Australasian Transport Research Forum 2016 Proceedings
16 – 18 November 2016, Melbourne, Australia
Publication website: http://www.atrf.info

Graph-based Analysis of City-wide Traffic Dynamics using Time-evolving Graphs of Trajectory Data

Jiwon Kim[1], Kai Zheng[2], Sanghyung Ahn[1], Marty Papamanolis[1], Pingfu Chao[2]

[1] School of Civil Engineering, The University of Queensland, Brisbane, Australia

[2] School of Information Technology and Electrical Engineering, The University of Queensland Brisbane, Australia

Email for correspondence: jiwon.kim@uq.edu.au

Abstract

This paper proposes a graph-based approach to representing spatio-temporal trajectory data that allows an effective visualization and characterization of city-wide traffic dynamics. With the advance of sensor, mobile, and Internet of Things (IoT) technologies, vehicle and passenger trajectories are increasingly being collected in massive scale and are becoming a critical source of insight into traffic pattern and traveller behaviour. To leverage such trajectory data to better understand traffic dynamics in a large-scale urban network, this study develops a trajectory-based network traffic analysis method that converts individual trajectory data into a sequence of graphs that evolve over time (known as dynamic graphs or time-evolving graphs) and analyses network-wide traffic patterns in terms of a compact and informative graph-representation of aggregated traffic flows. First, we partition the entire network into a set of cells based on the spatial distribution of data points in individual trajectories, where the cells represent spatial regions between which aggregated traffic flows can be measured. Next, dynamic flows of moving objects are represented as a time-evolving graph, where regions are graph vertices and flows between them are treated as weighted directed edges. Given a fixed set of vertices, edges can be inserted or removed at every time step depending on the presence of traffic flows between two regions at a given time window. Once a dynamic graph is built, we apply graph mining algorithms to detect change-points in time, which represent time points where the graph exhibits significant changes in its overall structure and, thus, correspond to change-points in city-wide mobility pattern throughout the day (e.g., global transition points between peak and off-peak periods).

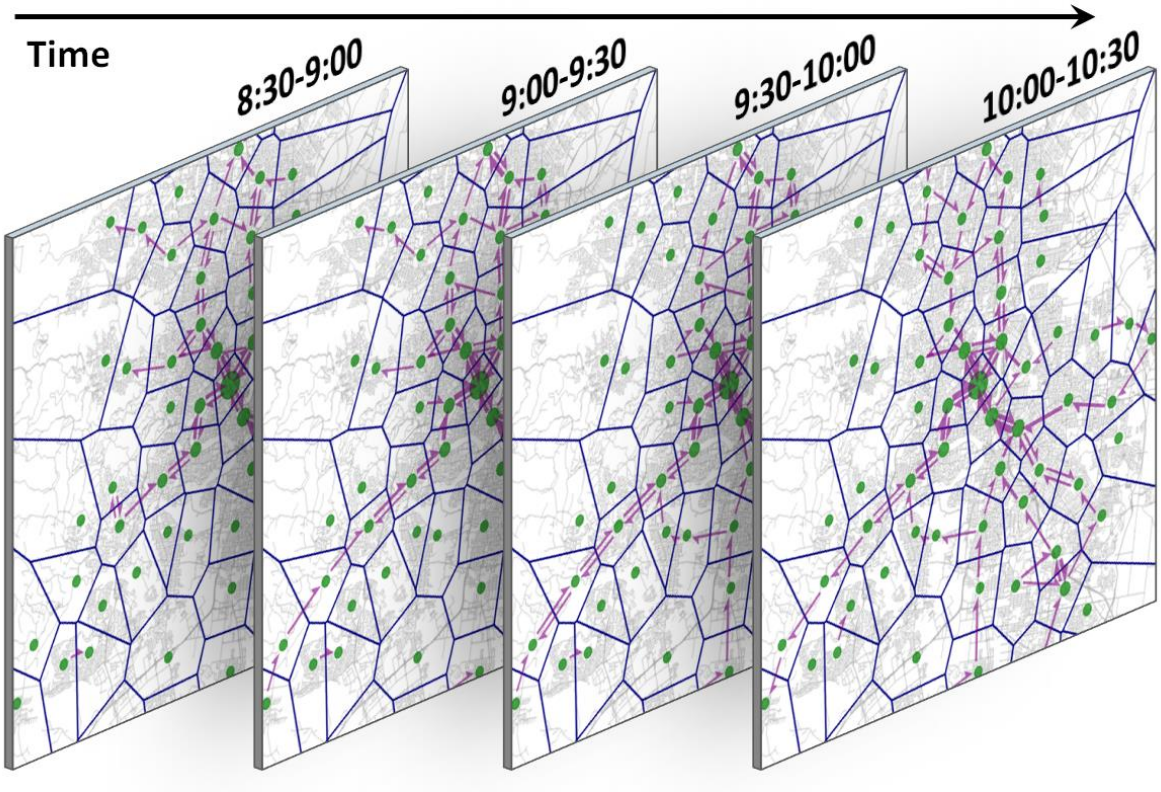
1. Introduction

A graph-based representation provides a powerful means of describing complex relationships among entities in a dataset in a simple and intuitive way, and studying the properties of these graphs provides useful information about the internal structure of the dataset. In many real-world systems, relationships between entities evolve over time and such time-varying relationships or connections can be represented as a sequence or time-series of graphs, namely a dynamic graph or time-evolving graph. Traffic flows in a road network can be naturally represented as a dynamic graph, where different parts of the network are represented as graph vertices and traffic flows between them are represented as directed edges with time-varying weights. Unlike the traditional road network graph, where vertices and edges represent intersections and links, respectively, and network connections represent static routes, this “flow graph” or “flow map” connects geographically distributed areas through dynamic flows of vehicles or people and is constantly undergoing changes to its network connection structure (von Landesberger et al., 2016). For instance, given a fixed set of vertices, which represent a pre-defined set of regions of interest, edges

