Spatiotemporal Partitioning of Transportation Network Using Travel Time Data

Article · October 2017

DOI: 10.3141/2623-11

5 authors, including:

Clélia Lopez
Ecole Nationale des Travaux Publics de l'Etat
6 PUBLICATIONS **3** CITATIONS
SEE PROFILE

Ludovic Leclercq
Institut Français des Sciences et Technologies …
89 PUBLICATIONS **1,351** CITATIONS
SEE PROFILE

Nicolas Chiabaut
Ecole Nationale des Travaux Publics de l'Etat
44 PUBLICATIONS **429** CITATIONS
SEE PROFILE

J.W.C. Van Lint
Delft University of Technology
173 PUBLICATIONS **2,403** CITATIONS
SEE PROFILE

Spatiotemporal Partitioning of Transportation Network Using Travel Time Data

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

Today, the deployment of sensing technology permits the collection of massive amounts of spatiotemporal data in urban areas. These data can provide comprehensive traffic state conditions for an urban network and for a particular day. However, data are often too numerous and too detailed to be of direct use, particularly for applications such as delivery tour planning, trip advisors, and dynamic route guidance. A rough estimate of travel times and their variability may be sufficient if the information is available at the full city scale. The concept of the spatiotemporal speed cluster map is a promising avenue for these applications. However, the data preparation for creating these maps is challenging and rarely discussed. In this study, that challenge is addressed by introducing generic methodologies for mapping the data to a geographic information system network, coarsening the network to reduce the network complexity at the city scale, and estimating the speed from the travel time data, including missing data. This methodology is demonstrated on the large-scale urban network of Amsterdam, Netherlands, with real travel time data. The preprocessed data are used to build the spatiotemporal speed cluster by using three partitioning techniques: normalized cut, density-based spatial clustering of applications with noise, and growing neural gas (GNG). A new posttreatment methodology is introduced for density-based spatial clustering 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, the normalized cut computes three-dimensional zones with the best intercluster dissimilarity, and GNG has the fastest computation time.

Graph partitioning is a common challenge in several 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. At least three applications are distinguished that can be improved by using three-dimensional (3-D) homogeneous regions: traffic control, macroscopic traffic modeling, and route guidance:

1. For traffic control purposes, 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 a regional level is computationally more plausible than that at a link level. A time-adaptive scheme can be considered from the 3-D regions.
2. The macroscopic fundamental diagram is more likely to be well defined in a homogeneous network (*8–12*). Refining 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 subregions within a network; this approach is essential for a multireservoir modeling approach such as the macroscopic fundamental diagram modeling framework (*10, 14*). This macroscopic scale model facilitates the development of traffic management strategies.
3. 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 3-D regions; that is, the travel time objective function $f(t, x)$ is refined by the space–time properties of the 3-D regions. An investigation can be made to identify the best time to start a journey by minimizing the total route time through a spatiotemporal network.

To this end, Ji and Geroliminis investigate the performance of *k*-means in partitioning urban networks by considering spatial locations of the road as new features in the data (*1*). Saeedmanesh and Geroliminis (*2*) elaborate a second method based on the definition of a similarity matrix between observations and the application of the normalized cut (Ncut) algorithm (*3*). Such regions are defined in two dimensions; that is, 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 two-dimensional (2-D) clusters (a quasi-stationary approximation). The intercluster 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 are often flawed and incomplete, especially data collected by classic urban measurements. Therefore, data preparation is needed to create a validated data set for partitioning. The aim of the data preparation is to (*a*) remove travel time outliers, (*b*) coarsen the large-scale network to improve the computation time, and (*c*) estimate speed, including an extension for missing data. In this study, methodologies are introduced that address these issues. Real data gathered from

C. Lopez, L. Leclercq, and N. Chiabaut, University of Lyon, ENTPE, IFSTTAR, LICIT, UMR_T 9401, F-69518, Lyon, France. P. Krishnakumari and H. van Lint, Delft University of Technology, Stevinweg 1, 2600 GA Delft, Netherlands. Corresponding author: C. Lopez, clelia.lopez@entpe.fr.

Transportation Research Record: Journal of the Transportation Research Board, No. 2623, 2017, pp. 98–107.
http://dx.doi.org/10.3141/2623-11

the Amsterdam, Netherlands, urban area were used. 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 are available for every time period. This deficiency necessitates the development of algorithms to impute these missing data. In the next section the estimation of missing data and how to minimize the problem of missing data are discussed. Based on a complete (albeit partially imputed) record of all data, the Amsterdam network can now be used to assess the performance of different partitioning techniques.

Two classes of clustering techniques are investigated: an extension of Ncut based on a new expression of the similarity matrix, called "snakes" (*1, 2*), and classical methods such as density-based spatial clustering of applications with noise (DBSCAN) and growing neural gas (GNG) from the data-mining field. These clustering techniques are used to construct 3-D clusters of road links based on their average speed. It is necessary to ensure that all clusters contain a single connected component (CC) in order for all links to be reached within the same clusters. This requirement is central for applications like travel time estimation or macroscopic traffic modeling. Clustering consists of finding the optimal decomposition of the graph into a CC with the lowest internal variance of the speed while retaining a reasonable intercluster dissimilarity. However, the classical methods that are used here—DBSCAN and GNG—produce unconnected clusters. Hence, a new posttreatment method is introduced in this work to generate CCs from unconnected clusters in two and three dimensions.

The methodology for building both a validated network and a speed data set for partitioning is described next. Then the construction of the 3-D CCs using the three clustering techniques along with the posttreatment and cross-evaluation criteria is described.

