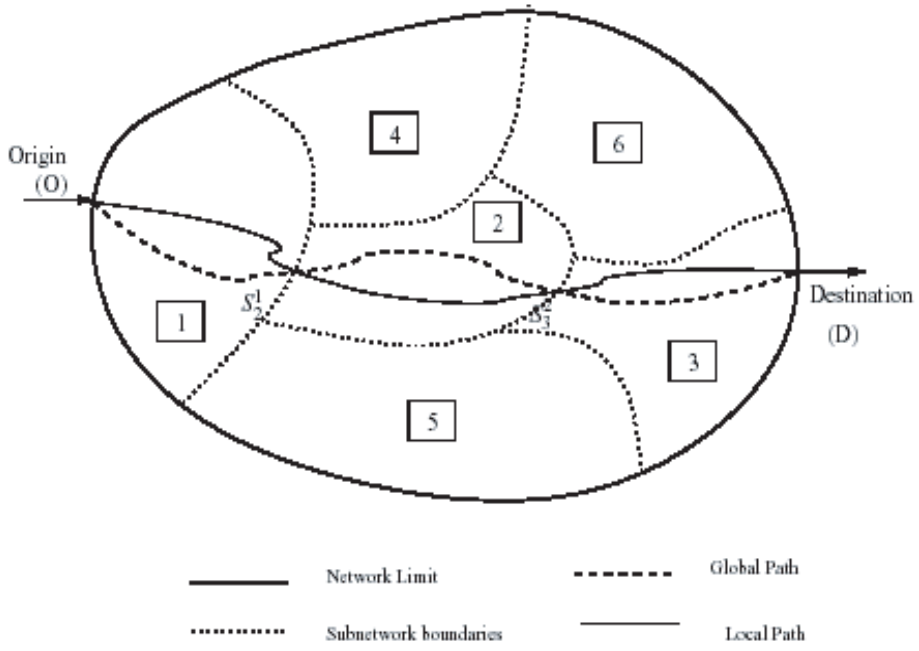

Figure 1: Route guidence approach

In the example presented in Figure 1, the network is decomposed into 6 subnetworks namely, 1,2,3,4,5 and 6. The vehicle enters the network at origin O and wants to go to the destination D. When the vehicle enters inside subnetwork 1

(point O), the global path linking O to D is calculated. This is the path O-S_2^1-S_3^2-D. From the global path we find that the exit node of the sub-network 1 is S_2^1, therefore we will now calculate the local path between origin O and the exit point S_2^1 of the sub-network 1. This path is computed using the rules obtained from hybrid clustering. When the vehicle exits at point S_2^1, then the next step is to find out the local path for the next sub-network to be visited i.e. sub-network 2. The local path for subnetwork 2 is computed between the entry point S_2^1 and exit point S_3^2. After exiting from the subnetwork 2, the vehicle enters the subnetwork 3. The entry point for subnetwork 3 is S_3^2 and the exit point is D. Now the local path between these two points will be calculated. It can be seen that the vehicle leaves the subnetwork 3 at the exit point D which is the final destination of the vehicle. It is evident from this example that the local path computation inside the subnetworks is done over a rolling horizon.

3. Local Path Computation: Methodology

This section is dedicated to the search of the best local path inside the sub-networks. When a vehicle enters a subnetwork, we are interested in not only knowing the path that will connect the point of entry to the point of the exit (or at the end of the path if it appears in the subnetwork), but also the optimisation of travel time on this path taking into account the real time state of the system.

When a driver enters into a sub-network, we know:

- The input flows at the entry points of the subnetwork.
- The number of vehicles inside the arcs of the subnetwork.

The input flows being stepwise constant, we can assume that they are going to remain stable at least for few minutes and will sufficiently allow a vehicle to traverse this sub-network. These flows modify the initial state of the system and therefore influence the decisions of the drivers who want to reach the exits of the sub-networks as fast as possible. Hybrid clustering is used to compute the local paths which will be used by the drivers in order to reach the exits in minimum time. The traffic data for hybrid clustering is generated using a traffic flow simulation software developed by us. The inputs to the software are the input flows and the arc states and the outputs from the software are the different paths connecting the origins and the destinations of the network. The traffic data before conducting the hybrid clustering is prepared in the following manner:

- We generate at random a large number of input flows and arc states within a fixed range. Two types of preparatory activities are done before starting the simulation:

- Finding the domain or the range in which each of the variables will take an important percentage (95% for example) of the observed values.
- Finding the correlation between the variables arc system state and the input flow.

For example, it is less probable that an arc will be saturated if its predecessors arcs are empty. Therefore, there exists certainly a strong correlation between the number of cars in arc l_1 and the number of cars in arc l_2. To cope up with this, we will perform random generation of system states.

The preliminary study has not been conducted in the paper. We have simply assumed that the arc state and the input flow variables are non-correlated and that they take their values between 0 and the maximum value generated from a uniform probability function.

- For each of these pairs (input flow, system state), we compute the paths and their corresponding travel times among all the origin-destination pairs of the network. This calculation is done using simulation.

We use the results obtained from the numerous simulations for the construction of a memory or database that will inform the user the best way to traverse the sub-network in which he wants to enter. This requires the knowledge of the pair (arc system state, input flows) to extract the corresponding best path from the memory. The best paths are stored in the memory in the form of rules. These rules provide the fastest path as a function of the input flow and the system state of the arcs.

Remark: For developing a hierarchical route guidance system, we also developed algorithms for decomposing large scale urban networks into small sub-networks such that they are as far as possible independent from each other (Awasthi *et al.*, 2004). In other words, we tried to obtain sub-networks in a way such that the total number of input and output nodes or the boundary nodes for all the sub-networks is minimized. This also leads to the minimisation of the volume of the traffic data for each subnetwork.

4. Classification of Traffic Data

The classification of traffic data consists of transforming the input data (input flow, arc states) and the output data (paths) obtained from the simulation software into classes. The inputs of the simulation software i.e. the input flow and the number of vehicles in each arc of the system are divided into three classes. We assume that the values of input flow or the number of vehicles in arcs have same impact on path travel times as their ranks, therefore the division of traffic

data into classes or ranks is justified. Each traffic variable is provided a class out of the three classes as follows:-

- It belongs to class 1 or has rank 1 if its value lies between 0 and 60% of the maximum value.
- It belongs to class 2 or has rank 2 if its value lies between 60% and 90% of the maximum value.
- It belongs to class 3 or has rank 3 if its value lies between 90% and the maximum value.

