Spatio-temporal partitioning of transportation network using travel time data

Clélia Lopez, Panchamy Krishnakumari, Ludovic Leclercq, Nicolas Chiabaut, Hans van Lint

SPATIO-TEMPORAL PARTITIONING OF TRANSPORTATION NETWORK USING TRAVEL TIME DATA

Clélia Lopez, Corresponding Author
Univ. Lyon, ENTPE, IFSTTAR, LICIT
UMR _T 9401, F-69518, LYON, France
Tel : +33 (0)47 204 7051
Email : clelia.lopez@entpe.fr

Panchamy Krishnakumari
Delft University of Technology,
Stevinweg 1, 2600 GA Delft, The Netherlands
Email: p.k.krishnakumari@tudelft.nl

Ludovic Leclercq
Univ. Lyon, ENTPE, IFSTTAR, LICIT
UMR _T 9401, F-69518, LYON, France
Email : ludovic.leclercq@entpe.fr

Nicolas Chiabaut
Univ. Lyon, ENTPE, IFSTTAR, LICIT
UMR _T 9401, F-69518, LYON, France
Email : nicolas.chiabaut@entpe.fr

Hans van Lint
Delft University of Technology,
Stevinweg 1, 2600 GA Delft, The Netherlands
Email: j.w.c.vanlint@tudelft.nl

Submitted in August 2016 for presentation at the Transportation Research Board 96th Annual Meeting, 2017 and publication at the journal of Transportation Research Record

5984 words + 1 Table (1 X 250) + 5 Figures (5 X 250) = 7484 words

Abstract

Nowadays, the deployment of sensing technology permits to collect massive spatio-temporal data in urban cities. These data can provide comprehensive traffic state conditions for an urban network and for a particular day. However, they are often too numerous and too detailed to be of direct use, particularly for applications like delivery tour planning, trip advisors and dynamic route guidance. A rough estimation of travel times and their variability may be sufficient if the information is available at the full city scale. The concept of spatio-temporal speed cluster map is a promising avenue for these applications. However, the data preparation for creating these maps is a challenging and rarely discussed topic. In this paper, we address this challenge by introducing generic methodologies for mapping the data to a Geographic Information System (GIS) network, coarsening the network for reducing the network complexity at the city scale and also estimating the speed from the travel time data, including missing data. We demonstrate this on a large scale urban network of Amsterdam with real travel time data. The preprocessed data is used to build the spatio-temporal speed cluster using three partitioning techniques – *Normalized cut*, DBSCAN and Growing Neural Gas (GNG). A new post-treatment methodology is introduced for DBSCAN and GNG, which are based on data point clustering, to generate connected zones. A preliminary cross comparison of the clustering techniques shows that GNG performs best in generating zones with minimum internal variance, *Normalized Cut* computes 3D zones with the best inter-cluster dissimilarity and GNG has the faster computation time.

Keywords: Graph-partitioning, coarse network, GIS, missing data, homogeneous spatio-temporal regions

1 INTRODUCTION

Graph partitioning is a common challenge in different fields, such as transportation (1; 2) and image segmentation (3). Partitioning a heterogeneous network, such as an urban network, into homogeneous zones can be extremely useful for many applications. We distinguished, at least, three applications that can be improved by using 3D homogeneous regions: (i) traffic control, (ii) macroscopic traffic modeling and (iii) route guidance. (i) For traffic control purpose, different traffic management schemes can be identified for regions of a heterogeneous network (4; 5). Traffic signal control is computationally expensive for a large network (6), especially if the scheme is required to be generated in real-time (7). Developing a scheme at the region scale is a computationally plausible approach than at a links scale. A time adaptive scheme can be considered from the 3D regions. (ii) The Macroscopic Fundamental Diagram (MFD) is more likely to be well-defined in a homogeneous network (8; 9; 10; 11; 12). Refine the equilibrium analysis by zone is a promising approach to investigate the effect of route choice behavior (13). Partitioning methods make it possible to define sub-regions within a network which is essential for a multi-reservoir modeling approach such as the MFD modelling framework (10; 14). This macroscopic scale model facilitates the development of traffic management strategies. (iii) Tour planning, trip advisors and dynamic route guidance can be refined by network partitioning results. The routing models can calculate the least cost routes including the day-to-day 3D regions, i.e. the travel time objective function $f(t,x)$ is refined by the space time properties of the 3D regions. An investigation can be done to identify the best time to start a journey by minimizing the total route time through spatio-temporal network.

To this end, (1) investigates the performance of k-means in partitioning urban networks by considering spatial locations of the road as new features in the data. (2) elaborates a second method based on the definition of a similarity matrix between observations and the application of the *Ncut* algorithm (3). Such regions are defined in 2D, i.e. only measurements of a single time period are considered to identify homogeneous areas. The question of traffic dynamics is only addressed by iterating the algorithms for each time step without direct connections between the 2D clusters (quasi-stationary approximation). It should be noted that the inter-cluster dissimilarity is a central property in these studies because the main targeted application is traffic control.

To perform network partitioning, both the network topology and the link speeds for all time periods are needed. However, real data is often flawed and incomplete, especially for data collected by classic urban measurements. Therefore, data preparation is needed to create a validated dataset for partitioning. The aim of the data preparation is (1) travel time outlier removal, (2) coarsening the large scale network to improve the computation time, and (3) speed estimation including an extension for missing data. In this paper, we introduce methodologies that address these issues. We use real data gathered from Amsterdam urban area. The network topology is derived from both cameras and Geographic Information System (GIS) data from Open Street Maps. In the Amsterdam case, not all speeds for every time period are available. This necessitates the development of algorithms to impute these missing data. We discuss the estimation of missing data and how to minimize the problem of missing data in section 2. Based on a complete (albeit partially imputed) record of all data, we can now use the Amsterdam network to assess the performance of different partitioning techniques.