can be inserted or removed at every time step depending on the presence of traffic flows between two regions at a given time window. The recent availability of area-wide trajectory data such as GPS vehicle trajectories or smart card passenger trajectories has enabled the construction of such dynamic flow graphs (Kim and Mahmassani, 2015).

This study proposes a graph-based analysis framework that converts individual trajectory data into a compact and informative representation of dynamic graphs and explores new ways of measuring, visualizing, and characterizing network-wide traffic dynamics and mobility patterns. A particular application addressed in this paper is to create a robust profile of daily network traffic patterns using a graph summarization method applied to a *graph series* or a sequence of aggregated flow graphs throughout the day. Figure 1 illustrates an example of daily graph series that is constructed from bus passenger trajectories obtained in Brisbane, Australia. The entire day is divided into a sequence of time slices with a fixed time window length (30 minutes in this example) and a graph of aggregated flows is constructed for each time slice, which provides a snapshot of network-wide traffic flow pattern or spatial distribution of flows for that particular time window. These snapshots are then connected in a sequence in the order of time, providing the information on the temporal evolution of the spatial traffic distribution patterns. Once we obtain such a graph series of daily traffic, we can apply a change point detection technique to identify time points where the graph structure changes significantly compared to their previous time slices and segment the daily graph series into a set of graph segments that can be used to characterise and summarise overall traffic patterns in different time-of-day periods.

Figure 1: Illustration of a trajectory flow map, a dynamic graph of aggregated traffic flows constructed from trajectory data. The presented example is based on bus passenger trajectories obtained in Brisbane, Australia.



2. Methodology

In this section, we describe detailed methodology and techniques for a framework that aims to characterise spatial and temporal patterns of network-wide traffic flows using a *trajectory flow graph*. The term "trajectory flow graph" is used to represent a static graph of aggregated flows at a particular time slice, while emphasizing the fact that the graph is built from trajectory data. Table 1 provides a summary of notations used in this paper. The proposed framework consists of three stages: (i) network partitioning, (ii) graph generation, and (iii) traffic profiling. Detailed procedures for each stage are discussed below.

Table 1: Summary of notations

Notation	Definition
p	Data point in a trajectory, represented as 3-tuple (x, y, τ)
P	Point set that combines all data points in trajectories, $p \in P$
s	Seed point that is used to partition the network into cells
S	Set of seed points, $s \in S$
c	Space cell; basic unit of an area that encloses spatially grouped data points
C	Set of cells, $c \in C$
γ	Desired cell size (radius), in km
Δt	Length of time slice, time duration between time t and t+1, in minutes
G	Trajectory flow graph series, a dynamic graph or a time-ordered sequence of trajectory flow graphs with a finite number of time steps T; $G = \{G_t\}_{t=1}^{T}$
G_t	Trajectory flow graph, a static graph of aggregated flows representing traffic flows during time slice t , $G_t = \left(V, E_t \subseteq (V \times V)\right)$
V_t	Vertex set for the graph at time slice t, corresponding to cell set C
E_t	Edge set for graph at time slice t, representing traffic flows between cells
T	Number of time slices
T^c	Set of change points
$G_{(s)}$	The sth graph segment, a grouping of temporally adjacent graphs between change points, $G = \left\{G_{(s)}\right\}_{s=1}^{\lvert T^c \rvert + 1}$

2.1. Partitioning a Network into Cells

A trajectory of moving object is a time-ordered sequence of points $p(x, y, \tau)$ that represent the x and y coordinates of the moving object at time τ. In order to convert trajectory data into a dynamic graph of aggregated flows, the first step is to define spatial regions for which aggregated traffic flow will be computed. Below we describe an approach to partitioning the entire network into cells with similar size by clustering the data points in trajectories. Alternatively, one could use suburb or postal area boundaries to define spatial regions for the traffic flow analysis. The network partitioning method adopted in this study is based on the method developed by Adrienko and Adrienko (2011), where trajectory data points are grouped in space based on a desired spatial radius. Then, each point group is represented by a cell or partition in the network, where the cell boundaries are determined by a Voronoi tessellation. Our approach is different, however, in that we perform an additional merging

process that further clusters cells into homogenous regions in terms of their underlying trajectories, which produces a more meaningful network partitioning for the purpose of our study.

Given a set of trajectories, we first extract "seed points" that will be used in constructing point groups or cells. One could use all the data points in the trajectory dataset or select only a subset of data points. Depending on the choice of seed points, the resulting network partitioning may have different implications. For instance, if we use trajectories' origin and destination points only, the point groups will tend to be cantered at major origin and destination points and, thus, the network will end up being partitioned into a set of representative origin and destination regions.