For instance, if the input flow at an entry point lies between 90% and the maximum flow permitted at that entry point, then we will say that the input flow belongs to class 3 or has rank 3. Likewise, if an arc state has rank 2, this means that the number of vehicles present in the arc at that instant lie between 60% and 90% of the maximum capacity of the arc. Note that the classification of traffic variables is done into 3 classes only because these ranges allow us to sufficiently view the impact of input flows and arc states on travel times. However, other users can choose different classes or ranges depending upon their utility.

The classification or ranking of the paths is done on the basis of travel times. They are classified in three ranks as follows:

- The fastest path is assigned rank 1.
- The second fastest path is assigned rank 2.
- The third fastest path is assigned rank 3.

Each simulation result i.e. input flow, arc state and path is thus transformed into ranked data in the form of 1,2 or 3 for all the origin nodes, arcs and paths of the network. If n_1 is the total number of origin nodes, m is the total number of arcs and p_1 is the total number of paths of the network, then the number of input parameters are $n_1 + m + p_1$. Since each parameter can take 3 ranks, therefore the total number of traffic parameter combinations available for hybrid clustering is equal to $3^{n_1+m+p_1}$. Note that all the traffic parameters do not have the same importance and the objective of the method presented in the next paragraph is only to extract the parameters that are most significant from the point of view of decision making. We will therefore observe the importance of these parameters by keeping a limit on the size of the network i.e. by reducing m.

Table 1 presents the various traffic parameters (input flow, arc state, path) obtained from the simulation. It can be seen in Table 1 that the arc states, input flows and paths are divided into 3 classes i.e. rank 1, 2 and 3 and that one simulation experiment comprises of only one X allotted to the triplet of

columns assigned for the different traffic parameters. This means that during one simulation run, the impact of one class of input flows (at different entry points) and one class of arc states (for different arcs) is observed on the paths (between different origin-destination pairs) of the network.

Table 1: Simulation results

No.	Input 1			Input 2			–	$Arc(i,j)$			$Arc(k,l)$			–	$Path\ C1$			$Path\ C2$			–
1	x			x			–	x			x			–	x					x	–
2			x		x		–	x				x		–			x		x		–
3		x				x	–		x				x	–		x			x		–
7			x			x	–		x		x			–	x					x	–
–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

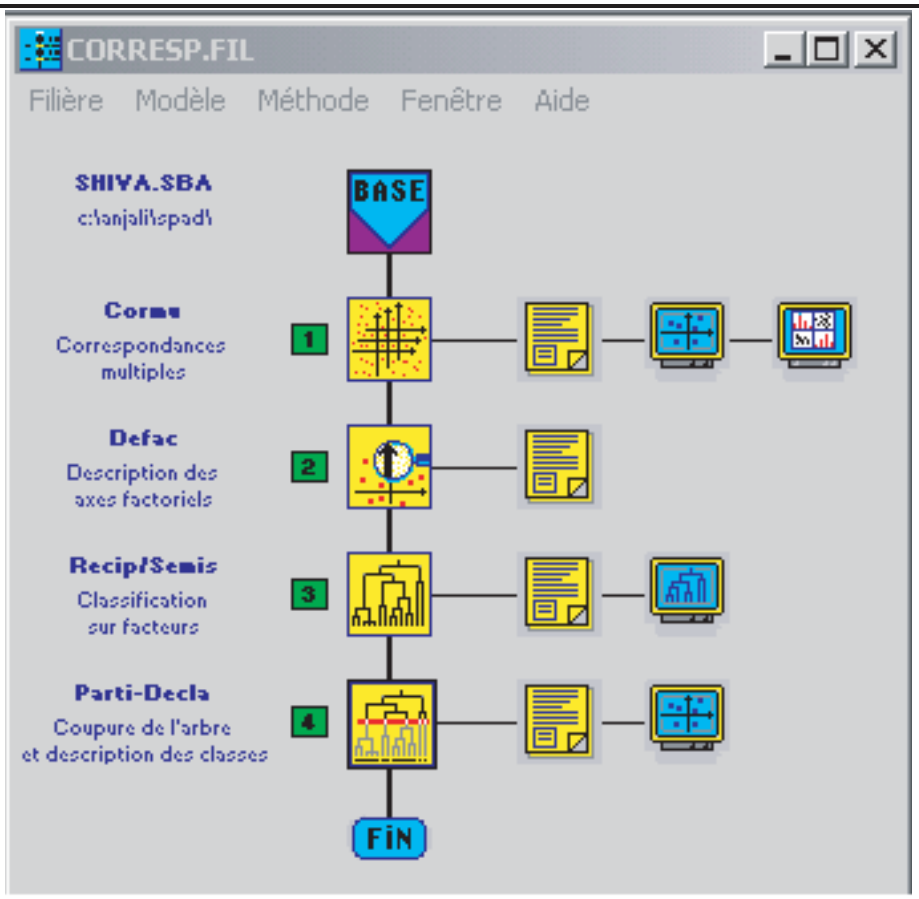

Figure 2: Four methods in SPAD

5. Hybrid Clustering

Hybrid Clustering is performed on the ranked data obtained from traffic simulation. The statistical software used for conducting the hybrid clustering is SPAD[1]. The four steps of the hybrid clustering (described in section 1) were performed by four modules of SPAD namely CORMU, DEFAC, SEMIS and PARTI/DECLA. Figure 2 depicts the four modules of SPAD.

5.1 Multiple correspondence analysis

The first step of hybrid clustering consists of multiple correspondence analysis (MCA). This step was achieved through CORMU module of SPAD. Before commencing MCA, the data's are filtered in the following manner: Let us consider Table 1. We count the number of crosses (X) in each column and divide it by the total number of simulations or the total number of rows of the table in order to compute a ratio. This ratio gives us proportion in which the variable of the column appears in the total number of simulations. If this ratio is inferior than a threshold value r, then we reject that column from further study because it is considered to be contributing insignificantly to the study of the problem. Note that the value of r is provided by the user and we will see later in the numerical example section that $r = 0.02$.

The filtered data is subjected to multiple correspondence analysis. The results are the projection of data points (each point being obtained from the simulation) on the factor space. The MCA graphs represent the projections, i.e. two points that are close on this graph are not necessarily close on the complete dimensions (even for a small network like that presented in the numerical example section, the number of dimensions needed to explain the total variance are 59). It is therefore important to correctly interpret the coordinates of these projected data points in the space defined by the factors.