In this paper, we investigate two classes of clustering techniques – an extension of *Ncut* based on a new expression of the similarity matrix, called *snakes* (1; 2) and classical methods such as DBSCAN and Growing Neural Gas (GNG) from the data-mining field. These clustering techniques are used to construct 3D clusters of road links based on their average speed. We want to ensure that all clusters contains a single connected component (CC) in order all links be reached within the same clusters. This requirement is central for application like travel time estimation or macroscopic traffic modeling. Clustering consists of finding the optimal decomposition of the graph into connected clusters with lowest internal variance of the speed while it retains a reasonable inter-cluster dissimilarity. However, the classical methods that are used in this paper – DBSCAN and GNG produce unconnected clusters. Hence, a new post-treatment method is introduced in this work to generate connected clusters from unconnected clusters in 2D and 3D.

In section 2, we describe the methodology for building both a validated network and speed dataset for partitioning. The construction of the 3D connected clusters using the three clustering techniques along with the post-treatment and cross evaluation criteria are described in section 3. Section 4 provides the results of this preliminary evaluation and section 5 concludes the paper with the major findings of the paper and future recommendations.

2 DATA PREPARATION

2.1 Network Reconstruction

In our study, raw data are the individual travel times from the city of Amsterdam. The data includes the ID and location of the start camera and end camera, recorded individual passing times and travel times for 41 days. There

are a total of 314 unique pairs of start and end camera observations in the data for all the days. The first step to build a network from this dataset is to create a route table. For this purpose, we need to locate the camera on the Amsterdam GIS network obtained from OpenStreetMaps which consists of *147059* links/edges as shown in FIGURE 1(a). The mapping of the camera coordinates to the GIS network is required since these coordinates may not be located in the road, hence location correction is needed. This snapping is done by finding the point-to-segment distance to find the nearest location in the network. Once the coordinates have been snapped to the network graph, we find the path between the start and end camera location. There are various path finding algorithms available in the literature. For this work, the shortest path was found using the algorithm employed in OSRM (OpenStreet Maps Routing Machine) which is based on contraction rules (14). This algorithm is fast and robust for real time applications as well. There were 314 pairs of start and end cameras leading to 314 shortest paths. Mapping these to the GIS Amsterdam network provides a network with 7512 links as shown in FIGURE 1(b).

FIGURE 1 Camera data to validated network

2.2 Network Coarsening

After mapping the shortest path to the Amsterdam GIS network, there are still 7512 links, which is quite large. Hence, we employ network coarsening techniques to remove nodes that satisfy certain criteria. The idea behind coarsening is that a multiscale graph G_{i+1}can be constructed from the previous fine scale graphs G_i by collapsing together the nodes that have similar matching criteria. The matching criteria can differ according to the application. In this paper, the matching criterion relates to the differences in edge weights. In our case, these weights represents the speed of each link/edge. If the edges have the same weight, the node that connects the edges will be collapsed/removed. The network coarsening in this paper is based on a constrained version of contraction hierarchies (14).

The construction of the coarsened graph in this work is based on three steps: (a) the nodes are prioritized or ranked for contraction, i.e. a node with a lower rank will be removed before a node with a higher rank, (b) the contraction rules are determined based on the edge difference, and (c) the new weights of the link for the coarse graph are calculated. In this work, the contraction rules are governed by the edge weights and the connectivity of the network. The weights for each link/edge is the estimated speed of each link. The weight w_{uv} of link (u, v) is the speed of the link s_{uv}. As we only require speed for one time slice for assigning the weights, a peak period time slice (16:00) was chosen. This is because the network will exhibit the most variance during peak period with most links having different speeds at this time. If we chose a non-peak period, these variances will be smoothed out. A more detailed description of node ranking and the contraction rules are given below.

Node Ranking

The order in which the nodes are removed is important for graph coarsening. There are different node ranking or ordering techniques as introduced in relation to contraction hierarchies (14). In this work, the nodes are ordered based on their densities. The more number of neighbors the node has, the higher the rank is, and the lower the chance that the node will be removed. A dense node might connect a lot of edges and might be important for transportation purpose application. Given a graph $G = (V, E)$ with node set V and an edge set E. Suppose, $(u, v) \in E$ where $u, v \in V$ then the rank of the nodes u and v will satisfy the following condition:

$$\boldsymbol{r(u) > r(v)}\ ,\ \boldsymbol{if\ n(u) > n(v)} \qquad (1)$$

where $n(u)$ and $n(v)$ are the number of neighbors of node u and v respectively. Thus, based on the contraction rule, v will be contracted before u. The neighbors of the node are found by determining all the incoming and outgoing links from the node. A link (u, v) is said to be incoming w.r.t node v if v is the target node of the link and outgoing if v is the source node. Once the neighbors are found for all the nodes, the nodes are sorted based on their ranks in ascending order with the lowest rank first.

Contraction Rules

Once the nodes are sorted according to the rank, the contraction rules are used to remove them. First, a criterion has been set to decide if the given node is eligible for contraction to decrease the complexity. The criteria are that if the node removal results in the same or more number of links than before removal, then the node is not removed. TABLE 1 presents an overview of different cases of node contraction. The link color represents weights; different color implies different weights. For example, in case (5) of TABLE 1, removing the nodes results in 4 new links which is the same number of links as that before node contraction. Hence, this node will not be considered for removal. It was also observed that contracting nodes with more than four neighbors, in the majority of the case, leads to 5 or more links. Therefore, an initial constraint has been imposed on the node contraction only to contract nodes with neighbors less than or equal to 4 links as given below:

$$\boldsymbol{c_1(v)} = \begin{cases} \boldsymbol{1},\ \boldsymbol{n(v) \le 4} \\ \boldsymbol{0},\ \boldsymbol{otherwise} \end{cases} \qquad (2)$$

A given node v will be contracted only if $c_1(v)$ is equal to 1; where $c_1(v)$ is the criterion. When the initial criterion has been satisfied for a given node v, edge difference is then used as the next rule for inclusion or exclusion of that node for contraction.

$$\boldsymbol{c_2(u, v, w)} = \begin{cases} \boldsymbol{1},\ \boldsymbol{w_{uv} - w_{vw} \le threshold} \\ \boldsymbol{0},\ \qquad\qquad \boldsymbol{otherwise} \end{cases} \qquad (3)$$

Here, W_{uv} and W_{vw} are the weights of the links (u, v) and (v, w) respectively. In this work, the weight is the estimated speed for each link. In our work, we set the threshold to 0, so that only if the links have the same speed, nodes can be contracted.