Let S be the set of pre-defined seed points and γ represents a parameter specifying the desired radius of a cell. The procedure for grouping seed points are described as follows:

Procedure 1: Grouping points into cells with a desired radius
Input: *set of seed points S, desired cell radius γ*
Output: *set of cells C*

1. *Initialize C = { }.*
2. *For each seed point s in S,*
    a. *Find the closest cell c in C, where the distance between s and c's centroid is less than or equal to γ and c's centroid is closer to s than any other cell centroids in C.*
    b. *If such c is not found, create a new cell c and assign s to c. Add c to C.*
    c. *Assign s to c and update c's centroid as the mean point of all member points in c.*
3. *For each cell c in C, remove all member points while keeping its centroid.*
4. *For each point s in S, find the closest cell c in C and redistribute s to c.*

Once all the seed points are grouped into cells and the centroid of each cell is obtained based on the mean position of the seed points assigned to the cell, the cell boundaries can be determined by building a Voronoi diagram, using those cell centroids as Voronoi generator points (Adrienko and Adrienko, 2011; Aurenhammer, 1991). A Voronoi diagram partitions a plane into convex polygons such that each polygon contains exactly one generator point and all locations in a Voronoi polygon are close to its own generator point than any other cells' generator points in the diagram. Correspondingly, the points on the boundary between two Voronoi polygons are equidistant to their two generator points.

Using the Voronoi diagram constructed from the cell centroids, we partition the network into spatial cells. Figure 2 (a)-(b) illustrate this procedure. Figure 2 (a) shows original trajectory data displayed on the region map. The trajectory data used in this example include 1000 bus passenger trajectories obtained in Brisbane, Australia. Points in trajectories represent bus stops and the origin point of each trajectory is shown as green circle in the figure. In Figure 2 (b), all the points in the original trajectories are divided into cells, where different point colours represent different point clusters. The centroids of point groups are shown as black circles which are then used as generator points for constructing Voronoi polygons shown in thick lines.

While it is not the case when γ is large (e.g., γ = 5 km), when γ is small (e.g., γ = 0.5 km) and the resulting cells cover small areas of road segments, many adjacent cells along the same roadway contain exactly the same set of trajectories (e.g., cells along a freeway without an on-ramp or off-ramp between them). We introduce an additional step that allows the merging of highly overlapping cells so that the final cells represent network partitions that are more distinct in terms of their users or entities. The merging also helps reducing the number of vertices in the resulting graph, thereby decreasing computational burden. For merging, we use a well-known density-based clustering algorithm, DBSCAN (Density-based spatial

clustering of applications with noise), which requires two parameters: Eps, the maximum neighbourhood distance and $MinPts$, the minimum number of required points in a cluster (Ester et al., 1996). In order to apply the DBSCAN algorithm on the cell object in our study, we slightly modify the definition of neighbours. Let ε be the maximum allowable radius of the neighbourhood (similar to Eps in the original DBSCAN) and ρ be the minimum required trajectory overlap between two cells expressed in terms of proportion ($0 \leq \rho \leq 1$). Then, cell i, denoted by c_i, is considered as a neighbour of cell j, denoted by c_j, if (i) the two cells are adjacent in terms of their Voronoi polygons, (ii) the distance between the two cell centroids is less than or equal to ε, and (iii) the trajectory overlapping ratio between two cells is greater than or equal to ρ. This is formally expressed as follows:

$$\begin{aligned} c_i.boundaries \cap c_j.boundaries &\neq \emptyset \\ Distance(c_i.centroid,\ c_j.centroid) &\leq \epsilon \\ Overlap(c_i.traj,\ c_j.traj) = \frac{|c_i.traj \cap c_j.traj|}{|c_i.traj \cup c_j.traj|} &\leq \rho \end{aligned} \quad \textbf{(1)}$$

where *c.boundaries* denotes the set of line segments forming the associated Voronoi polygon, *c.centroid* denotes the point coordinate of the centroid of cell c, and *c.traj* denotes the set of trajectories whose points belong to cell c. The first line indicates that two cells must share a boundary segment. In the second line, the Euclidean distance can be used as a distance function. In the third line, the Jaccard index is used to define trajectory overlapping ratio, which is shown in Eq. (1).

Let $MinCls$ represent the minimum number of cells in the neighbourhood (similar to $MinPts$ in the original DBSCAN). Then, we perform a DBSCAN clustering on the set of cells obtained from Procedure 1 using three parameters $MinCls$, ε, and ρ. The output of the DBSCAN is a label that maps each cell into either a new merged cell cluster or the noise group, which is a collection of cells that do not belong to any merged cluster. Then we create a new cell for each merged cell cluster while using each noise cell again as a separate cell. The merging procedure is described below.

Procedure 2: Merging adjacent cells based on overlapping trajectories
Input: *set of original cells* C, $MinCls$, ε, ρ,
Output: *set of merged cells* C

1. *Create a temporary set* C' = { }.
2. *Cluster cells in* C *using DBSCAN given* $MinCls$, ε, and ρ. *Separate clusters (merged cell groups) from noises (individual cells with no assigned cluster)*
3. *For each cluster,*
    a. *Create a new cell combining all the cells in that cluster.*
    b. *Update this new cell's centroid as the mean point of all member points obtained from the merged cells.*
    c. *Add this new cell to* C'.
4. *For each noise cell, consider it as a separate region and add this cell to* C'
5. *Set* $C = C'$ *and return* C

Figure 3 compares network partitioning cases with and without cell merging Figure 3 (a)-(b) show initial cells that are constructed based on a desired radius of 1km. Figure 3 (c)-(d) show cells after merging based on parameters ε=2km, ρ=0.5, and $MinCls$=2, which merge any two adjacent cells if their centroids are closer than 2km and the ratio of the number of overlapping trajectories to the total number of trajectories observed in either cell is greater than or equal 0.5. It can be observed from Figure 3 (c) that many consecutive cells are merged into a single cell.