Treatment 1: *The multiple correspondence analysis allows us to detect the relationship between the input flow ranks, arc state ranks and the paths of rank 1 or the fastest paths. Besides, it also allows us to identify the principal factors containing the maximum information i.e. which contribute maximum to the total inertia.*

5.2 Reduction of variable space

Let d be the number of factors used to define the traffic variables. This

[1]CISIA (1997). *SPAD Reference Manuals.* Centre International de Statistique et d'Informatique Appliques, France 1997.

means that each traffic data i.e. input flow, arc state and fastest path can be represented by a point in d dimension space. We then select the factors that contain the maximum amount of information. Let f denote the number of these factors and $f < d$. In the numerical example presented in section 8, $d = 59$ and $f = 10$.

The DEFAC module of SPAD allows us to project the group of points (the simulation data) initially situated in a space of dimension d into a reduced space of dimension f.

Treatment 2: *The module DEFAC allows us to represent the data points obtained from MCA in a reduced dimension space. The data points are represented in the new coordinates defined by the factors that contain the maximum amount of information(i.e. which contribute maximum to the total inertia).*

5.3 Classification of points in the new space

The third method consists of applying k-means & Ward's clustering method on the data points obtained in the reduced factor space. The k-means clustering[2]is performed by the SEMIS module of SPAD. Let N be the total number of points to be classified in the dimension space f. The k-means clustering method can be explained as follows:-

k-means clustering

Step 1: Initialization

Let $N = \{1, 2, 3...n\}$. We select $k^o < n$ elements from N at random. These elements represent the initial centres and are denoted by U_z^o. The set containing the centres $U_1^o, U_2^o, ...$ etc. is represented by S^o. In other words $S^o = \{U_1^o, U_2^o, U_3^o, ..U_{k^o}^o\}$ where $S^o \subset N$.

Step 2: Assignment

For $i = 1, 2, ..n$ we assign an element i to cluster C_z^o if

$$d(i, U_z^o) = Min_{\{j=1,2,..k^o\}} d(i, U_j^o)$$

Step 3: Computation of S^1

For each C_z^o, we compute the mean value of the elements of C_z^o. Let U_z^1 be this value and $S^1 = \{U_1^1, ...U_{k^1}^1\}$. Note that we may have $k^1 < k^o$.

Step 4: Test

If $S^1 \equiv S^o$ stop, otherwise set $k^o = k^1, S^o = S^1$ and go to step 2.

[2]Legendre, P. (2000), Program k-means. User's guide, Departement de sciences biologiques, Universite de Montreal. Available from http://www.fas.umontreal.ca/biol/legendre/.

It has been proved that this algorithm converges. In practice, this method is conducted many times in order to obtain the stable classes i.e. classes which are formed always irrespective of the initial choice of random centers at each experiment.

Treatment 3: *The k-means clustering method is run multiple times using randomly chosen initial centers in order to obtain the stable classes. We know that the data points contained in a class are very close and therefore a class can be replaced merely by a unique point.*

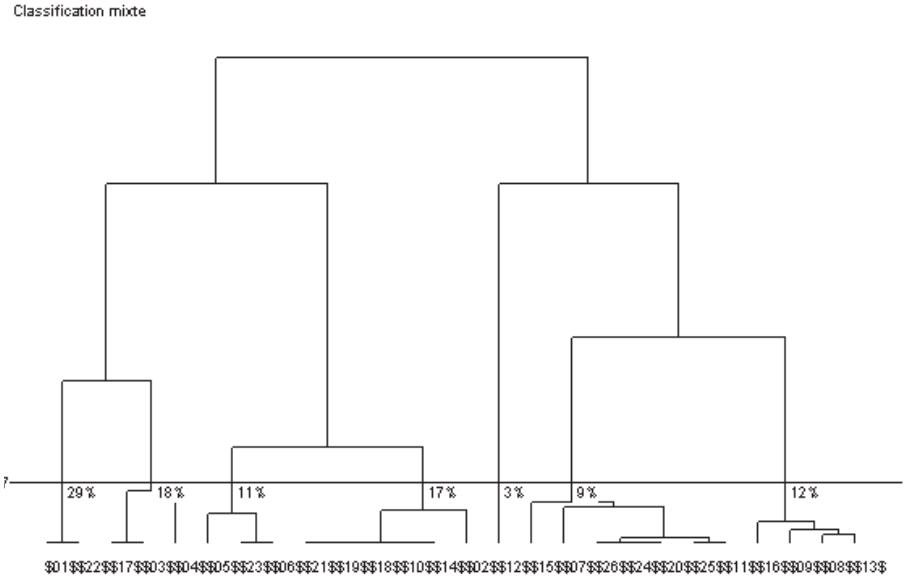

Figure 3: Ward's hierarchical tree/dendogram

5.4 Regroupment of classes closest to each other

This step performs hierarchical agglomerative clustering using Ward's method. An agglomerative hierarchical clustering procedure produces a series of partitions of the data, P_n, P_{n-1}, ... , P_1. The first P_n consists of n single object 'clusters', the last P_1, consists of single group containing all n cases. At each particular stage the method joins together two clusters which are closest to each other (most similar). In the Ward's method, the union of those two clusters occur at each step whose fusion results in minimum increase in 'information loss'. Information loss is defined by Ward in terms of an error sum-of-squares criterion (Subsection 5.4.1).

PARTI/DECLA module of SPAD is used to perform this step (see Figure 2). The classes which are close to each other are consecutively grouped together until

a predefined number of classes (i.e. $\geq$ threshold value provided by the user) have been obtained. For example, In Figure 3, 7 classes have been obtained from 10 initial classes using Ward's method. It can be seen that these 7 classes explain 29%, 18%, 11%, 17%, 3%, 9% and 12% of the total inertia.

Note that the higher is the threshold value, the greater are the number of classes obtained at the end of Ward's clustering. However, higher number of classes may also yield results that are less precise i.e. there is less possibility that the elements of one class are present in other classes. Ward's method is explained as follows:

Ward's method

Ward (1963) considered that any stage of regroupment of elements, the loss of information which results from the grouping of elements into clusters or classes can be measured by the total sum of squared deviations of every point from the mean of the cluster to which it belongs. At each step, union of those two possible clusters which cause minimum increase in the Error Sum of Squares (E.S.S) is considered. If x_{ij} denotes the j^{th} element of cluster j, and the total number of clusters is k, the number of elements in each cluster being n_j , then the E.S.S for cluster c represented by v_c is given by:

$$E.S.S.(v_c) = \sum_{j=1}^{k} \sum_{i=1}^{n_j} (x_{ij} - \overline{x_j})^2$$

where

$$x_j = (1/n_j) \sum_{i=1}^{n_j} x_{ij}$$

Initially, each cluster consists of a single element, therefore E.S.S = 0 for all clusters.