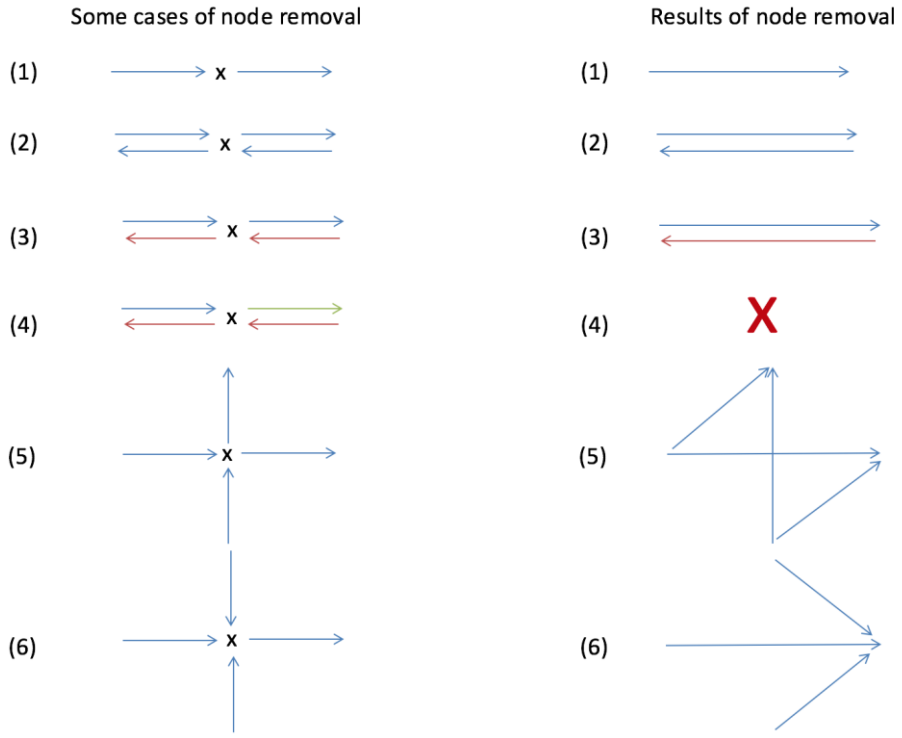
Based on these two criteria, we will now discuss the rulesets for node contraction. Given the nodes that have been ranked, the steps for coarsening are as follows:

1. Set the node with the lowest rank as v.
2. Check if $c_1(v)$ is satisfied.
3. If it is satisfied, we find all the incoming and outgoing links of node v. If $c_1(v)$ returns 0, go to step 9.

4. Pair up all the incoming and outgoing links while considering the direction. For example, in case (6) of TABLE 1, if (u, v), (x, v) and (y, v) are the incoming links and (v, w) the outgoing link, the pairs are (u, v, w), (x, v, w) and (y, v, w).
5. U-turn is not considered in this work, thus (u, v, u) is not considered as a valid pairing as shown in case (2) of TABLE 1.
6. Check if c_2 criteria is satisfied for all the valid incoming-outgoing pairs.
7. If one of the pairings does not satisfy the condition, assign the next node in the rank list as v and repeat from step 2.
8. If c_2 is satisfied for all pairs, then we remove the node from the node list and update the node ranking as new links are formed. Repeat from 1.
9. Stop the iteration.

In the case (4) of TABLE 1, the node cannot be contracted as the edge difference did not satisfy the threshold criteria. The node coarsening is applied to the mapped Amsterdam network with 7512 links. As explained before, the peak period was estimated to be around 4pm and hence the speed at this time slice is used for the network coarsening. The new coarsened network contains 411 links as shown in FIGURE 1(c).

TABLE 1 Examples of contraction rules



2.3 Speed Estimation

This section described the methodology to estimate the link speed based on individual travel times recorded by the cameras. First, a cleansing process will remove outliers. Second, the speed will be estimated for each time period.

The moving average process has been used to remove travel time outliers. There are usually alternative paths for a given Origin-Destination (OD). It can be k-shortest paths, representing the common route choices. We defined travel times significantly higher than the distribution as outliers. To remove these outliers, we propose two approaches: (1) treat the travel times higher than the third quantile as outliers. In order to keep the dynamics, distributions can be split by periods. The disadvantage of this approach is that distribution is sensitive to the number of observations. Thus, outliers can be smoothed into time periods with few travel times. (2) The approach that we used for removing the outliers is based on the moving average process. We define the moving average $\bar{\tau}$ as

$$\bar{\tau}_n = \frac{1}{k}\sum_{i=0}^{k-1} \tau_{n-i} \quad (4)$$

where τ_n is the n^{th} realized travel time. Outliers are defined by $\bar{\tau}_n + \Delta\tau$, where $\Delta\tau$ is the travel time window. In our study, we set $\Delta\tau$ to the standard deviation of the peak demand. $\Delta\tau$ has be tuned by a graphical inspection of the effects of $\Delta\tau$ on the number of outliers. We considered only the upstream window to remove travel time outliers as the downstream window is not relevant for our study, i.e. we do not consider that fast travel times could be outliers. The travel time window is refined after two iterations. FIGURE 2 gives an example with the x axis being the observed time and y axis the travel time. The red curve is the moving average and the black curve represents the upstream travel time window.

Individual travel times have been aggregated by period. We consider the mean of individual travel times for a given OD at a given period.