DATA PREPARATION

Network Reconstruction

In the current study, raw data are the individual travel times from the city of Amsterdam. The data include the ID and location of the start and end cameras, recorded individual passing times, and travel times for 41 days. There is 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 data set is to create a route table. For this purpose, the camera needs to be located on the Amsterdam GIS network obtained from OpenStreetMaps, which consists of 147,059 links and edges as shown in Figure 1*a*. Mapping the camera coordinates to the GIS network is required since these coordinates may not be located in the road, and hence location correction (snapping) is needed. This snapping is done by determining the point-to-segment distance to find the nearest location in the network. Once the coordinates have been snapped to the network graph, the path between the start and end camera locations is found. Various path-finding algorithms are available in the literature. For this work, the shortest path was found by using the algorithm employed in the OpenStreet Maps Routing Machine, which is based on contraction rules (*15*). This algorithm is fast and robust for real-time applications. There were 314 pairs of start and end cameras leading to 314 shortest paths. Mapping these paths to the Amsterdam GIS network provides a network with 7,512 links as shown in Figure 1*b*.

Network Coarsening

After the shortest path to the Amsterdam GIS network is mapped, there are still 7,512 links, quite a large amount. Hence, network coarsening techniques were employed 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 the nodes that have similar matching criteria. The matching criteria can differ according to the application. In this study, the matching criterion relates to the differences in edge weights. Here these weights represent the speed of each link or edge. If the edges have the same weight, the node that connects the edges will be collapsed or removed. The network coarsening here 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; that is, a node with a lower rank is removed before a node with a higher rank; (*b*) the contraction rules are determined on the basis of 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 weight for each link or edge is the estimated speed of each link. The weight w_{uv} of link (u, v) is the speed of the link s_{uv}. Since only speed is required for one time slice to assign the weights, a peak period time slice (4:00 p.m.) was chosen because the network will exhibit the most variance during peak periods with most links having different speeds at this time. If a non–peak period is chosen, these variances will be smoothed out. A more detailed description of node ranking and the contraction rules are given next.

Node Ranking

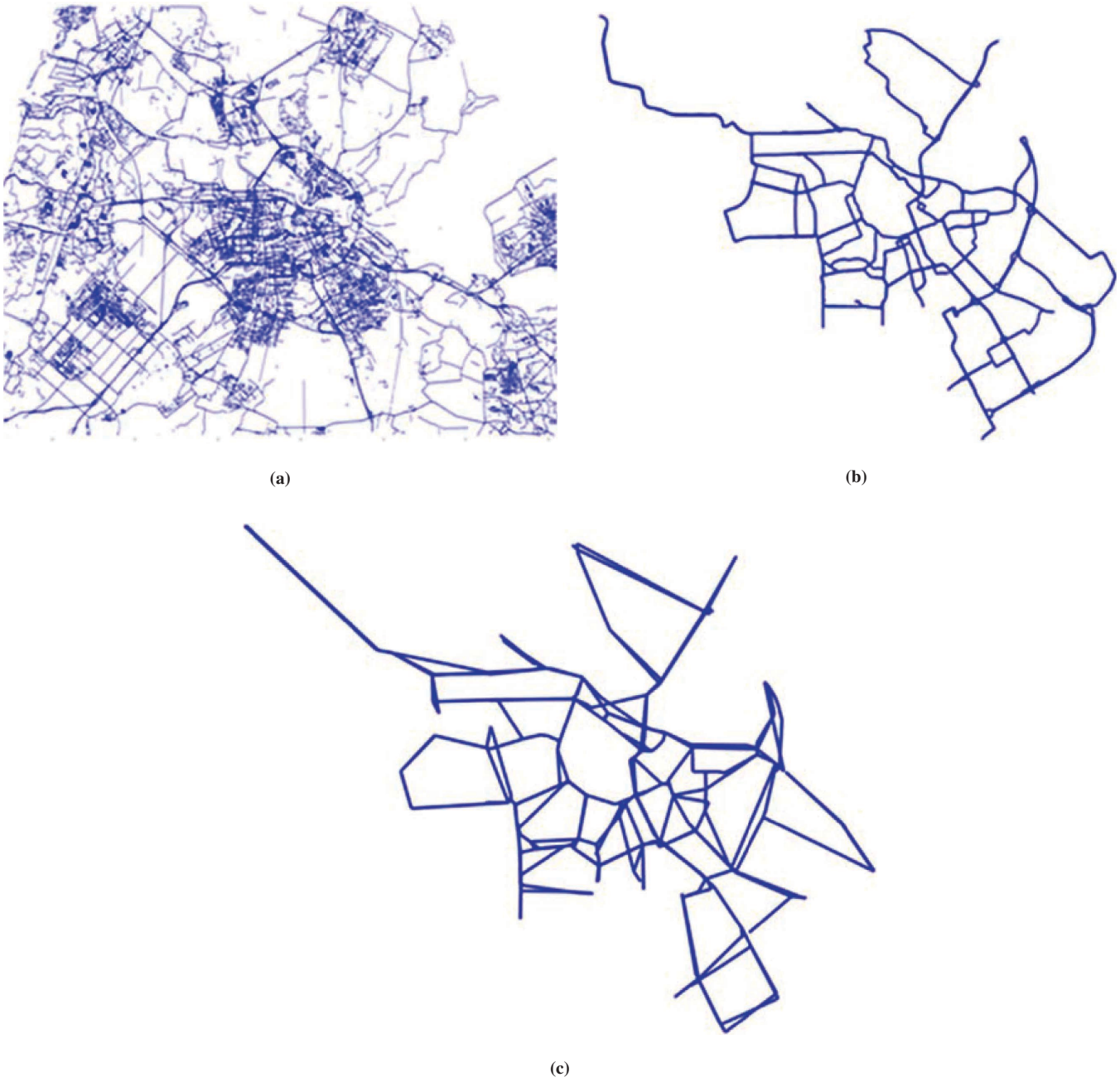
The order in which the nodes are removed is important for graph coarsening. There are different node ranking or ordering techniques as introduced in relation to contraction hierarchies (*15*). In this work, the nodes are ordered on the basis of their densities. The more neighbors the node has, the higher 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 applications. Given a graph $G = (V, E)$ with node set V and an edge set E, suppose that $(u, v) \in E$, where $u, v \in V$; then the rank of the nodes u and v will satisfy the following condition:

$$r(u) > r(v) \qquad \text{if } n(u) > n(v) \tag{1}$$

where $n(u)$ and $n(v)$ are the number of neighbors of nodes 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 with regard to 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 on the basis of their ranks in ascending order with the lowest rank first.