Treatment 4: *The data points clustered together at the fourth stage lead to classes (also called groups or clusters) each of which contains elements that we considered at the start of the analysis i.e. ranked input flows, arc system states and paths.*

It is likely that in certain classes, we will observe paths of rank 1 along with different arc states and input flows. This implies that the ranks of these input flows and arc states govern the fastest path. This approach will be adopted for interpreting the results. Later on, we will also find the degree of correlation between the input flow ranks, arc state ranks and the path ranks in each class.

5.5 Canonical correlation analysis

Canonical correlation analysis is used to find out the correlation between the input flow ranks, arc state ranks and the path ranks of each class. A threshold value is set for the correlation coefficient in order to identify the strength of relationship among the traffic elements.

Treatment 5: *The canonical correlation analysis allows us to find out the input flow and arc state ranks which are correlated with the paths of rank 1 in each class. The strength of co-relationship between the input flow ranks, arc system state ranks with paths of rank 1 is determined by comparison against a threshold value.*

6. Results

The results of hybrid clustering are presented in the form of decision rules that will permit the driver to decide the fastest path in real time in order to reach the destination. The decision rules can be of the following type:

IF *($r(i_1, j_1) = 1$)* **AND** *($r(i_2, j_2) = 2$)* **AND** *($r(e_1) = 3$)* **AND** *($r(e_2) = 1$)* **THEN** *the fastest path is (a1, a2, ..as).*

In this rule:

- $r(i_1, j_1)$ represents the state of the arc joining node i_1 to node j_1. The condition $r(i_1, j_1) = 1$ indicates that the rank of the state of arc (i_1, j_1) is 1 or in other words the number of vehicles in the arc (i_1, j_1) lies between 0% and 60% of its maximum capacity. The reverse of this condition would be $r(i_1, j_1) \neq 1$. Let us assume that in a rule we observe $r(i_1, j_1) \neq 1$. The interpretation of this condition would be that the system state of the arc (i_1, j_1) does not lies between 0% - 60% of its maximum capacity. This means that either it has rank 2 (60% - 90% of the maximum capacity) or rank 3 (90% - 100% of the maximum capacity), but never rank 1.
- The condition $(r(e_1) = 1)$ indicates that the input flow at the entry node e_1 has rank 1 or it lies between 0-60% of the maximum flow. As illustrated before, we may across a condition of the type $(r(e_1) \neq 1)$ which would mean than input flow never possesses rank 1 though other ranks i.e. 2 or 3 are possible.

Therefore, decisions for predicting the fastest paths are done using rules. Whenever a driver enters the network, the current ranks of the input flows and the arc states are searched inside the database to find out the decision rule which will provide the fastest path under these traffic conditions. Following cases are possible:

1. There is no rule in the database for certain pairs of input flow ranks and arc state ranks. This signifies that we have not succeeded in identifying the fastest path under these conditions. This situation may arise due to two cases. The first case occurs when all the initial arc states and the desired input flows have not been simulated i.e. they have not been considered while performing the simulations (result of a badly conducted study). In the second case, insufficient number of simulations have been conducted on the initial data. In both of these cases, no rule should be provided and the shortest path is predicted as the fastest path.

2. There are many rules in the database for fastest path between a given origin-destination pair. This would mean that there exist various combinations of arc state and input flow ranks that lead to the selection of the fastest path.

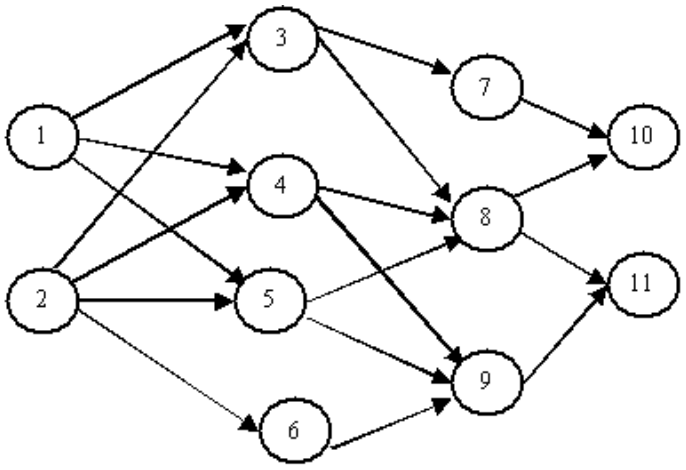

Figure 4: A multiple origin destination network

7. Important Remarks

- It is clear from the previous sections that the quality of the results depends on the traffic data (i.e. input flow ranks, arc state ranks and path ranks) and the number of experiments chosen for simulation. If the number of simulation experiments conducted do not correspond to reality, then the rules obtained are not efficient and in fact they can be counter-productive also.

- It is absolutely important that the route guidance system should be aware of the input flows and arc states on the network on a real time basis. However, this information can be limited to only those that appear in the rules (which are in most of the cases less than the information needed for all arc states and input flows for all the origin nodes). Finally, when a vehicle enters a sub-network, it knows its origin and destination (obtained from global path). We therefore need only a fixed number of rules corresponding to the required origin and destination for finding the fastest path inside the sub-network (local path).

8. Numerical Example

8.1 Input data

Let us consider the network depicted in Figure 4.

The traffic simulation on this network was conducted under varying input flows and arc states. The input flows and arc states were generated at random between 0 and their maximum value. Ranks were allotted to the input variables i.e. input flows and arc states as follows:

- The variable has rank 1 if its value lies between 0 and 60% of the maximum value.
- The variable has rank 2 if its value lies between 60% and 90% of the maximum value.
- The variable has rank 3 if its value lies between 90% and 100% of the maximum value.

The classification of various paths of the network was done as following :

- The fastest path is assigned rank 1.
- The second fastest path is assigned rank 2.
- The third fastest path is assigned rank 3.

The maximum permitted value of input flows at the entry points of the network was set equal to 70. These flows remained constant during a time interval of 200 time units. The input flows arriving at the two entry points of the network varied between 0 and the maximum permitted value i.e. 70. Different ids were assigned to the input flows depending upon their value. For example, for the origin node 1:

- The input flow was allotted an id I1 if its rank was 1.
- The input flow was allotted an id I2 if its rank was 2.
- The input flow was allotted an id I3 if its rank was 3.

Likewise, I4, I5 and I6 represent the ids of the ranked input flows for the origin node 2.