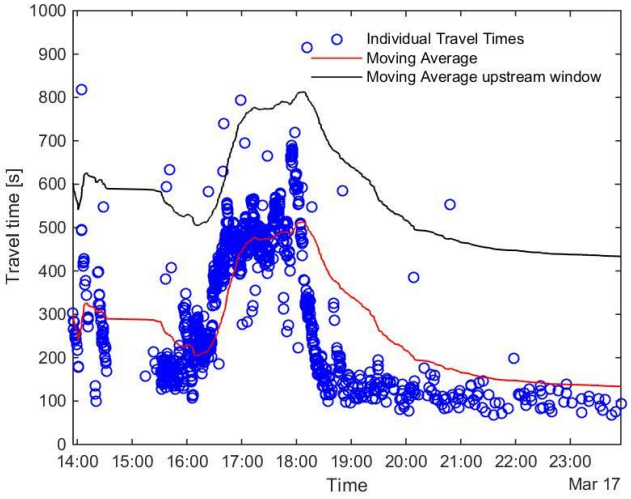


FIGURE 2 Travel times exceeding the travel time window (black curve) are considered as travel time outliers

An OD denoted by od_i is characterized by a succession of links $L_i = (l_1, l_2, \dots, l_n)$, i.e., represents a path. The first shortest path has been used to keep speed estimation zones compact. During a given period t, the travel time of od_i is $tt_{(i,t)}$. Then, it is easy to estimate the speed of the path i by $s_i = \frac{dist(L_i)}{tt_{i,t}}$. It can happen that a given link l_k belongs to different OD's paths, i.e. $l_k \in (od_1, od_2, \dots, od_m)$. For this case, its speed is defined as the mean of the corresponding speeds of the paths.

When no data is available to measure $tt_{(i,t)}$, we distinguished three approaches to estimate the speed: (1) set links without data to the limited speed, which does not consider traffic lights. We assume that setting limited speed at links without data will increase the variance of the zones. (2) Compute new speed value through the speeds of a specific neighborhood. We assumed that the average speed of a given specific neighborhood will smooth the speed. Thus, congestion pockets can be less homogeneous and location zones can be less identifiable in space and time. The phenomenon is more common with link belonging to the boundaries of the zones, i.e. the link without data is being connected with links from different zones. (3) Duplicate speed from an identified link. We assumed that the duplication of speed minimize the speed variance of zones. We use the third approach by using a cost function. Let $G = (V, E)$ be the weighted graph where N_v and N_e are the numbers of vertices and edges respectively, A is the directed adjacency matrix of the network, C is the directed weight matrix and D is the cost paths matrix. A represents relationship of the finite graph G. In our study, G is the graph of the 3D network, i.e. a repetition of the same network at N_t time slices. Through the time, a vertex v_i is connected to itself at time steps $t+1$ and $t-1$ where $t \in (1, N_t)$. Its edge weight is set to the period duration. Through the space, we denote the edge weight by c_{ij} ,where i and j are connected links. Directions are considered to set the edge weight. For a given edge, if vertex j named v_j is downstream to v_i then the edge weight $c_{ij} = \frac{length(l_j)}{s_j}$, else if vertex v_j is upstream to v_i then the weight cost $c_{ij} = \frac{length(l_j)}{w}$, where w is the backward wave speed and it is set to 5 m/s in this study. If no data is available for v_j which is downstream to v_i, then $s_j = \max(s) + \frac{\max(s)}{2}$. Dijkstra's algorithm (16) is used to estimate the shortest path between two vertices i and j in G. The cost of the shortest path is denoted by D_{ij}. Thus, the link

without data will be assigned the speed of the most relevant link. The most relevant link is identified as the link that minimizes the cost function. For computation efficiency, the process is performed strictly for links without data and a time window is used to constrain the search for the relevant link.

2.4 Experimental setup

The data preparation process - the coarsening methodology and the speed estimation of the link - considers a weighted directed network. The partitioning methods used in this paper requires strongly connected graph, i.e. a directed path exists for every pair of vertices. A real network is strongly connected when a vehicle can reach any link from any starting point. For both fine and coarse resolutions networks, this constraint is not true. Thus, direction is not a convenient attribute to partition the Amsterdam network.

Our study application focuses on only one-day data. This given day is a common weekday. The analyzed period is from 7am to 5pm. It is a time window of 8 hours, one third of a day representing the morning peak demand. After analyzing the travel time data, it was found that not all of the ODs are used on all days. Only used ODs have been considered for reconstructing the network based on the shortest path finding in the network with 411 links. They are mapped to create a new coarsened network as explained in Section 2.1. From the used ODs, around 16% of data are missing.

3 SPATIO-TEMPORAL PARTITIONING TECHNIQUES

3.1 Normalized Cut based on *snakes* similarities

One of the contributions of this paper is to adapt the existing methodology of *snakes* for a spatio-temporal network, i.e. a repetition of the same network at numerous time slices. A s*nake* (2) is composed of a sequence of links, which iteratively grows by adding adjacent link that are similar to itself. The connectivity is ensured by a link addition constraint which considers links strictly belonging to the neighborhood of the *snake*. Let S_i be the snake initialized by the link l_i where $S_{ik} \in S_i$ is the subset containing the first k elements. For a fixed time, the neighborhood of S_{ik} of a given snake is defined as the links spatially connected to it. We consider this neighborhood both in space and time. The link l_i at time t is denoted by $l_{(i,t)}$. We make duplicates of link $l_{(i,t)}$ at instants $t-1$ and $t+1$, denoted by $l_{(i,t+1)}$ and $l_{(i,t-1)}$, respectively. The *snake* similarities are defined as follows (2)

$$\boldsymbol{w_{ij} = \sum_{k=1}^{N} p^k \, intersect(S_{ik}, S_{jk})} \quad (5)$$

where the weight coefficient p is fixed to $p \leq 1$. Let W be the *snakes* similarities matrix with $W(i,j) = w_{ij}$. *NCut* (3) is a measure of dissociation based on this W. The complexity of *NCut* is NP-complete.