Contraction Rules

Once the nodes are sorted according to rank, the contraction rules are used to remove them. First, a criterion has to be set to decide if



(a) (b) (c)

FIGURE 1 Camera data to validated network: (*a*) Amsterdam GIS network with 147,059 links, (*b*) shortest paths mapped to network with 7,512 links, and (*c*) coarsened network with 411 links.

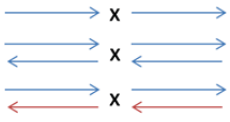
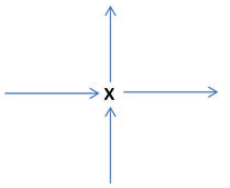
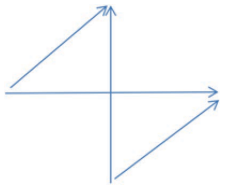
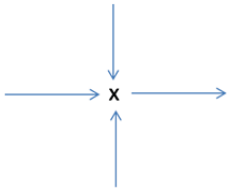
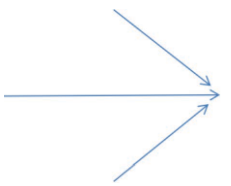
the given node is eligible for contraction to decrease the complexity. The criterion is that if the node removal results in the same or an increased number of links than before removal, the node is not removed. Table 1 presents an overview of different cases of node contraction. The link color represents weights; different colors imply different weights. For example, in Case 5, removing the nodes results in four new links, the same number of links as before node contraction. Hence, this node will not be considered for removal. It is also observed that in the majority of cases, contracting nodes with more than four neighbors leads to five or more links. Therefore, an initial constraint is imposed on the node contraction to only contract nodes with neighbors less than or equal to four links as given below:

$$c_1(v) = \begin{cases} 1 & \text{if } n(v) \le 4 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

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, the edge difference is then used as the next rule for inclusion or exclusion of that node for contraction:

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

TABLE 1 Examples of Node Contraction Rules

Case	Some Cases of Node Removal	Results of Node Removal
1		
2		
3		
4		na
5		
6		

NOTE: na = not applicable.

where 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. The threshold was set to zero so that nodes can be contracted only if the links have the same speed.

On the basis of these two criteria, the rule sets for node contraction are now discussed. 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, all the incoming and outgoing links of node v are found. If $c_1(v)$ returns 0, go to Step 9.
4. Pair 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) is the outgoing link, the pairs are (u, v, w), (x, v, w), and (y, v, w).
5. A U-turn is not considered in this work, and thus (u, v, u) is not considered as a valid pairing, as shown in Case 2 of Table 1.
6. Check if the c_2 criterion is satisfied for all the valid incoming–outgoing pairs.
7. If one of the pairs 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, the node is removed from the node list and the node ranking is updated as new links are formed. Repeat from Step 1.
9. Stop the iteration.

In the Case 4 of Table 1, the node cannot be contracted since the edge difference did not satisfy the threshold criterion. The node coarsening is applied to the mapped Amsterdam network with 7,512 links. As explained before, the peak period was estimated to be around 4:00 p.m. 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*.

Speed Estimation

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

The moving average process was used to remove travel time outliers. There are usually alternative paths for a given origin–destination (O-D) pair. They can be k-shortest paths, representing the common route choices. Travel times significantly higher than the distribution were defined as outliers. To remove them, two approaches were proposed: the first was to 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 the distribution is sensitive to the number of observations. Thus, outliers can be smoothed into time periods with few travel times. The second approach for removing the outliers is based on the moving average process. The moving average $\bar{\tau}$ is defined as

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

where τ_n is the nth realized travel time. Outliers are defined by $\bar{\tau}_n + \Delta\tau$, where $\Delta\tau$ is the travel time window. In this study, $\Delta\tau$ was set to the standard deviation of the peak demand; $\Delta\tau$ was tuned by a graphical inspection of the effects of $\Delta\tau$ on the number of outliers. Only the upstream window was considered to remove travel time outliers since the downstream window is not relevant for the current study; that is, fast travel times could not be considered outliers. The travel time window is refined after two iterations. Figure 2 gives an example with the x-axis being the observed time and the 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 were aggregated by period. The mean of individual travel times was considered for a given O-D at a given period.

An O-D denoted by od_i is characterized by a succession of links $L_i = (l_1, l_2, \ldots, l_n)$; that is, it represents a path. The first shortest path was 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

$$s_i = \frac{\text{dist}(L_i)}{tt_{i,t}}$$

It can occur that a given link l_k belongs to different O-D paths, that is, $l_k \in (od_1, od_2, \ldots, od_m)$. For this case, its speed is defined as the mean of the corresponding speeds of the paths.

When no data are available to measure $tt_{(i,t)}$, three approaches were distinguished to estimate the speed. The first approach sets links without data to the limited speed, which does not consider traffic lights. It was assumed that setting limited speed at links without data would increase the variance of the zones. The second approach computes a new speed value through the speeds of a specific neighborhood. It was assumed that the average speed of a given specific neighborhood would 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 a link belonging to the boundaries of the zones; that is, the link without data is being connected with links from different zones.