Figure 2: Overall procedure of converting trajectory data into a dynamic graph of aggregated flows: (a) 1000 bus passenger trajectories, (b) partition the network into cells based on the data points in trajectories (c) set a graph vertex at each cell representing a geographical region; (d) connect vertices with bi-directional edges based on traffic flows between two regions.

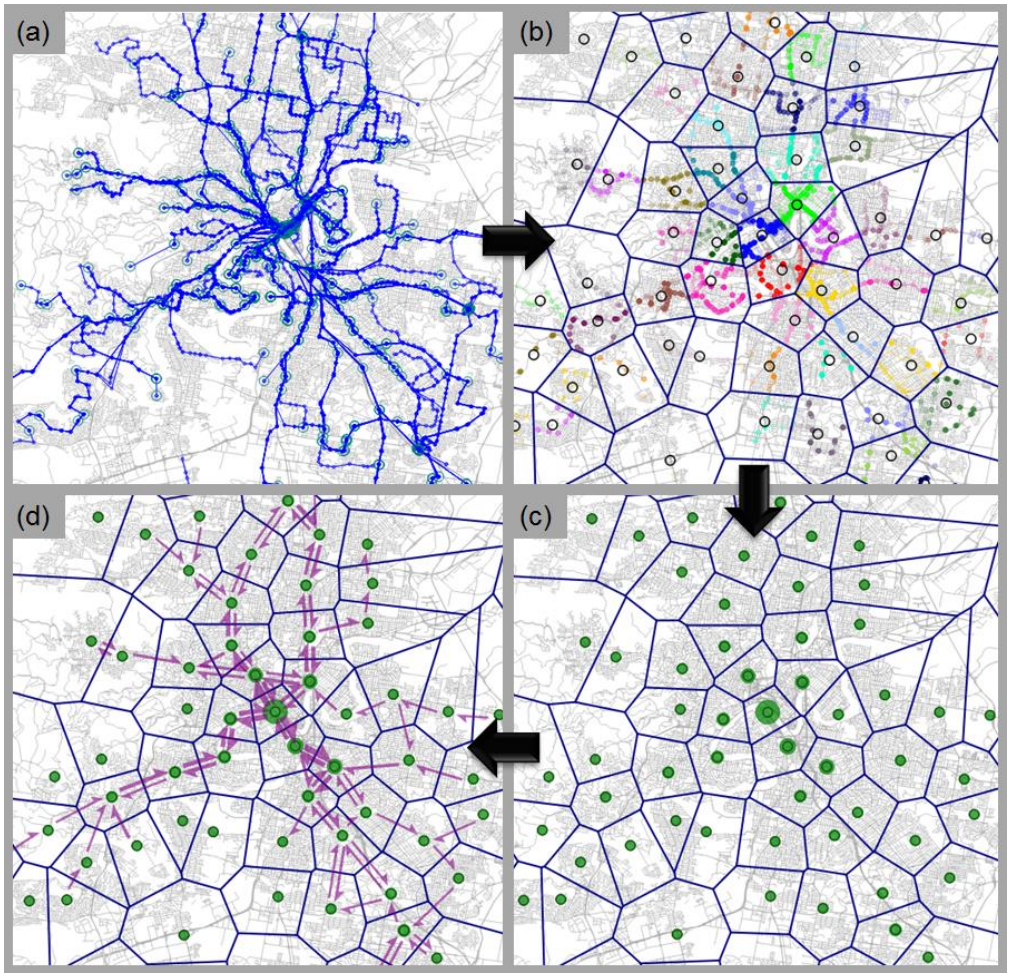
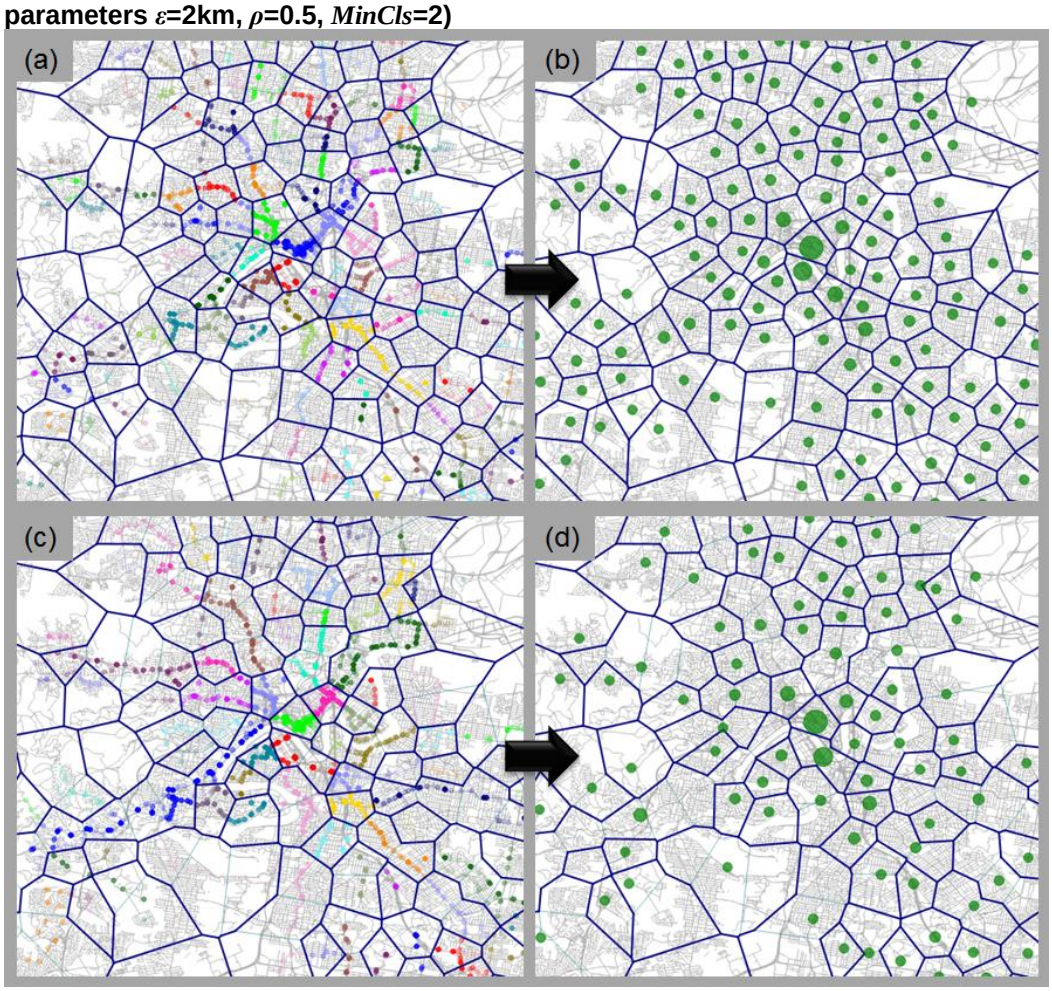