Table 2: Network parameters at time t_o

Arc Id (i, j)	Maximum Capacity	Arc identities as per ranks 1, 2 & 3
(1, 3)	175	S1, S2, S3
(1, 4)	210	S4, S5, S6
(1, 5)	240	S7, S8, S9
(2, 3)	200	S10, S11, S12
(2, 4)	160	S13, S14, S15
(2, 5)	150	S16, S17, S18
(2, 6)	180	S19, S20, S21
(3, 7)	260	S22, S23, S24
(3, 8)	180	S25, S26, S27
(4, 8)	165	S28, S29, S30
(4, 9)	132	S31, S32, S33
(5, 8)	192	S34, S35, S36
(5, 9)	180	S37, S38, S39
(6, 9)	140	S40, S41, S42
(7, 10)	119	S43, S44, S45
(8, 10)	120	S46, S47, S48
(8, 11)	130	S49, S50, S51
(9, 11)	132	S52, S53, S54

Table 2 presents the characteristics of the network. The first column represents the arc ids. An arc id (i, j) indicates i is the tail and j is the head of the arc. The second column represents the traffic accommodation capacity of the arc or in other words the maximum number of vehicles that can be present inside the arc (i, j). Finally, the last column represents the identities assigned to various arcs of the network on the basis of their ranks. For instance,

- S1 represents the id of arc (1,3) when its state lies between 0 and 60% of its maximum capacity 175, i.e. between 0 and 105 vehicles.

- S2 represents the id of arc (1,3) when its state lies between 60% and 90% of its maximum capacity 175, i.e. between 105 and 157.5 vehicles.

- S3 represents the id of arc (1,3) when its state lies between 90% and 100% of its maximum capacity 175, i.e. between 157.5 and 175 vehicles.

Table 3: Path details

O-D Pair	Paths	Length	Path identities for ranks 1, 2 & 3
1-10	1-3-7-10	27	P1, P2, P3
	1-3-8-10	25	P4, P5, P6
	1-4-8-10	26	P7, P8, P9
	1-5-8-10	28	P10, P11, P12
1-11	1-3-8-11	27	P13, P14, P15
	1-4-8-11	28	P16, P17, P18
	1-4-9-11	29	P19, P20, P21
	1-5-8-11	30	P22, P23, P24
	1-5-9-11	28	P25, P26, P27
2-10	2-3-7-10	30	P28, P29, P30
	2-3-8-10	28	P31, P32, P33
	2-4-8-10	27	P34, P35, P36
	2-5-8-10	26	P37, P38, P39
2-11	2-3-8-11	30	P40, P41, P42
	2-4-8-11	29	P43, P44, P45
	2-4-9-11	30	P46, P47, P48
	2-5-8-11	28	P49, P50, P51
	2-5-9-11	26	P52, P53, P54
	2-6-9-11	27	P55, P56, P57

Table 3 presents the rank ids for different paths of the network. The first column represents the origin-destination pair. For each of these pairs:

- The column 2 represents the various paths that join this origin-destination pair.

- The column 3 represents the lengths of these paths.

- The column 4 represents the ids of the paths. For example, path 1-3-7-10 has an id P1 if it is the fastest path, an id P2 if it is the second fastest path and an id P3 if it is the third shortest path.

8.2 Application of the method

The total number of simulations carried out were 10,000 where the number of iterations per simulation were 200. During each simulation experiment, we generated input flows and arc system states at random (as explained in section 4). SPAD statistical software[3] was used to perform hybrid clustering on the traffic data obtained from simulation. The four methods of hybrid clustering were performed by the CORMU, DEFAC, SEMIS and PARTI/DECLA modules of SPAD.

Multiple correspondence analysis (MCA) was done on the traffic data using CORMU. Before subjecting to MCA all the traffic variables were classified as nominal active and variables that contributed less than 2% to the simulation results were discarded. Figure 5 presents the results obtained from MCA. In the Figure 5, the paths, arc states and input flow ranks are represented by their ids.

Figure 5: Multiple correspondence analysis

It can be seen in Figure 5 that the paths and arcs are well-separated with most of the arcs and input flows located around the centre and the paths spread far away. However, few arcs are situated close to the paths and can be seen around the centre. For instance P39 and S39. The path id P39 (path 2-5-8-10 with rank 3) is close to the arc id S39 (arc (5,9) with rank 3). The MCA findings

[3]CISIA (1997). *SPAD Reference Manuals*. Centre International de Statistique et d'Informatique Appliques, France 1997.

indicate that a total of 59 factors or axes are required for explaining 100% of the total inertia.

The second method of hybrid clustering namely reduction of factors was done using DEFAC. The outcome was 10 principal factors that contribute maximum to the total inertia. The variances explained by these factors are 7.55%,7.35%, 6.33%,3.59%, 2.59%, 2.01%, 2.0%, 1.86%, 1.85% and 1.84%. (= 36.96% of the total inertia). At this stage, the number of factors reduce from 59 to 10.

The third method namely k-means clustering was performed using SEMIS. For generating the clusters in the traffic data, 10 centres were randomly chosen and 20 iterations of clustering were done. In this way 3 clustering experiments were done with randomly chosen initial centres. At the end of each experiment 10 clusters were obtained. These clusters where then intersected to find out the stable classes. 26 stable classes were obtained at the end of the experiment.

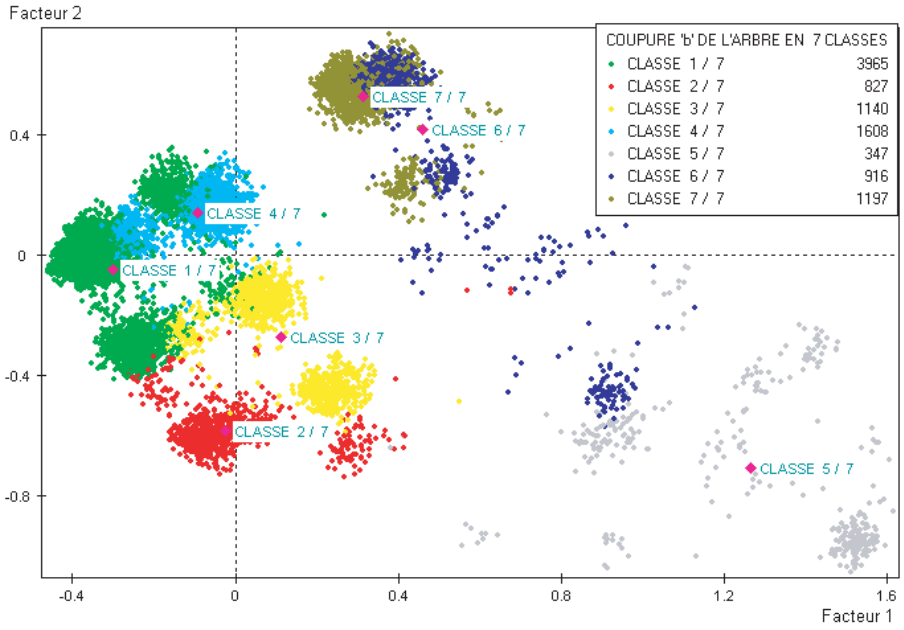

Figure 6: Cluster analysis

These 26 stable classes obtained from k-means clustering were subjected to Ward's agglomerative hierarchical clustering. The PARTI/DECLA module of SPAD was used at this stage. The output of the Ward's clustering are 7 classes or clusters that can be seen in Figure 6. Each cluster contains individuals called parangons that are normally the closest points to the centre of each class and best represent the cluster. The parangons can be seen at the centre of each cluster in Figure 6.