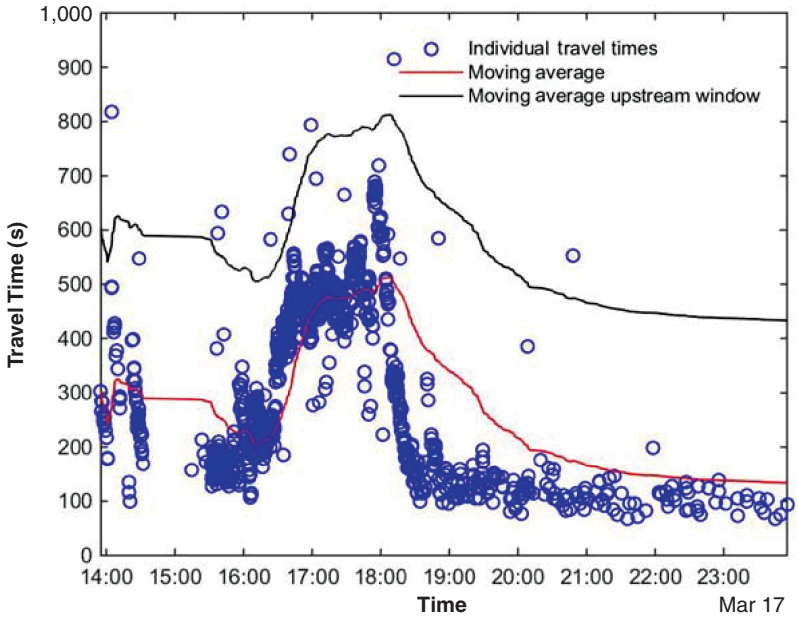


FIGURE 2 Travel times exceeding travel time window (black curve), which are considered travel time outliers.

The third approach duplicates the speed from an identified link. It was assumed that the duplication of speed minimizes the speed variance of zones. The third approach used 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 path matrix. A represents the relationship of the finite graph G. In this study, G is the graph of the 3-D network, that is, a repetition of the same network at N_t time slices. Through 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, the edge weight is denoted 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, the edge weight is

$$c_{ij} = \frac{\text{length}(l_j)}{s_j}$$

else if vertex v_j is upstream to v_i, the weight cost is

$$c_{ij} = \frac{\text{length}(l_j)}{w}$$

where w is the backward wave speed, which is set to 5 m/s in this study. If no data are available for v_j, which is downstream to v_i, then

$$s_j = \max(s) + \frac{\max(s)}{2}$$

Dijkstra's algorithm is used to estimate the shortest path between two vertices i and j in G (*16*). The cost of the shortest path is denoted D_{ij}. Thus, the link without data will be assigned the speed of the most relevant link, which is identified as the link that minimizes the cost function. For computational efficiency, the process is performed strictly for links without data and a time window is used to constrain the search for the relevant link.

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 study require a strongly connected graph; that is, 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 resolution networks, this constraint is not true. Thus, direction is not a convenient attribute to partition the Amsterdam network.

This study application focuses on only 1 day of data. This given day is a common weekday. The analyzed period is from 7:00 a.m. to 5:00 p.m. It is a time window of 8 h, one-third of a day representing the morning peak demand. After the travel time data were analyzed, it was found that not all of the O-Ds are used on all days. Only used O-Ds were 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 earlier. From the O-Ds used, around 16% of the data is missing.

SPATIOTEMPORAL PARTITIONING TECHNIQUES

Normalized Cut Based on Snake Similarities

One of the contributions of this study is to adapt the existing methodology of snakes for a spatiotemporal network that is a repetition of the same network at numerous time slices. A snake is composed of a sequence of links, which iteratively grows by adding adjacent

links that are similar to itself (*2*). 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. This neighborhood was considered in both space and time. The link l_i at time t is denoted by $l_{(i,t)}$. Duplicates of link $l_{(i,t)}$ were made at instants $t-1$ and $t+1$ and denoted $l_{(i,t+1)}$ and $l_{(i,t-1)}$, respectively. The snake similarities are defined as follows (*2*):

$$w_{ij} = \sum_{k=1}^{N} p^k \,\text{intersect}\left(S_{ik}, S_{jk}\right) \tag{5}$$

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

One of the main inconveniences of the 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 spatiotemporal 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 searches 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, that is, a network in which every node has a link with every other node. Both these examples correspond to the best and the worst cases. Their associated complexities are, respectively, $O(n^2)$ and $O(n^3)$. Any realistic situation is between these boundaries. For example, the complexity of the algorithm in this case study was numerically investigated. It was shown that it is $O(n^{2.2})$, where 2.2 was obtained by a regression of the complexity on the snake's length n. 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, too short a snake cannot discover the entire topology of the network in both space and time. The short snake length may also provide cluster results in which a cluster contains links that are not all connected with each other. Therefore, the length was set to 38% of the spatiotemporal size of the network. In addition, the weight coefficient was set to $p = 0.8$.

Partitioning Based on Data Point Clustering

The two other classic clustering methods that are considered for this study are GNG (*17*) and DBSCAN (*18*). The 3-D network was represented in a data set containing four variables—link coordinates with their corresponding speed and time measurement (x, y, t, s). These four quantitative variables were normalized. After normalization, the speed column was multiplied by a fixed coefficient equal to 3 to be sure that speed is the predominant variable over spatial and temporal coordinates during clustering.

1. DBSCAN is a density-based method. It 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 the current study, the parameters were set to $\varepsilon = 0.01$ and MinPts = 50.
2. 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 was 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 iterations m, L; the adaptation threshold ε_b, ε_n; the neighborhood size α, δ; and the time T, which were set as $N = 10$, $m = 20$, $L = 50$, $\varepsilon_b = 0.2$, $\varepsilon_n = 0.005$, $\alpha = 0.5$, $\delta = 0.995$, and $T = 50$.