One of the main inconveniences of snake is its heavy computational cost. The complexity to run a *snake* is $O(n)$ for the best case and $O(n^2)$ for the worst case, where $n = N_l$ and $n = N_l * N_t$ for a spatial *snake* and a spatio-temporal *snake* respectively. The complexity of running n snakes is $O(n^2)$ for the best case and $O(n^3)$ for the worst case. The growing snake algorithm search its neighbors iteratively. The neighborhood size depends on the growing snake pattern and the network topology. In particular, the neighborhood of a snake growing in a corridor is equal to 2 (the upstream and the downstream neighbors of the current snake). It is much smaller than the neighborhood of a snake growing in a strongly connected network, i.e. a network where every nodes have a link with every other nodes. These both examples correspond to the best and the worst cases. Their associated complexity are respectively $O(n^2)$ and $O(n^3)$. Any realistic situation is between these boundaries. For example, we numerically investigated the complexity of the algorithm in our case study. We show that it is $O(n^{2.2})$, where 2.2 was obtained by a regression of the complexity. It can be seen that the similarities decrease exponentially with the weight coefficient p. A sensitivity analysis was done to investigate the performance of *snakes* for different lengths. Results show that the quality of *NCut* partitioning is independent of the *snake* length. Nevertheless, the *snake* length - k - has to be set at a minimal threshold to keep the connectivity. Thus, a too short *snake* cannot discover the entire topology of the network in both space and time. The short *snake* length may also provide clusters results where a cluster contains links that are not all connected with each other. Therefore, we set the *snake* length to 38% of the spatio-temporal size of the network. In addition, the weight coefficient is set to $p = 0.8$ in our study.

3.2 Partitioning based on Data Points Clustering

The two other classic clustering methods that are considered for this study are: (i) GNG (17) and (ii) DBSCAN (18). We represented the 3D network into a data set containing four variables, link coordinates with their corresponding speed and time measurement (x, y, t, s). The four quantitative variables have been normalized. After normalization,

we multiply the speed column by a fixed coefficient equal to 3 to be sure that speed is the predominant variable over spatial and temporal coordinates during clustering.

(i) DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based method. DBSCAN has two user-specified parameters. The radius parameter $\varepsilon > 0$ specifies the radius neighborhood and the MinPts parameter specifies the density threshold of dense regions. For our study, parameters have been set to $\varepsilon = 0.01$ and $\text{MinPts} = 50$.

(ii) GNG is an Artificial Neural Network variant of Neural Gas (19). GNG begins with two neurons and the network grows during the execution of the algorithm. GNG has been adapted for clustering through a two-step process: running GNG and reconstructing data point clusters based on GNG centroids. The user-specified parameters are the number of centroids N, the maximum number of iteration m, L, the adaptation threshold ε_b, ε_n, the neighborhood size α, δ, the time T, which have been set as $N = 10$, $m = 20$, $L = 50$, $\varepsilon_b = 0.2$, $\varepsilon_n = 0.005$, $\alpha = 0.5$, $\delta = 0.995$, $T = 50$.

Post-treatment

The clustering results provide clusters which are not connected as shown in FIGURE 3(a). The homogeneous zone partition needs to be a single connected cluster. Therefore, post-treatment is needed on the clusters that is obtained from DBSCAN and GNG to obtained connected clusters with minimum inter-cluster speed variance. There are three steps for the post-treatment algorithm and these are:

- Identifying the CCs in each of the cluster
- Assigning the biggest CCs as the initial clusters
- Assigning all the other CCs to the initial clusters

Given that there are N number of clusters from the data point clustering methods, there might be more than one CC for each cluster as shown in FIGURE 3(a). Ideally each cluster should contain a single CC. In order to achieve this, all CCs within each cluster are identified. Then, these CCs are sorted according to the CC size. Given that the target cluster size is M, the biggest M CCs from different clusters are chosen as the initial cluster. This process is illustrated with a simple example in FIGURE 3. In FIGURE 3(a), there are 2 CCs in blue cluster, 1 CC in red, 1 CC in green and so on. It can be clearly seen that there are 2 CCs for the blue cluster.

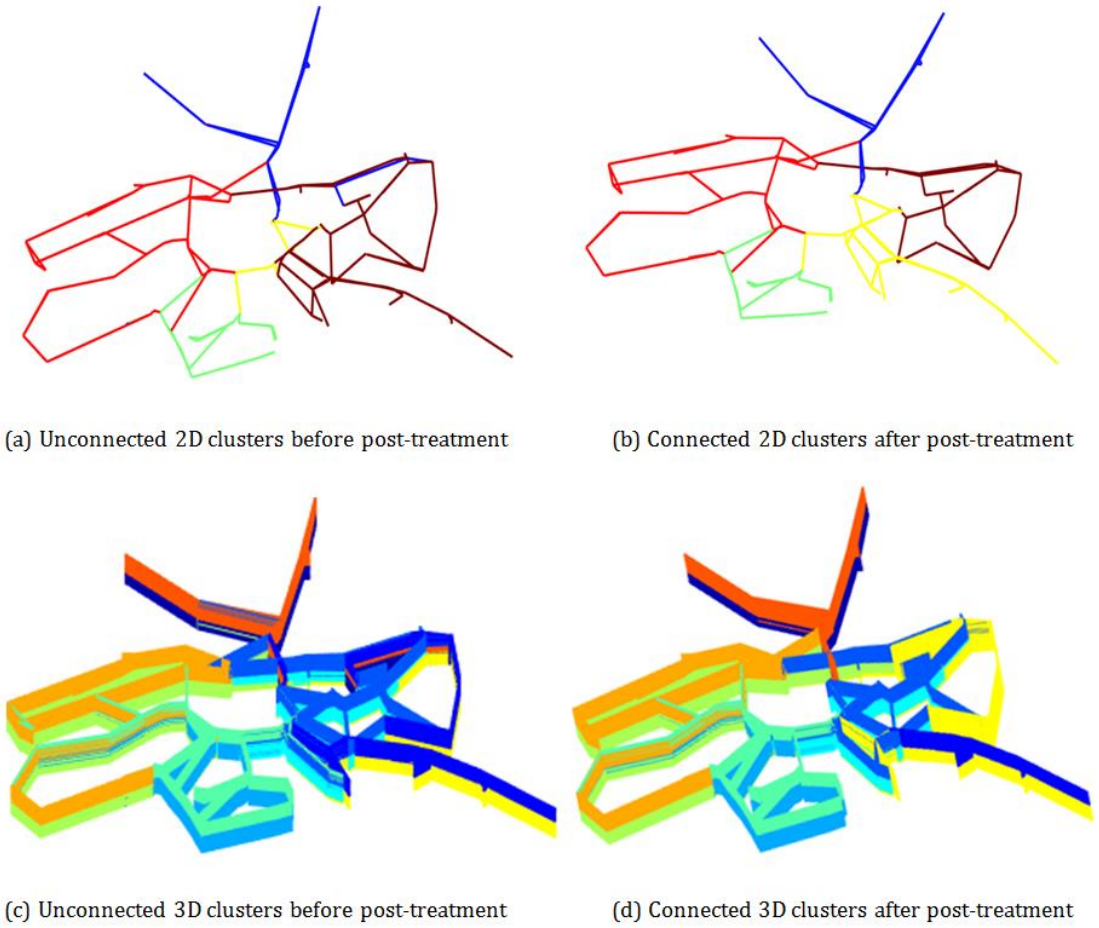
Assuming that there are a total of X CCs from all the clusters, there are $(X - M)$ clusters that still need to be assigned to one of the initial cluster. The rest of the $(X - M)$ CCs are merged with the M initial cluster and the following two parameters are found for each pair.