Figure 3: Merging cells: (a, b) construct Voronoi polygons (desired radius γ = 1km) and find cell centroids (green circles, size shows weight in terms of the number of trajectories passing a cell); (c, d) merge cells that share a high proportion of same trajectories (DBSCAN parameters ε=2km, ρ=0.5, $MinCls$=2)



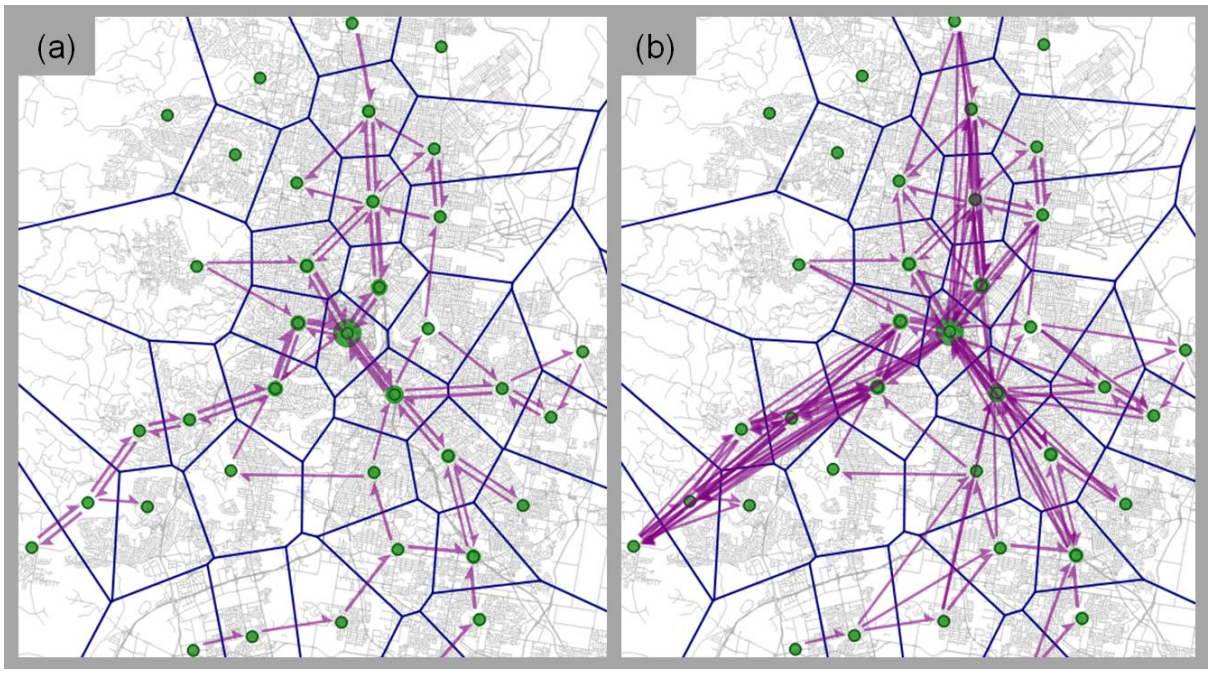
2.2. Generating a Dynamic Graph of Aggregated Flows