Table 4 presents the constituent elements of the 7 classes obtained from Ward's clustering. Each class contains few traffic elements namely input flows, arc states and paths. The identities of the paths, arcs and input flows are present in column 3 of Table 4. It can be seen in Table 4 that all the classes contain paths and arc states while input flows are present only in class number 2,4,5,6 and 7. The reason being the topology of the network and the short time period used for conducting the simulation which was not large enough for significantly affecting the paths.

Table 4: Seven clusters

Cluster	Frequency	Elements
1	.2868	P31,P2,P4,P18,P23,P50,P29,P40,P45,P36 S33,S34,P39,S28,S25,P9,S6,P38,S15,S5,S10,P13 S1,S14,S38,P12,S40
2	.1848	P17,P24,P31,P29,P2,P4,P40,P44 P51,P39,S4,S39,S25,S13,S34,S9,P9,P18 S28,S31,S10,S1,S32,P35,P13,I1,S8,S19,S17,P12
3	.1092	P17,P24,P28,P5,P1,P32,P39,P44,P40 P51,S31,S36,S27,S9,S30,S39,P36,S4,S18,P6 S3,S13,S29,S26,S42,S12
4	.1732	P5,P1,P28,P50,P32,P23,P18,P39,P40,S12 P45,S36,S30,S27,P36,P9,S6,S33,S26,S5,S15,S3 S29,S14,P13,S21,S9,S37,S32,S35,S8,P12,I2,P57
5	.0347	P34,P8,P3,P43,P24,P30,P51,S4,P17 P41,S13,P35,P33,P32,P4,S28,S25,S34,P44,S31 S42,S49,S32,S39,I1
6	.0917	P33,P49,P41,P1,P5,P38,P37,P28 P35,S37,S7,S27,S16,S36,S30,S31,P18,P43,P23 S12,S3,P57,P9,S13,S42,I3
7	.1196	P37,P49,P30,P41,P32,P18,P23,S7,P2 P4,P45,S16,S37,S34,S25,P36,S28,P9,P57,S15 S33,S1,S21,S52,I1

Each of the classes obtained from hybrid clustering were subjected to canonical correlation analysis for measuring the correlation among the traffic elements i.e. input flows, arc states and paths. The canonical R value for each cluster was found to be greater than 0.56. A threshold value equal to 0.1 was set for testing the correlation between the traffic elements of each class. The correlation coefficient between the traffic elements i.e. input flow vs fastest path and arc state vs fastest path was checked against the threshold value for each class. Only

those elements having correlation coefficient equal to or exceeding the threshold value were retained.

Table 5 presents the canonical correlation analysis results for the fastest paths obtained from the 7 clusters. It can be seen in Table 5 that the fastest paths are more correlated with arc states than the input flows. The reason being the weak correlation coefficient between the input flows and the paths of rank 1 which was found to be less than 0.1 in all the 7 clusters.

Thus, we can say that in this example the time required by a car to join an input node to an output node depends on the state of the arcs but not on the input flows. This does not mean that this condition always holds true because the impact of traffic flows on travel times also depends on the location of the input nodes inside the network.

Table 5: Fastest paths versus arc states

O-D Pair	Fastest Path	Arc States
1-10	1-3-7-10(1)	(1,3)(3)(.11),(2,3)(3)(.103),(3,8)(3)(.295), (4,8)(3)(.27), (5,8)(3)(.31)
	1-3-8-10(1)	(1,3)(1)(.13),(2,3)(1)(.13),(3,8)(1)(.37), (4,8)(1)(.34), (5,8)(1)(.37)
	1-3-8-10(1)	(4,8)(1)(.34),(5,8)(1)(.37),(3,8)(1)(.37)
	1-3-8-10(1)	(5,8)(1)(.37),(3,8)(1)(.37),(4,8)(1)(.34), (1,3)(1)(.13)
1-11	1-3-8-11(1)	Indifferent to arc states
2-10	2-3-7-10(1)	(3,8)(3)(.21),(4,8)(3)(.25),(4,9)(1)(.13), (5,8)(3)(.302)
	2-3-7-10(1)	(3,8)(3)(.21),(4,8)(3)(.25),(4,9)(3)(-.13), (5,8)(3)(.302)
	2-3-8-10(1)	(3,8)(1)(.19),(4,8)(1)(.19), (4,9)(3)(.11), (5,8)(1)(.22)
	2-3-8-10(1)	(3,8)(1)(.19),(4,8)(1)(.19), (4,9)(1)(-.13), (5,8)(1)(.22),(5,9)(3)(.23)
	2-4-8-10(1)	(1,4)(1)(.2107),(2,4)(1)(.1526)
	2-5-8-10(1)	(1,5)(1)(.28),(2,5)(1)(.25),(5,9)(1)(.19)
2-11	2-3-8-11(1)	(1,5)(3)(.203),(2,5)(3)(.174),(5,9)(3)(.23)
	2-3-8-11(1)	(1,5)(3)(.203),(5,9)(1)(-.23)
	2-4-8-11(1)	(1,4)(1)(.24),(2,4)(1)(.16)
	2-4-8-11(1)	(2,4)(1)(.16)
	2-5-8-11(1)	(5,9)(1)(.24),(1,5)(1)(.31),(2,5)(1)(.27)
	2-5-9-11(1)	(1,5)(1)(.31),(2,5)(1)(.27),(5,9)(1)(.24), (4,9)(1)(.13)

We will use the information depicted in Table 5 for generating decision rules which will be used for making predictions about the fastest paths on the network. For instance, using Table 5 we can say that for O - D pair 1 - 10, if we observe that arcs (1,3), (2,3), (3,8), (4,8) and (5,8) have state ranking 3 i.e. their state lies between 0.9× maximum state and maximum state, then the fastest path to take between 1 and 10 is 1-3-7-10. It can also be seen in Table 5 that one path can be the fastest path under two different states of an arc. For instance, consider the path 2-3-8-10 for origin-destination pair 2-10. This path is the fastest path when arcs (3,8), (4,8) and (5,8) have rank 1 and arc (5,9) has rank 3. However for arc (4,9), we observe two ranks 1 and 3. It can be seen that positive correlation (+.11) exists for rank 3 and negative correlation(-.13) exists for rank 1. This means that the path 2-3-8-10 is chosen as the fastest path whenever rank 3 for arc(4,9) is observed and if rank 1 is observed then the path 2-3-8-10 should be avoided.