$$\boldsymbol{c(x,m)} = \begin{cases} \mathbf{1}, & \boldsymbol{x \cap m \text{ is connected}} \\ \mathbf{0}, & \boldsymbol{x \cap m \text{ is not connected}} \end{cases} \quad (6)$$

$$\boldsymbol{v(x,m)} = \begin{cases} \boldsymbol{variance(s_x, s_m)}, & \boldsymbol{x \cap m \text{ is connected}} \\ \infty, & \boldsymbol{x \cap m \text{ is not connected}} \end{cases} \quad (7)$$

where $x \in (X - M)$ CCs, $m \in M$ initial CCs, s_x and s_m are the speed of each cluster. Once the $c(x,m)$ and $v(x,m)$ have been calculated for all the CC pairs, the $(X - M)$ CCs are sorted in decreasing order according to the $\sum c(x,m)$ for all $m \in M$ CCs. The x CCs with the largest $\sum c(x,m)$ is selected and it is merged with the initial cluster m that has $c(x,m) = 1$ and have the smallest variance $v(x,m)$ among all the $m \in M$. Once merged, this cluster is removed from the $(X - M)$ CCs and $c(x,m)$ and $v(x,m)$ is updated for all $(X - M)$ CCs as the initial clusters are updated. This process is repeated until all the $(X - M)$ CCs are assigned to the M initial clusters.

FIGURE 3 shows an example of post-treatment results in 2D and 3D. FIGURE 3(a) shows the cluster before post-treatment and FIGURE 3(b) shows post-treatment with the same number of clusters as the input for a single time slice. FIGURE 3(d) shows the result of post-treatment in 3D with same number of clusters as the input which is shown in FIGURE 3(c). There is no difference in post-treatment methodology between 2D and 3D. The only difference is in calculating the 3D adjacency matrix for finding the 3D CCs and the connectivity. The 3D adjacency matrix is defined by creating bi-directional links between the time slices.



(a) Unconnected 2D clusters before post-treatment (b) Connected 2D clusters after post-treatment

(c) Unconnected 3D clusters before post-treatment (d) Connected 3D clusters after post-treatment

FIGURE 3 Post-treatment results for data point clustering methods

3.3 Evaluation Metrics

The three clustering techniques have been compared using three indicators in this study: (i) Total Variance normalized (TNn), (ii) Connected Clusters Dissimilarity (CCD), and (iii) time computation. (i) TV is the original indicator of the *snake* similarities partitioning and is defined as $TV = \sum_{A \in C} N_A * Var(A)$ (1). TV has been normalized using the following equation:

$$\boldsymbol{TVn} = \frac{\mathbf{1}}{\boldsymbol{N}} \frac{\sum_{A \in C} \boldsymbol{N_A} * \boldsymbol{Var(A)}}{\boldsymbol{S^2}} \quad (8)$$

This indicator is based on the assumption that a given cluster is composed of links characterized by similar speeds. The speed variance is highlighted. (ii) The second metric used is the CCD. The criterion is the dissimilarity between a given cluster and its neighboring cluster, i.e. clusters touching the given cluster. CDD is defined as follows:

$$\text{CCD} = \frac{\sum_{i=1}^{n} \sum_{k=1+i}^{n} \boldsymbol{\delta_{ik}} |\bar{x}_i - \bar{x}_k|}{\sum_{i=1}^{n} \sum_{k=1+i}^{n} \boldsymbol{\delta_{ik}}} \quad (9)$$

$$\boldsymbol{\delta_{ik}} \begin{cases} \mathbf{1} & \boldsymbol{if\ k\ and\ i\ are\ connected\ clusters} \\ \mathbf{0} & \boldsymbol{otherwise} \end{cases} \quad (10)$$

(iii) The time computation indicator evaluates the computational cost of the algorithms. The complexity has to be considered for another size of network and number of time slices. Two different field of partitioning methods have been compared: (1) *NCut* from the graph theory based on the *snakes* similarities and (2) DBSCAN and GNG

from data point clustering. The basic data point clustering methods are faster but a post-treatment process is required. The computational cost of the post-treatment is heavy because it checks the connectivity of the previous results and iteratively updates the clusters. The time computation evaluation includes both field partitioning methods and all the internal processes.

4 RESULTS AND DISCUSSION

In this section, results from the three methods are analyzed and compared - *NCut* based on *snakes* similarities, DBSCAN with post-treatment and GNG with post-treatment. The comparison focuses on two conceptually different fields to partition a transportation network. FIGURE 4 illustrates the partitioning results from three methods under a fixed number of clusters – set to 2. Both the data point clustering methods do not produce the same spatio-temporal zones by the post-treatment algorithm. The zones shapes present a reasonable 3D covers, i.e. a given zone is roughly compacting by space and time. We use three indicators to evaluate the three methods. TVn and CCD measure the quality of clusters representing the homogeneous zones and the inter-cluster dissimilarity respectively. The time computation quantifies the computational cost applied to our case study. FIGURE 4(d,e) shows the TVn and CCD for a systematic number of clusters from 2 to 9. It can be observed that GNG is the best method that minimizes the TVn. *NCut* is the best method that maximizes the CCD. However, the time computation for GNG is found to be the fastest.