The clustering results provide clusters that are not connected, as shown in Figure 3*a*. The homogeneous zone partition needs to be a single connected cluster. Therefore, posttreatment is needed on the clusters; that treatment is obtained from DBSCAN and GNG to obtained CCs with minimum intercluster speed variance. The three steps for the posttreatment algorithm are

- Identifying the CCs in each of the clusters,
- Assigning the biggest CCs as the initial clusters, and
- 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 arrangement, 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 two CCs in the blue cluster, one CC in red, one CC in green, and so on. It can be clearly seen that there are two CCs for the blue cluster.

Assuming that there is 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 clusters. The rest of the $(X-M)$ CCs are merged with the M initial cluster and the following two parameters are found for each pair:

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

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

where $x \in (X-M)$ CCs, $m \in M$ initial CCs, and 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 CC with the largest $\sum c(x, m)$ is selected, and it is merged with the initial cluster m, which has $c(x, m) = 1$ and 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 $v(x, m)$ are 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 cluster.

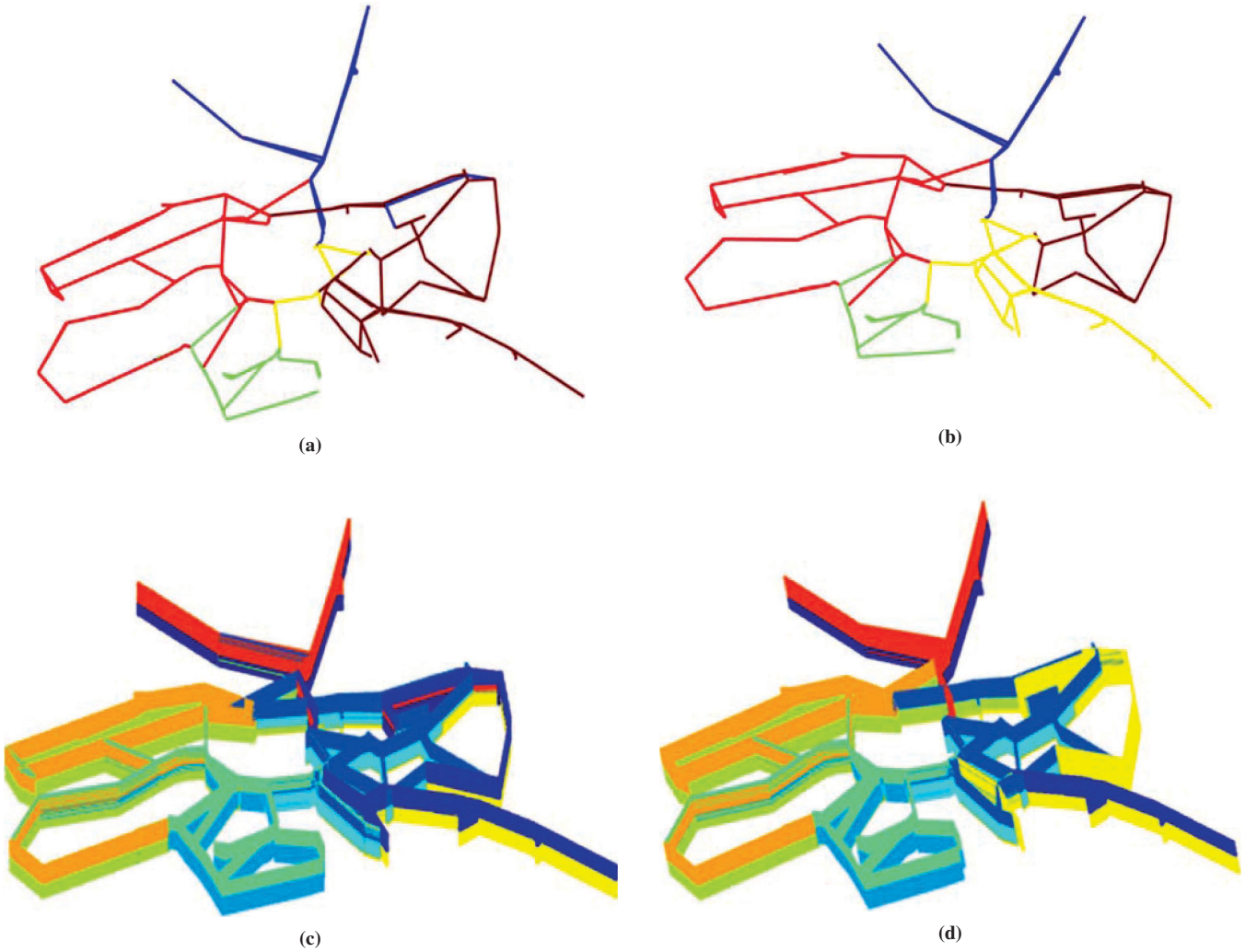


FIGURE 3 Posttreatment results for data point clustering methods: (*a*) unconnected 2-D clusters before posttreatment, (*b*) connected 2-D clusters after posttreatment, (*c*) unconnected 3-D clusters before posttreatment, and (*d*) connected 3-D clusters after posttreatment.

Figure 3 shows an example of posttreatment results in two and three dimensions. Figure 3*a* shows the cluster before posttreatment and Figure 3*b* shows posttreatment with the same number of clusters as the input for a single time slice. Figure 3*d* shows the result of the posttreatment in three dimensions with same number of clusters as the input shown in Figure 3*c*. There is no difference in the methodology between the 2-D and 3-D posttreatments. The only difference is in calculating the 3-D adjacency matrix for finding the 3-D CCs and the connectivity. The 3-D adjacency matrix is defined by creating bidirectional links between the time slices.