Once the network is partitioned into cells, these cells are used to define the vertex set of a trajectory flow graph that we will build in the second stage of the framework. In order to build a dynamic graph, we divide a day into T time slices with a fixed length Δt. For each time slice t = 1, ..., T, we then build a trajectory flow graph, $G_t = \left(V, E_t \subseteq (V \times V)\right)$, where V is the vertex set that represents the spatial cells identified above and E_t is the edge set at time slice t that represent traffic flows between cells. A time-ordered sequence of graph snapshots forms a trajectory flow graph series, denoted by $G = \{G_t\}_{t=1}^{T}$. It is noted that the vertex set V is not time dependent since we assume that the spatial regions of interest are fixed as we are only interested in characterising time-varying traffic flow patterns across those cells that are already defined. This is to provide a common ground for comparing graphs across different times of day as well as across different days.

For each time slice, we identify portions of trajectories that pass each cell during that time slice and calculate the number of trajectory flows between each pair of cells. A directed edge is added between vertices whenever a non-zero trajectory flow exists during a given time slice. The edges can be weighted by flow magnitude (the number of trajectories connecting

two cells). An example of trajectory flow graph is presented in Figure 2 (c)-(d), where the green circles represent the graph vertices, which are positioned at the associated cell centroids, and the purple arrows represent the graph edges with the arrow thickness proportional to the flow magnitude. Different types of flow patterns can be represented by a trajectory flow graph. Figure 4 show two examples: (a) ***cell-to-cell flow*** and (b) ***trajectory connectivity***. In the cell-to-cell flow representation, an edge is added from cell i to cell j if there is a trajectory passing cell i and cell j consecutively without visiting any other cells in between. This type of graph shows how overall flows are distributed across the cells and how each cell transfers flows from and to its adjacent cells through incoming and outgoing edges. In the trajectory connectivity representation, an edge is added from cell i to cell j if a trajectory was able to reach cell j from cell i within a given time slice, showing the connectivity or reachability between different parts in the network.

Figure 4: Different representations of aggregated traffic flows generated by trajectory flow graph: (a) cell-to-cell flow (b) trajectory connectivity



2.3. Creating a Concise Profile of City-wide Traffic Flows

One of the benefits of using a graph-based representation is that we can capture a more abstract view of the underlying data by applying various graph compression and summarization methods. In this paper, we use a graph partitioning technique to compress the trajectory flow graph for each time slice and attach similar time slices together to segment the entire daily graph series into a small number of graph segments, which allow us to exploit patterns and regularity in city-wide traffic flows during different time periods of the day. The third stage of the proposed framework is thus to create a temporal profile of city-wide traffic flows by segmenting a daily trajectory flow graph series. In segmenting a graph series, we apply a parameter-free change detection approach called *GraphScope* (Sun et al., 2007), which uses the *Minimum Description Language* (MDL) principle to automatically detect natural clusters of graph vertices in a graph snapshot and find "change points"—points in time where the entire structure of the graph changes—in a sequence of graph snapshots. In this approach, each graph snapshot G_t is expressed as a $|V| \times |V|$ binary adjacency matrix, where rows represent source cells and columns represent sink cells. We use the terms "source" and "sink", instead of "origin" and "destination", to refer to the cells

associated with the tail and head of a directed edge as the term “origin” and “destination” are used for journey origin and journey destination, i.e., the start and end point of a trajectory. In this study, all cells can be both source and sink. Entry (i, j) of the adjacency matrix is 1 if there is an edge from cell i to cell j and 0 otherwise. By reordering and grouping rows and columns, one can aim to divide the matrix entries into rectangular regions or blocks of ones and zeros as homogenous as possible, where a block of ones represents a fully interconnected vertex group and a block of zeros represents a group of vertices with no connection between them. Such a rearranged adjacency matrix will allow us to encode the entire graph structure in terms a few blocks of ones and zeros thereby reducing the cost of compressing the graph data. As such, this matrix rearrangement or graph partitioning can be achieved by minimizing the graph encoding cost measured by an MDL-based cost function, which is based on Shannon entropy. After finding the best row and column partitions in the adjacency matrix for each time slice, consecutive time slices that are similar in terms of their partition structures are grouped together such that the total encoding cost for the entire graph series can be minimised. For more details about the MDL-based graph partitioning and segmentation, readers are referred to the relevant literature (Chakrabarti et al., 2004; Sun et al., 2007).

Given graph series $G = \{G_t\}_{t=1}^{T}$, the outcome of this procedure is a set of change points T^c in the graph series and a set of graph segments $G = \{G_{(s)}\}_{s=1}^{|T^c|+1}$, where $G_{(s)}$ denotes the sth segment in the graph series representing a consecutive sequence of graph snapshots between the (s-1)th and sth change points. Each graph segment has its associated compressed adjacency matrix that describes the underlying graph connection structure for all the time slices belonging to that segment. Using these change points and identified graph segments, we can create a concise profile of the temporal evolution patterns in a network-wide traffic flow, which will be discussed in more detail in the case study below.