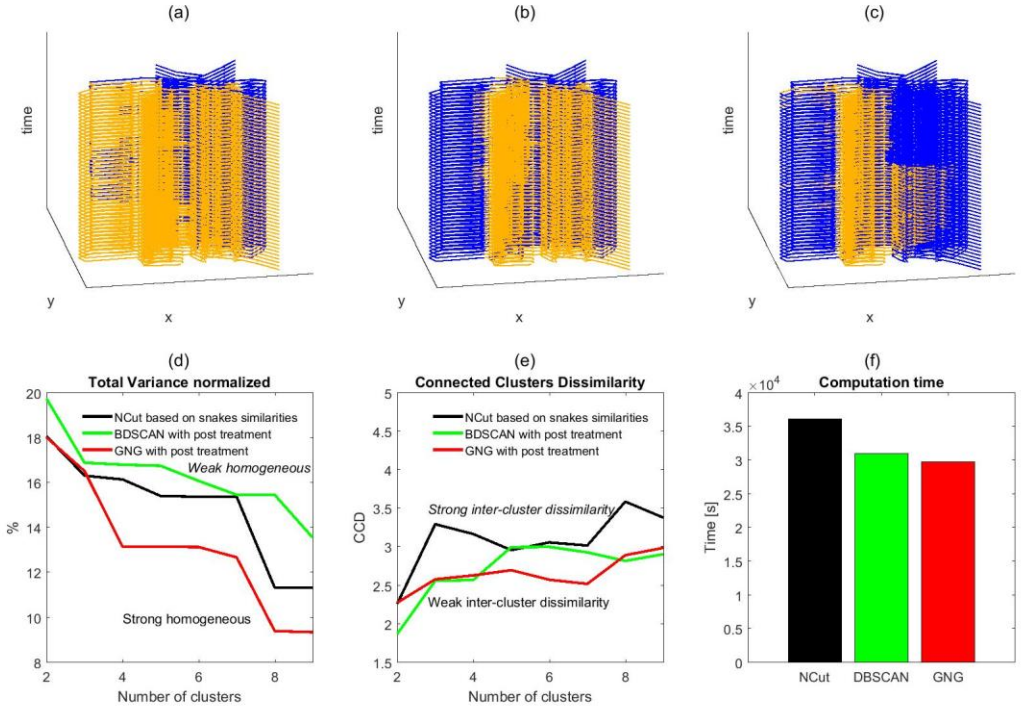


FIGURE 4 Amsterdam network (7am to 5pm) where 3D clusters ($n = 2$) obtained with (a) *NCut* based on snakes similarities, (b) DBSCAN with post-treatment, (c) GNG with post-treatment; and a comparison of the three partitioning methods by (d) TVn, (e) CCD and (f) the computation time

FIGURE 5(a,c,e) illustrates the GNG partitioning with different number of clusters ranging from two to four. FIGURE 5(b,d,f) are histogram plots of links speed for each cluster. Note that the maximal speed is around 40 m/s, corresponding to highway. The validated Amsterdam network used in this work contains both provincial roads and highways. The histogram validates that each cluster has speed variance as low as possible. For example, in FIGURE 5(d), most of the links in the blue cluster have speed of 5 to 10m/s. Only a few of the links in the blue cluster have 0 to 5m/s compared to the orange cluster. This observation remains valid for all the three cases shown here proving the hypothesis that the cluster results from post-treatment provides clusters that are connected and that minimizes the speed variance.

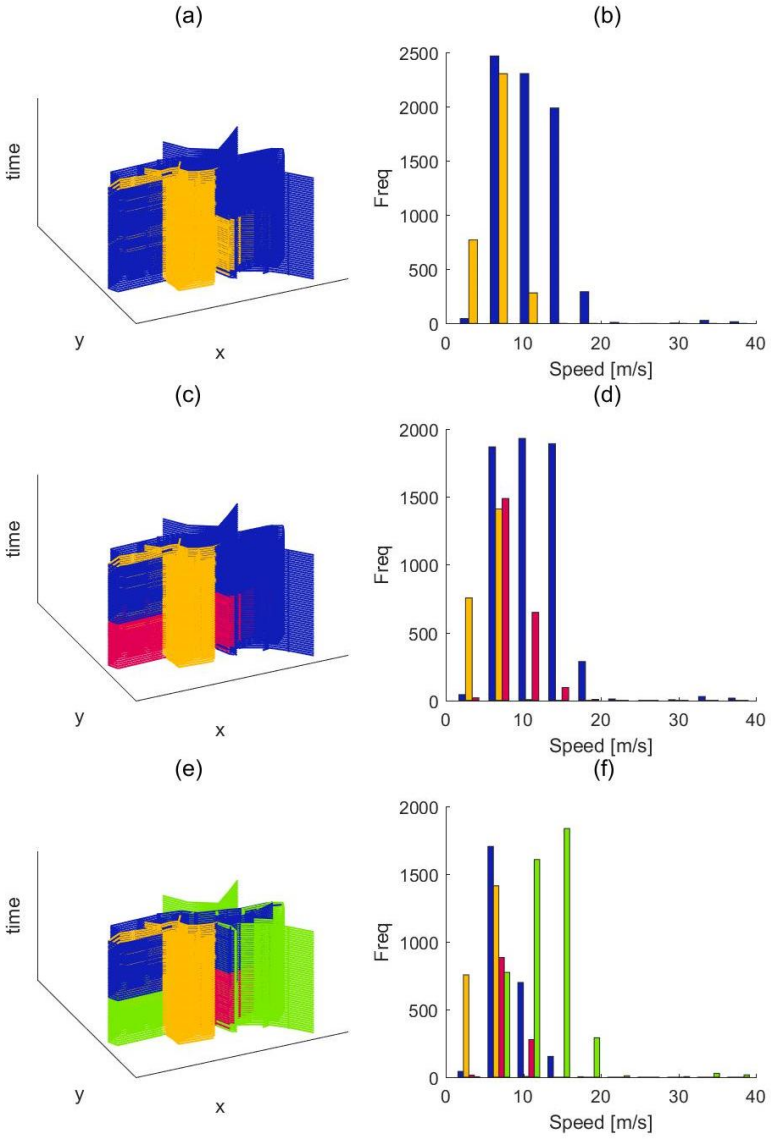


FIGURE 5 (a,c,e) 3D network visualization of the GNG partitioning into two to four spatio-temporal zones and (b,d,f) its corresponding histograms

5 CONCLUSION AND FUTURE WORK

This research describes a generic methodology to prepare data for transportation application. We reconstructed the network from GIS based on the coordinates of cameras and their corresponding recorded travel times. We also introduced coarsening techniques to reduce computational complexity. The speed has been estimated from incomplete and flawed individual travel times. The validated network along with the estimated speed has been used for partitioning.