Evaluation Metrics

The three clustering techniques were compared by using three indicators in this study: total variance normalized (TV*n*), connected clusters dissimilarity (CCD), and time computation. TV is the original indicator of the snake similarities partitioning and is defined as $\mathrm{TV} = \sum_{A \in C} N_A * \mathrm{Var}(A)$ (*1*). TV was normalized by using the following equation:

$$\mathrm{TV}n = \frac{1}{N} \frac{\sum_{A \in C} N_A * \mathrm{Var}(A)}{S^2} \tag{8}$$

This indicator is based on the assumption that a given cluster is composed of links characterized by similar speeds. The speed variance is highlighted.

The second metric used is the CCD. The criterion is the dissimilarity between a given cluster and its neighboring cluster, that is, clusters touching the given cluster. CCD is defined as follows:

$$\mathrm{CCD} = \frac{\sum_{i=1}^{n} \sum_{k=1+i}^{n} \delta_{ik} |\bar{x}_i - \bar{x}_k|}{\sum_{i=1}^{n} \sum_{k=1+i}^{n} \delta_{ik}} \tag{9}$$

$$\delta_{ik} \begin{cases} 1 & \text{if } k \text{ and } i \text{ are connected clusters} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

The time computation indicator evaluates the computational cost of the algorithms. The complexity has to be considered for another size network and number of time slices. Two different fields of partitioning methods were compared: Ncut from the graph theory based on the snake similarities and DBSCAN and GNG from data point clustering. The basic data point clustering methods are faster but a posttreatment process is required. The computational cost of the posttreatment 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.

RESULTS AND DISCUSSION

In this section, results from the three methods are analyzed and compared: Ncut based on snake similarities and DBSCAN and GNG with posttreatment. The comparison focuses on two conceptually different fields to partition a transportation network. Figure 4 illustrates the partitioning results from the three methods under a fixed number of clusters set to 2. Neither of the data point clustering methods produces the same spatiotemporal zones by the posttreatment algorithm. The zone shapes present a reasonable 3-D cover; that is, a given zone is roughly compacted by space and time. Three indicators were used to evaluate the three methods. TV*n* and CCD measure the quality of clusters representing the homogeneous zones and the intercluster dissimilarity, respectively. The time computation quantifies the computational cost applied to this case study. Figure 4, *d* and *e*, shows the TV*n* and CCD for a systematic number of clusters from two to nine. It can be observed that GNG is the best method to minimize TV*n*. Ncut is the best method to maximize the CCD. However, the time computation for GNG is found to be the fastest.

Figure 5, *a*, *c*, *e,* illustrates the GNG partitioning with different numbers of clusters ranging from two to four. Figure 5, *b*, *d*, *f*, shows histogram plots of link speed for each cluster. The maximal speed is around 40 m/s, corresponding to highway speed. The validated Amsterdam network used here contains both provincial roads and highways. The histogram validates that each cluster has a speed variance as low as possible. For example, in Figure 5*d*, most of the links in the blue cluster have a speed of 5 to 10 m/s. Only a few of the links in the blue cluster have speeds of 0 to 5 m/s compared with the orange cluster. This observation remains valid for all three cases shown here and proves the hypothesis that the cluster results from posttreatment provide clusters that are connected and that minimize the speed variance.