For the combination of other arc states not presented in Table 5, the correlations between the fastest paths and the arc states was found to be less than 0.1 indicating weak influence of other arc state ranks on the fastest paths of the network. Under these conditions we will consider that the shortest paths are also the fastest paths on the network.

Analysis of Table 5 leads us to the following rules for determining the fastest paths between various origin-destination pairs of the network given any combination of input flows and arc states. In the rules, the rank of arc (i, j) is denoted by $r(i, j)$ and the Boolean operators "equal to" and "not equal to" are denoted by "=" and "≠" respectively. Each rule comprises of several combinations of arc states and/or input flows. These elements are arranged in the rules in the increasing order of their risk for deciding the fastest path.

For instance, consider the rule for origin node 2 and destination node 10.

IF $r(1, 4) = 1$ **AND** $r(2, 4) = 1$ **THEN** the shortest path is (2,4,8,10).

Using this rule we can say that if both the arcs (1,4) and (2,4) are of rank 1, then the path 2-4-8-10 is the fastest path between origin 2 and destination 10.

8.3 Results

The various rules used to make decisions about the fastest paths among the origins and the destinations of the network of Figure 4 are mentioned below. In this set of conditions, the two first conditions must apply. The greater the number of conditions that apply, the lesser is the risk attached to this decision.

ORIGIN 1 - DESTINATION 10

IF $\{r(5, 8) = 3$ AND $r(3, 8) = 3$ AND $r(4, 8) = 3$ AND $r(1, 3) = 3$ AND $r(2, 3) = 3\}$ **THEN** the fastest path is (1,3,7,10).

IF $\{r(5,8) = 1$ AND $r(3,8) = 1$ AND $r(4,8) = 1$ AND $r(1,3) = 1$ AND $r(2,3) = 1\}$ **THEN** the fastest path is (1,3,8,10).

IF $\{r(5,8) = 1$ AND $r(3,8) = 1$ AND $r(4,8) = 1$ $\}$ **THEN** the fastest path is (1,3,8,10).

IF $\{r(5,8) = 1$ AND $r(3,8) = 1$ AND $r(4,8) = 1$ AND $r(1,3) = 1$ $\}$ **THEN** the fastest path is (1,3,8,10).

ORIGIN 1 - DESTINATION 11

Choose always the shortest path (1,3,8,11) as the fastest path.

ORIGIN 2 - DESTINATION 10

IF $\{r(5,8) = 3$ AND $r(4,8) = 3$ AND $r(3,8) = 3$ AND $r(4,9) \neq 3\}$ **THEN** the fastest path is (2,3,7,10).

IF $\{r(5,9) = 3$ AND $r(5,8) = 1$ AND $r(4,8) = 1$ AND $r(4,9) \neq 1$ AND $r(3,8) = 1$ $\}$ **THEN** the fastest path is (2,3,8,10).

IF $\{r(1,4) = 1$ AND $r(2,4) = 1\}$ **THEN** the fastest path is (2,4,8,10).

IF $\{r(1,5) = 1$ AND $r(2,5) = 1$ AND $r(5,9) = 1$ $\}$ **THEN** the fastest path is (2,5,8,10).

IF $\{$ $r(5,8) \neq 3$ AND $r(5,9) \neq 3$ AND $r(1,4) \neq 1$ AND $r(1,5) \neq 1$ $\}$ **THEN** the fastest path is (2,5,8,10).

ORIGIN 2 - DESTINATION 11

IF $\{r(5,9) \neq 1$ AND $r(1,5) = 3$ AND $r(2,5) = 3$ $\}$ **THEN** the fastest path is (2,3,8,11).

IF $\{r(1,4) = 1$ AND $r(2,4) = 1\}$ **THEN** the fastest path is (2,4,8,11).

IF $\{r(1,5) = 1$ AND $r(2,5) = 1$ AND $r(5,9) = 1$ AND $r(4,9) = 1$ $\}$ **THEN** the fastest path is (2,5,8,11).

IF $\{r(5,9) = 1$ AND $r(1,4) \neq 1$ AND $r(1,5) \neq 1$ $\}$ **THEN** choose (2,5,9,11) as the fastest path.

8.4 Validation of the results

The results obtained from the data analysis were validated by comparison with simulation results. The input data (input flows, arc states) were generated

at random for simulation. The value of input flow at node 1 was 70 and at node 2 was 60. The arc state of arc (1,3) was 50 (rank 2), arc (1,4) was 203 (rank 3), arc (1,5) was 80 (rank 1), arc (2,3) was 190 (rank 3), arc (2,4) was 152 (rank 3), arc (2,5) was 60 (rank 1), arc (2,6) was 174 (rank 3), arc (3,7) was 130 (rank 1) , arc (3,8) was 170 (rank 3), arc (4,8) was 55 (rank 1), arc (4,9) was 121 (rank 3), arc (5,8) was 144 (rank 2), arc (5,9) was 135 (rank 2), arc (6,9) was 60 (rank 1), arc (7,10) was 112 (rank 3), arc (8,10) was 96 (rank 2), arc (8,11) was 50 (rank 1) and arc (9,11) was 121 (rank 3). The results of the simulation (fastest paths for each origin-destination pair) are presented in Table 6.

Table 6: Simulation results

Input	Output	Fastest path
1	10	1-3-8-10
1	11	1-3-8-11
2	10	2-5-8-10
2	11	2-5-8-11

Table 7 presents the results of hybrid clustering i.e. rules for finding the fastest path between each origin-destination pair.