3. Case Study

3.1. Study area and data

The trajectory data used in this study are bus passenger trajectories generated from public transit smart card data in Brisbane, Australia. The smart card system in Brisbane (terms *go* card) is the major means by which over 80% of all urban public transport passengers pay transit fares. In the *go* card system, users need to touch their *go* cards to the on-board readers at the time of both boarding and alighting. Hence, the Brisbane smart card database includes information about both boarding and alighting locations and times. The *go* card data have been supplied by TransLink, the transit agency in Brisbane, Australia. By matching the *go* card transaction records with the underlying bus route information obtained from the associated General Transit Feed Specification (GTFS) data, we generate bus journey trajectories of individual bus passengers, where each trajectory represents a sequence of bus stops along the full journey of a particular passenger including transfers between his/her origin and final destination, with the location and time information for each stop.

The proposed framework is demonstrated using passenger trajectories from 11 March 2013, which was a normal Monday. 1000 trajectories are randomly sampled from the time period between 5:00am and 11:59pm and used for constructing a trajectory flow graph series. All the data points in trajectories are used as seed points S for building spatial cells. The following are the parameters used to perform the procedures described in sections 2.1 – 2.3.

- Network partitioning parameter: Desired cell radius γ = 1 km
- Cell merging parameters: $MinCls$ = 2, ε = 5 km, ρ = 0.5
- Graph building parameter: Time slice width Δt = 60 min
- Graph representation type: *Trajectory Connectivity*

3.2. Results

Figure 5 shows the result of the trajectory flow graph constructed using the above-mentioned parameters. After Procedure 1, a total of 251 cells are created. By merging overlapping cells based on Procedure 2, the cells are merged into the final 138 cells. Using these 138 cells as vertices, a dynamic graph of aggregated flows is created, where the edges are added to represent the trajectory connectivity defined in section 2.2. With the time slice width of 60min, a total of 19 graph snapshots are constructed to form a trajectory flow graph series, i.e., $G = \{G_t\}_{t=1}^{19}$. The graph series *G* is then segmented based on the GraphScope method.

Figure 6 shows the graph segmentation results. Figure 6(a) depicts how the graph series with 19 time slices are divided into 7 graph segments, i.e., $G = \{G_{(s)}\}_{s=1}^{7}$, where the y-axis represents the final (minimized) graph encoding cost for each identified graph segment, in the unit of cost per hour. We can observe that different segments incur different levels of encoding costs. The more complex the graph structure is, the higher the cost will be. Figure 6(b)-(d) show the adjacency matrices of the 2nd, 3rd, and 5th graph segments, where the rows and columns of each matrix has been reordered such that the matrix entries can be partitioned into a few homogenous blocks. The partitioned adjacency matrix for each graph segment then captures the underlying graph connection structure for all the time slices associated with that segment. We can see that there is a clear difference in the adjacency matrix between two consecutive graph segments, suggesting that the GraphScope method was able to detect points in time where the graph structure changes significantly. The structure of $G_{(2)}$, which represents the traffic during a morning peak period (6:00-11:00), is the most complex, followed by $G_{(5)}$, which represents the afternoon peak period (15:00-20:00). The $G_{(3)}$, the off-peak between the morning and afternoon peaks, shows relatively simple structure. To better understand the graph connection structure captured by each adjacency matrix, we take a closer look at selected partitions in these matrices by identifying the associated subgraphs as shown in Figure 7. The partitions A, B, and C in Figure 7 (a)-(c) correspond to the areas enclosed by three dotted rectangles A, B, and C in Figure 6(b)-(d), respectively. In Figure 7 (d)-(i), the subgraphs associated with the six matrix partitions shown in green circles are visualised on the network map view. In Figure 7 (a), the A-1 block shows a long vertical line indicating *many-to-one* connections, where edges are coming from many source cells and going to a single sink cell. This captures the inbound traffic pattern that is typical during the morning commuting period in Brisbane, where the single sink cell corresponds to the Brisbane central business district (CBD), as shown in Figure 7 (d). The square blocks A-2 and A-3 in Figure 7 (a) show vertex clustering or community structure where vertices are highly interconnected within each cluster but exhibit almost no connection between clusters. The corresponding flow graphs in Figure 7 (e)-(f) show that A-2 represents northbound and southbound traffic flows and A-3 represents traffic flows in northeast and southwest directions along the Brisbane River. After the morning peak period, the connections are concentrated in a few major cells, producing one single dense block B-1 in the adjacency matrix in Figure 7 (b). Within this block, one can observe a few vertical lines, suggesting that traffic flows are coming from many source cells to a few sink cells, as shown in Figure 7 (g). In Figure 7 (c), the C-1 block shows a long horizontal line indicating one-to-many connections, where edges are coming from a single source cell and going to many sink cells. This now captures the outbound traffic pattern that is typical during the afternoon peak period in Brisbane, as shown in Figure 7 (h). The patterns in A-1 and C-1 are consistent with our expectation that the Brisbane CBD becomes a major destination of the network traffic during the morning peak, while becoming the major origin during the afternoon peak. The C-2 block in Figure 7 (c) shows another community structure during the afternoon period and the associated vertex clusters are depicted in Figure 7 (i).