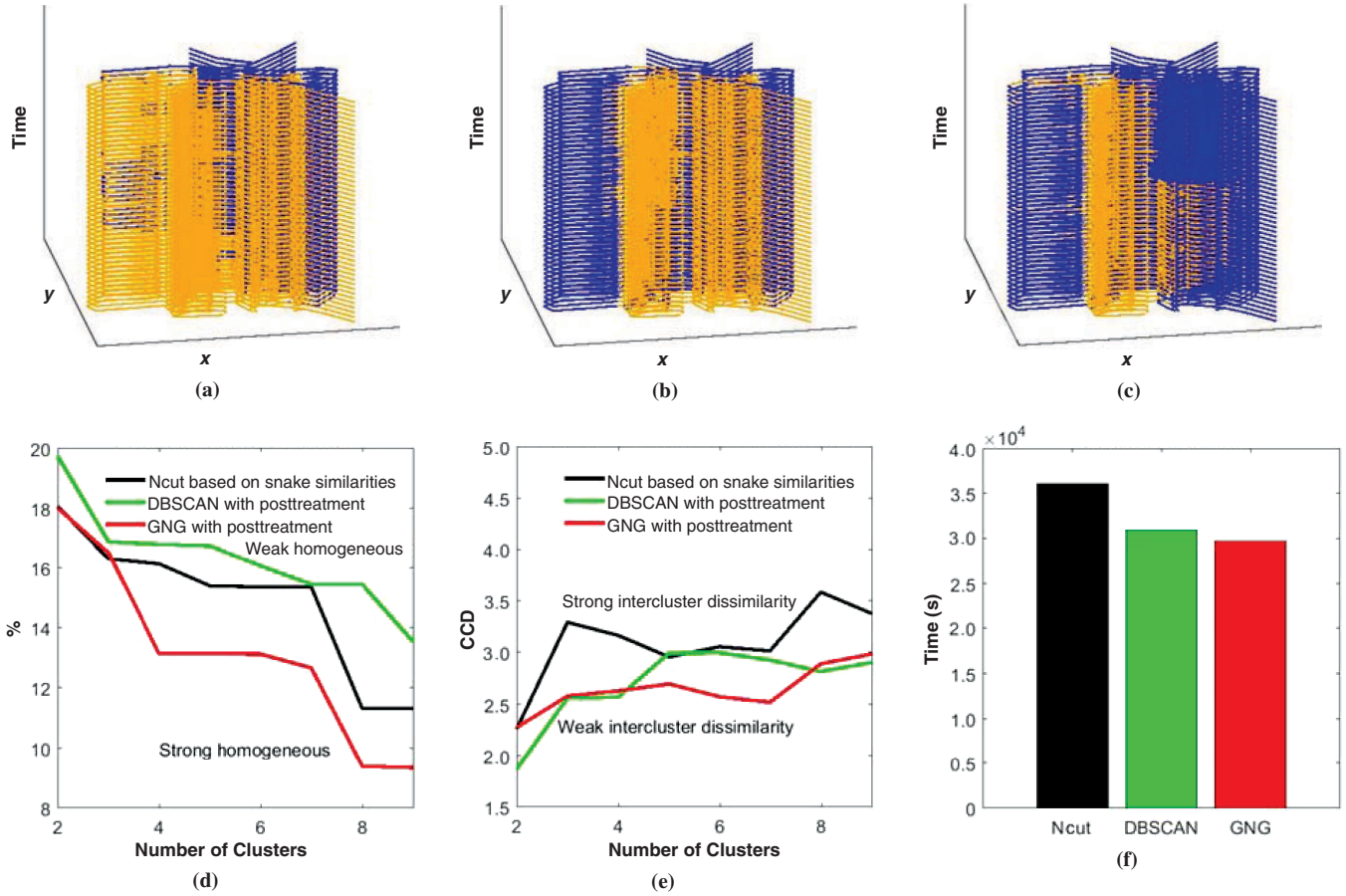


FIGURE 4 Amsterdam network (7:00 a.m. to 5:00 p.m.) with 3-D clusters (n = 2) obtained with (*a*) Ncut based on snake similarities, (*b*) DBSCAN with posttreatment, and (*c*) GNG with posttreatment; and comparison of three partitioning methods by (*d*) TV*n*, (*e*) CCD, and (*f*) computation time.

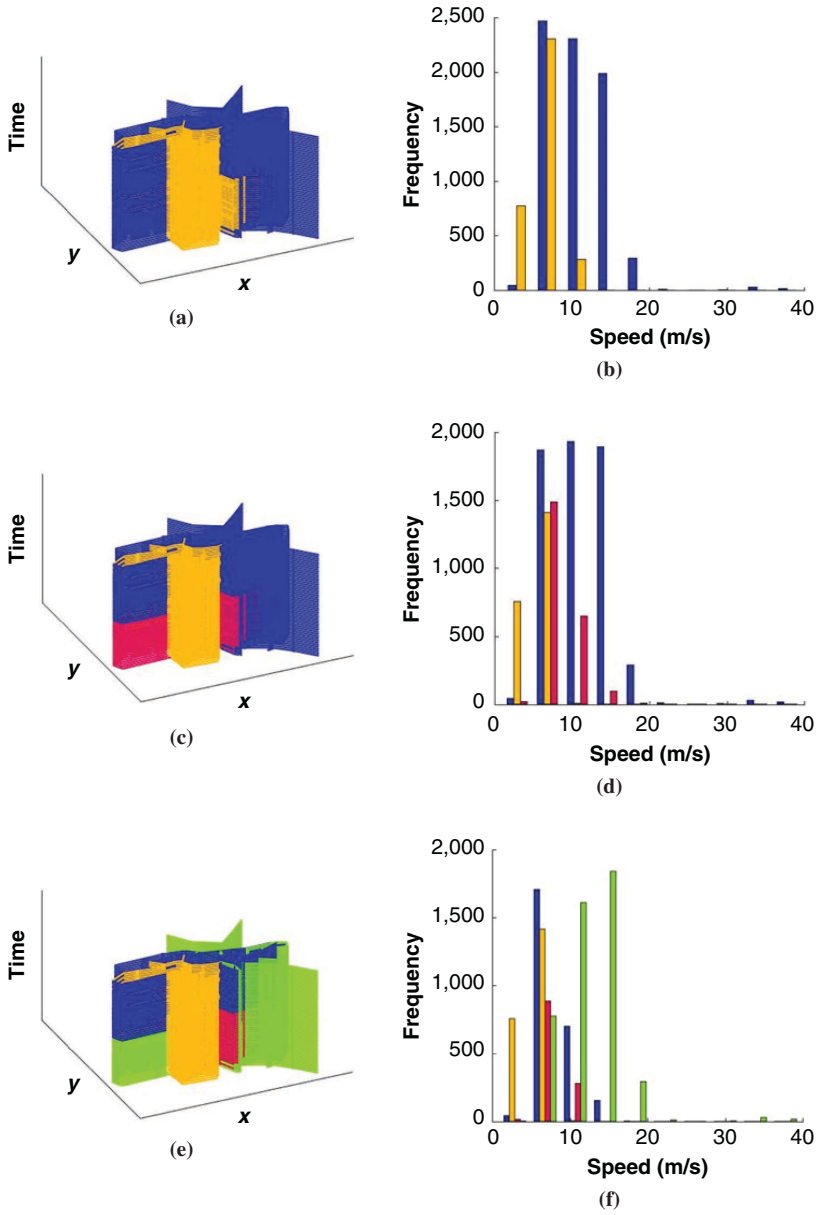


FIGURE 5 (*a*, *c*, *e*) 3-D network visualization of GNG partitioning into two to four spatiotemporal zones and (*b*, *d*, *f*) corresponding histograms.

CONCLUSION AND FUTURE WORK

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

Two concepts of spatiotemporal partitioning of a transportation network were compared in this work. Methods belonging to different fields were implemented: unsupervised learning and graph theory. The clustering approach considers links under a network as data points. This hypothesis allows the implementation of simple and efficient clustering algorithms, but it requires postprocessing for contiguity; that is, all clusters should be composed of a unique CC. In the second method, a transportation network at numerous time slices was considered as a graph. In this case, connectivity is considered through graph topology.