Table 7: Results provided by the rules

Input Node	Output Node	Rule	Fastest path
1	10	**IF** {r(5,8) = 1} **THEN** the fastest path is (1,3,8,10).	1-3-8-10
1	11	Choose always the shortest path (1,3,8,11) as the fastest path.	1-3-8-11
2	10	**IF** {r(1,5) = 1 AND r(2,5) = 1 AND r(5,9) = 1 }**THEN** the fastest path is (2,5,8,10).	2-5-8-10
2	11	**IF** {r(1,5) = 1 AND r(2,5) = 1} AND r(5,9) = 1 AND r(4,9) = 1 } **THEN** the fastest path is (2,5,8,11).	2-5-8-11

Let us now validate the data analysis results of Table 7 with the simulation results of Table 6. On verifying the input flow and arc state ranks used in the the rules of Table 7 with the input flow and arc state ranks presented above, we find that arc (5,8) has rank 2 when 1-3-8-10 is chosen as the fastest path between 1 and 10 whereas the rules yield 1-3-8-10 as the fastest path when arc (5,8) has rank 1. In this case, we can see that the simulation results do not match with decision rule results. Let us now consider the results for fastest path between origin 1 and

destination 11. It can be seen in Table 6 that the fastest path between 1 and 11 is 1-3-8-11 which conforms with the Table 7 results and remains uninfluenced by the arc's state ranks. The rules yield fastest paths between 2 and 10 and 2 and 11 as 2-5-8-10 and 2-5-8-11 using rank 1 of arcs (1,5) and (2,5), which also holds true for simulation results (Table 6) and the arc states. Therefore, we can say that the results obtained from the simulation experiment agree with the findings obtained from the hybrid clustering in 75% of the cases.

9. Conclusion

This paper presents a statistical approach for approximating fastest paths under stepwise constant input flows and initial states of the arcs on urban networks. Hybrid clustering and canonical correlation analysis have been used to find arc states and input flows that govern the fastest paths on the network. During the study it was found that once a car arrives at the entrance of the network then the input flows do not play a significant role and it is mainly the arc states that regulate the fastest paths on the network. However, the location of input nodes on the network also affects of the impact of traffic flow on travel times. There are certain arcs called critical arcs whose ranks decide the fastest paths on the network. Therefore, prediction of fastest paths would merely require testing the presence of these critical arc states with the ranks proposed by the rules. If agreement is found, then the next path to follow is the path extracted from the rule otherwise the shortest path is chosen as the fastest path.

Real networks are considerably huge in size and the present approach can be applied to networks of relatively small dimension. The next step of our study is to develop an algorithm for approximating fastest paths on real networks. This would be done by decomposing the real network into small sub-networks and computing the fastest path for the sub-networks using the statistical approach discussed in the paper. The fastest path between any origin destination pair of the real network will be obtained by joining the average fastest paths of the sub-networks.

References

Ambroise, C., Badran, F., Thiria, S., and Sze, G. (2000). Hierarchical clustering of self-organizing maps for cloud classification. *Neurocomputing* **30**, 47-52.

Awasthi, A., Parent, M. and Proth, J. M. (2004). Partitioning algorithms for large scale urban networks, submitted to *Journal of graph algorithms and applications*, November 2004.

Cutting, D., Karger, D., Pedersen, J., and Tukey, J. W. (1992). *Scatter/Gather: a cluster based approach to browsing large document collections*, In *Proc. 15th Annual*

Int. ACM SIGIR Conf., Copenhagen (Edited by N. J. Belkin, P. Ingwersen, and A. M. Pejtersen), 318-329. ACM Press.

Diday, E. and Simon, J. C. (1976). Clustering analysis In *Digital Pattern Recognition* (Edited by K-S. Fu), 47-94. Springer-Verlag, Heidelberg.

Diday, E., Govaert, G., Lechevallier, Y. and Sidi, J. (1978). Clustering in pattern recognition. In *4th International Joint Conference on Pattern Recognition*, Kyoto, Japan.

Diday, E., Lemaire, J., Pouget, P. and Testu, F. (1983). *Elements danalyse des donnes*. dunod.

Dougherty, M. S. (1995). A review of neural networks applied to transport. *Transportation Research* **3**, 247-260.

Everitt, B. S. (1974). *Cluster Analysis*. Heinemann Educational Books.

Hansen, P. and Jaumard, B. (1997). Cluster analysis and mathematical programming. *Mathematical programming* **79**, 191-215.

Jain, A. K., Murty, M. N. and Flynn, P. J. (1999). Data clustering: A Review. *ACM Computing Surveys* **31**, 264-323.

Kisgyorgy, L. and Rillett, L. R. (2002). Travel time prediction by advanced neural network. *Periodica Polytechnica Ser Civil Engineering 46*, 15-32.

Klein, G. and Aronson, J. E. (1991). Optimal clustering: A model and method *Naval Research Logistics* **38**, 447-461.

Kohonen T. (1988). *Self oOganization and Associative Memory*. 2nd edition. Springer-Verlag.

Marcotorchino, J.F., Proth, J.M., and Janesen, J., editors. (1985). *Data analysis in real life environments*. North Holland.

Meila, M. and Heckerman, D.(1998). An experimental comparison of several clustering and initialization methods, In *Proc. Uncertainty in Artificial Intelligence*, 386-395. Morgen Kaufmann.

Murtagh, F. (1995). Interpreting the Kohonen self-organizing feature map using contiguity-constrained clustering *Pattern Recognition Letters 16*, 399-408.

Park, D. and Rillett, L. R. (1998). Forecasting multiple period freeway link travel times using modular neural networks. *Transportation Research Record,1617*. National Research Council, Washington D.C.

Smith, R., Chou, J. and Romeijn, E.(1998). Approximating Shortest Paths in Large Scale Networks with Application to Intelligent Transportation Systems. *INFORMS Journal on Computing* **10**, 163-179.

Tomassone, R., Danzart, M., Daudin, J. J. and Masson, J.P. (1988). *Discrimination et classement*. Masson.

Roux, M., *Algorithmes de classification* (1986). Masson.

Ward, J. H. (1963). Hierarchical grouping to optimize an objective function *Journal of the American Statistical Association* **58**, 236-244.

Zhan, Benjamin F.(1997). Three fastest shortest path algorithms on real road networks: Data structures and procedures. *Journal of Geographic Information and Decision Analysis* **1**, 70-82.

Received April 10, 2004; accepted December 11, 2004.

Anjali Awasthi
Project-IMARA
INRIA Rocquencourt, B.P. 105
78153, Le Chesnay Cedex
France
anjali.awasthi@gmail.com

Yves Lechevallier
Project-AXIS
INRIA Rocquencourt, B.P. 105
78153, Le Chesnay Cedex
France
yves.lechevallier@inria.fr

Michel Parent
Project-IMARA
INRIA Rocquencourt, B.P. 105
78153, Le Chesnay Cedex
France
michel.parent@inria.fr

Jean-Marie Proth
Equipe SAGEP
INRIA Nancy
Ile de Saulcy, UFR MIM Metz
France
proth.jm@wanadoo.fr