Two different concepts of spatio-temporal partitioning of a transportation network has been compared in this work. We implemented methods belonging to different fields: (i) unsupervised learning and (ii) graph theory. (i) Clustering approach considers links under a network as data points. This hypothesis allows us to implement simple and efficient clustering algorithms but it requires post-processing for contiguity, i.e. all clusters should be composed of a unique CC. (ii) We considered transportation network at numerous time slices as a graph. In this case, connectivity is considered through graph topology.

This paper has presented three methodologies to partition a network both in space and time, which is demonstrated in a coarsened Amsterdam city network. The partition criterion is the speed. The comparison between

the three techniques has been evaluated by two metrics: TVn and CCD. Preliminary results show that none of the partitioning method is better w.r.t both metrics. TVn measures the homogeneous zones which can be spread throughout the 3D network keeping the connectivity. CCD indicator focuses on the inter-cluster dissimilarity but can forsake the homogeneity. The choice of a spatio-temporal partitioning method is a compromise among both criterions.

There are various future directions that can be pursued. The methodology can be improved to be more generic and computationally efficient. A more extended comparison study needs to be implemented. Also, we have only looked at one day in this work and the methodology can be iterated for several days. Another application that is promising for creating these spatio-temporal speed regions is that it can be used to derive future travel times. These claims still need to be researched and validated.

ACKNOWLEDGMENTS

The authors would like to thank Sjoerd Linders and AMS for providing the data for this study. This research is supported by the region Auvergne-Rhône-Alpes (ARC7 Research Program), by the réseau franco/néerlandais - Bourse Eole and partly by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 646592 – MAGnUM project). We also acknowledge the support of the SETA project funded from the European Union's Horizon 2020 research and innovation programme under grant agreement No 688082. The authors would like to thank Zhengui Xue for proof reading this paper.

REFERENCES

1. Ji, Y., Geroliminis, N., 2012. On the spatial partitioning of urban transportation networks. Transportation Research Part B, 46, 1639-1656.

2. Saeedmanesh, M., Geroliminis, N., 2016. Clustering of heterogeneous networks with directional flows based on "Snake" similarities. Transportation Research Part B: Methodological, 91, 250-569.

3. Shi, J., Malik, J., 2000. Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22 (8), 888-905.

4. Haddad, J., Ramezani, M., Geroliminis, N., 2013. Cooperative traffic control of a mixed network with two urban regions and a freeway. Transportation Research Part B: Methodological, 54, 17-36.

5. Ramezani, M., Haddad, J., Geroliminis, N., 2015. Dynamics of heterogeneity in urban networks: aggregated traffic modelling and hierarchical control. Transportation Research Part B: Methodological, 74, 1-19.

6. Ma, Y-Y., Chiu, Y-C., Yang, X-G., 2009. Urban traffic signal control network automatic partitioning using laplacian eigenvectors. Intelligent Transportation Systems Conference, 1-5.

7. Peeta, S., Ziliaskopoulos, A.K., 2001. Foundations of Dynamic Traffic Assignment: The Past, the Present and the Future. Networks and Spatial Economics, 1 (3), 233-265.

8. Godfrey, J. W., 1969. The mechanism of a road network. Traffic Engineering and Control, 11:323–327.

9. Mahmassani, H.S., Williams, J.C., Herman, R., 1987. Performances of Urban Traffic Networks. In: (Gartner, N.H., Wilson, N.H.M., Eds) Proceedings of the 10th International Symposium on Transportation and Traffic Theory, 1-20.

10. Geroliminis, N., Daganzo, C.F., 2008. Existence of urban-scale macroscopic fundamental diagrams: some experimental findings. Transportation Research Part B, 42(9):759-770.Godfrey, J.W., 1969. The mechanism of a road network. Traffic Engineering and Control, 11, 323–327.

11. Buisson, C., Ladier, C., 2009. Exploring the impact of homogeneity of traffic measurements on the existence of Macroscopic Fundamental Diagrams. Transportation Research Record: Journal of the Transportation Research Board, 2124, 127-136.

12. Leclercq, L., Chiabaut, N., Trinquier, B., 2014. Macroscopic Fundamental Diagrams: A cross-comparison of estimation methods. Transportation Research Part B: Methodological, 62, 1-12.

13. Yildirimoglu, M., Ramezani, M., Geroliminis, N., 2015. Equilibrium analysis and route guidance in large-scale networks with MFD dynamics. Transportation Research Part C: Emerging Technologies, 59, 404-420.

14. Daganzo, C. F. Urban gridlock: Macroscopic modeling and mitigation approaches. Transportation Research Part B: Methodological, 2007, 41, 49-62.

15. Geisberger, Robert, et al. "Contraction hierarchies: Faster and simpler hierarchical routing in road networks." International Workshop on Experimental and Efficient Algorithms. Springer Berlin Heidelberg, 2008.

16. Dijkstra, E.W., 1959. A note on Two Problems in Connexion with Graphs. Numerische mathematik, 1, 269-271.

17. Fritzke, B., 1995. A growing Neural Gas Network Learns Topologies. Advances in Neural Information Processing Systems 7, 625-632.

18. Ester, M., Kriegel, H.P., Sander, J., Xu, X., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the 2nd International Conference on Knowledge Discovery and Data mining, 226–231.

19. Martinetz, T., Schulten, K., 1991. A "neural gas" network learns topologies. Artificial Neural Networks, 397–402.