Three methodologies were presented to partition a network in both space and time; they were demonstrated in a coarsened Amsterdam city network. The partition criterion is the speed. The comparison between the three techniques was evaluated by two metrics: TV*n* and CCD. Preliminary results show that none of the partitioning methods is better with regard to both metrics. TV*n* measures the homogeneous zones, which can be spread throughout the 3-D network keeping the connectivity. The CCD indicator focuses on the intercluster dissimilarity but can forsake the homogeneity. The

choice of a spatiotemporal partitioning method is a compromise between both criteria.

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, only one day was considered in this work and the methodology can be iterated for several days. Another application that is promising for these spatiotemporal 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 thank Sjoerd Linders and the Amsterdam Institute for Advanced Metropolitan Solutions for providing the data for this study. This research was supported by the region Auvergne-Rhône-Alpes (ARC7 Research Program), by a Bourse Eole from the Réseau Franco–Néerlandais, and by the European Research Council under the European Union's Horizon 2020 research and innovation program. The authors also acknowledge the support of the SETA project funded by the European Union's Horizon 2020 program. The authors thank Zhengui Xue for editorial help with this paper.

REFERENCES

1. Ji, Y., and N. Geroliminis. On the Spatial Partitioning of Urban Transportation Networks. *Transportation Research Part B: Methodological,* Vol. 46, No. 10, 2012, pp. 1639–1656. https://doi.org/10.1016/j.trb.2012.08.005.
2. Saeedmanesh, M., and N. Geroliminis. Clustering of Heterogeneous Networks with Directional Flows Based on "Snake" Similarities. *Transportation Research Part B: Methodological,* Vol. 91, 2016, pp. 250–269. https://doi.org/10.1016/j.trb.2016.05.008.
3. Shi, J., and J. Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 22, No. 8, 2000, pp. 888–905. https://doi.org/10.1109/34.868688.
4. Haddad, J., M. Ramezani, and N. Geroliminis. Cooperative Traffic Control of a Mixed Network with Two Urban Regions and a Freeway. *Transportation Research Part B: Methodological,* Vol. 54, 2013, pp. 17–36. https://doi.org/10.1016/j.trb.2013.03.007.
5. Ramezani, M., J. Haddad, and N. Geroliminis. Dynamics of Heterogeneity in Urban Networks: Aggregated Traffic Modelling and Hierarchical Control. *Transportation Research Part B: Methodological,* Vol. 74, 2015, pp. 1–19. https://doi.org/10.1016/j.trb.2014.12.010.
6. Ma, Y.-Y., Y.-C. Chiu, and X.-G. Yang. Urban Traffic Signal Control Network Automatic Partitioning Using Laplacian Eigenvectors. In *12th International Conference on Intelligent Transportation Systems,* IEEE, New York, 2009, pp. 1–5. https://doi.org/10.1109/ITSC.2009.5309560.
7. Peeta, S., and A. K. Ziliaskopoulos. Foundations of Dynamic Traffic Assignment: The Past, the Present and the Future. *Networks and Spatial Economics,* Vol. 1, No. 3/4, 2001, pp. 233–265. https://doi.org/10.1023/A:1012827724856.
8. Godfrey, J. W. The Mechanism of a Road Network. *Traffic Engineering and Control,* Vol. 11, 1969, pp. 323–327.
9. Mahmassani, H. S., J. C. Williams, and R. Herman. Performance of Urban Traffic Networks. In *Proceedings of the 10th International Symposium on Transportation and Traffic Theory* (N. H. Gartner and N. H. M. Wilson, eds.), Elsevier, New York, 1987, pp. 1–20.
10. Geroliminis, N., and C. F. Daganzo. Existence of Urban-Scale Macroscopic Fundamental Diagrams: Some Experimental Findings. *Transportation Research Part B: Methodological,* Vol. 42, No. 9, 2008, pp. 759–770.
11. Buisson, C., and C. Ladier. Exploring the Impact of Homogeneity of Traffic Measurements on the Existence of Macroscopic Fundamental Diagrams. *Transportation Research Record: Journal of the Transportation Research Board,* No. 2124, 2009, pp. 127–136. https://doi.org/10.3141/2124-12.
12. Leclercq, L., N. Chiabaut, and B. Trinquier. Macroscopic Fundamental Diagrams: A Cross-Comparison of Estimation Methods. *Transportation Research Part B: Methodological,* Vol. 62, 2014, pp. 1–12. https://doi.org/10.1016/j.trb.2014.01.007.
13. Yildirimoglu, M., M. Ramezani, and N. Geroliminis. Equilibrium Analysis and Route Guidance in Large-Scale Networks with MFD Dynamics. *Transportation Research Part C: Emerging Technologies,* Vol. 59, 2015, pp. 404–420. https://doi.org/10.1016/j.trc.2015.05.009.
14. Daganzo, C. F. Urban Gridlock: Macroscopic Modeling and Mitigation Approaches. *Transportation Research Part B: Methodological,* Vol. 41, No. 1, 2007, pp. 49–62. https://doi.org/10.1016/j.trb.2006.03.001.
15. Geisberger, R., P. Sanders, D. Schultes, and D. Delling. Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. In *International Workshop on Experimental and Efficient Algorithms,* Springer, Berlin, 2008, pp. 319–333. https://doi.org/10.1007/978-3-540-68552-4_24.
16. Dijkstra, E. W. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik,* Vol. 1, No. 1, 1959, pp. 269–271. https://doi.org/10.1007/BF01386390.
17. Fritzke, B. A Growing Neural Gas Network Learns Topologies. *Advances in Neural Information Processing Systems,* Vol. 7, 1995, pp. 625–632.
18. Ester, M., H. P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining,* AAAI Press, Menlo Park, Ca., 1996, pp. 226–231.
19. Martinetz, T., and K. Schulten. A "Neural Gas" Network Learns Topologies. In *Artificial Neural Networks,* North-Holland, Amsterdam, Netherlands, 1991, pp. 397–402.

The Standing Committee on Traffic Flow Theory and Characteristics peer-reviewed this paper.