Figure 5: Trajectory flow graph constructed for the case study: (a) network partitioning results with merging (number of cells reduced from 251 to 138), (b) graph vertices positioned at cell centroids (138 vertices), and (c) graph edges representing trajectory connectivity (time slice 6:00-7:00)

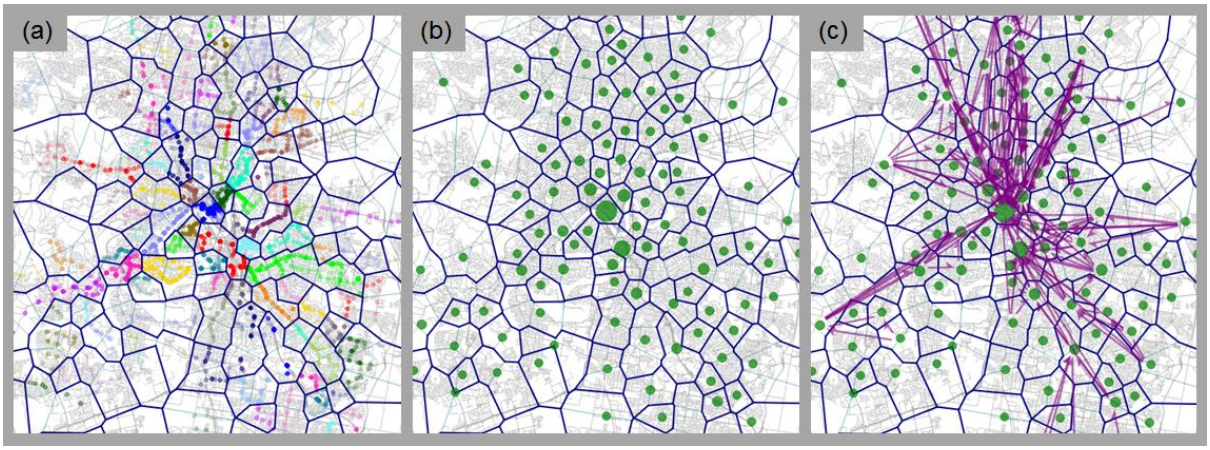


Figure 6: Graph segmentation results: (a) minimized MDL-based graph encoding cost for each identified graph segment, where dotted vertical lines indicate change points; (b-d) compressed adjacency matrices for three selected graph segments

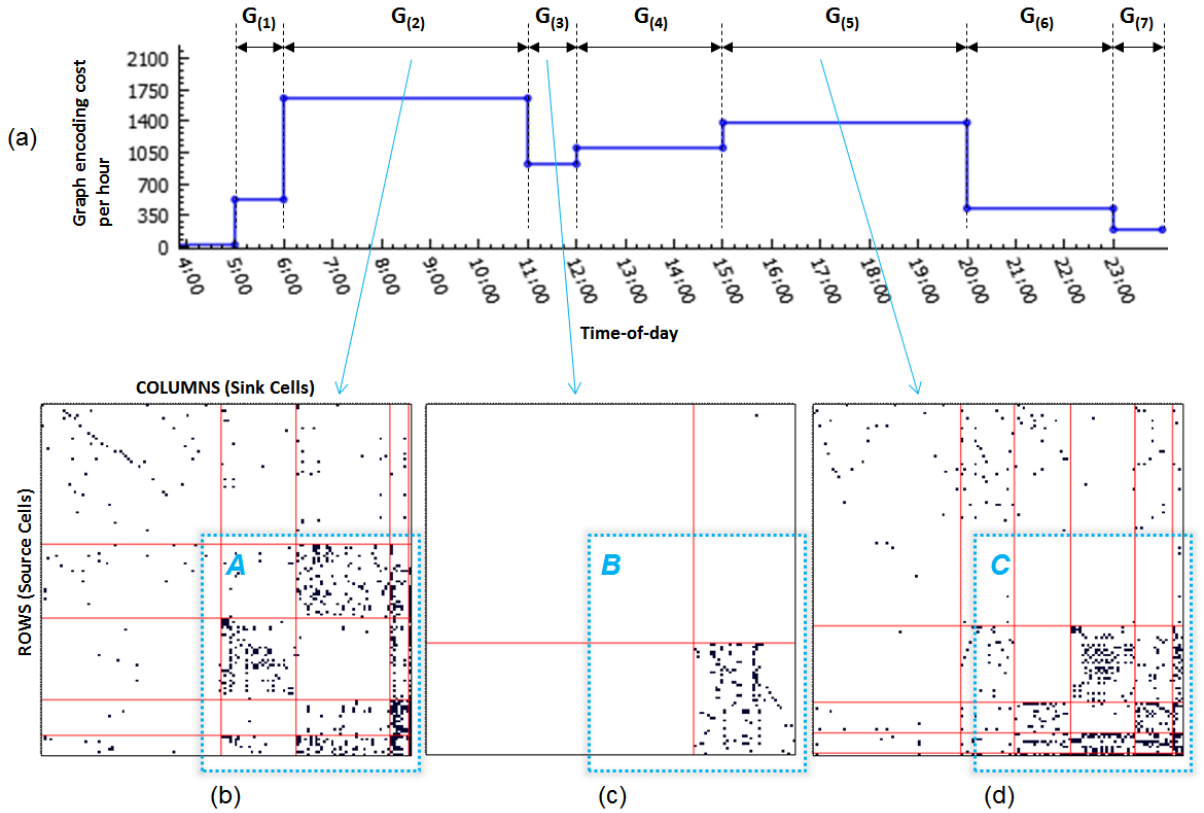
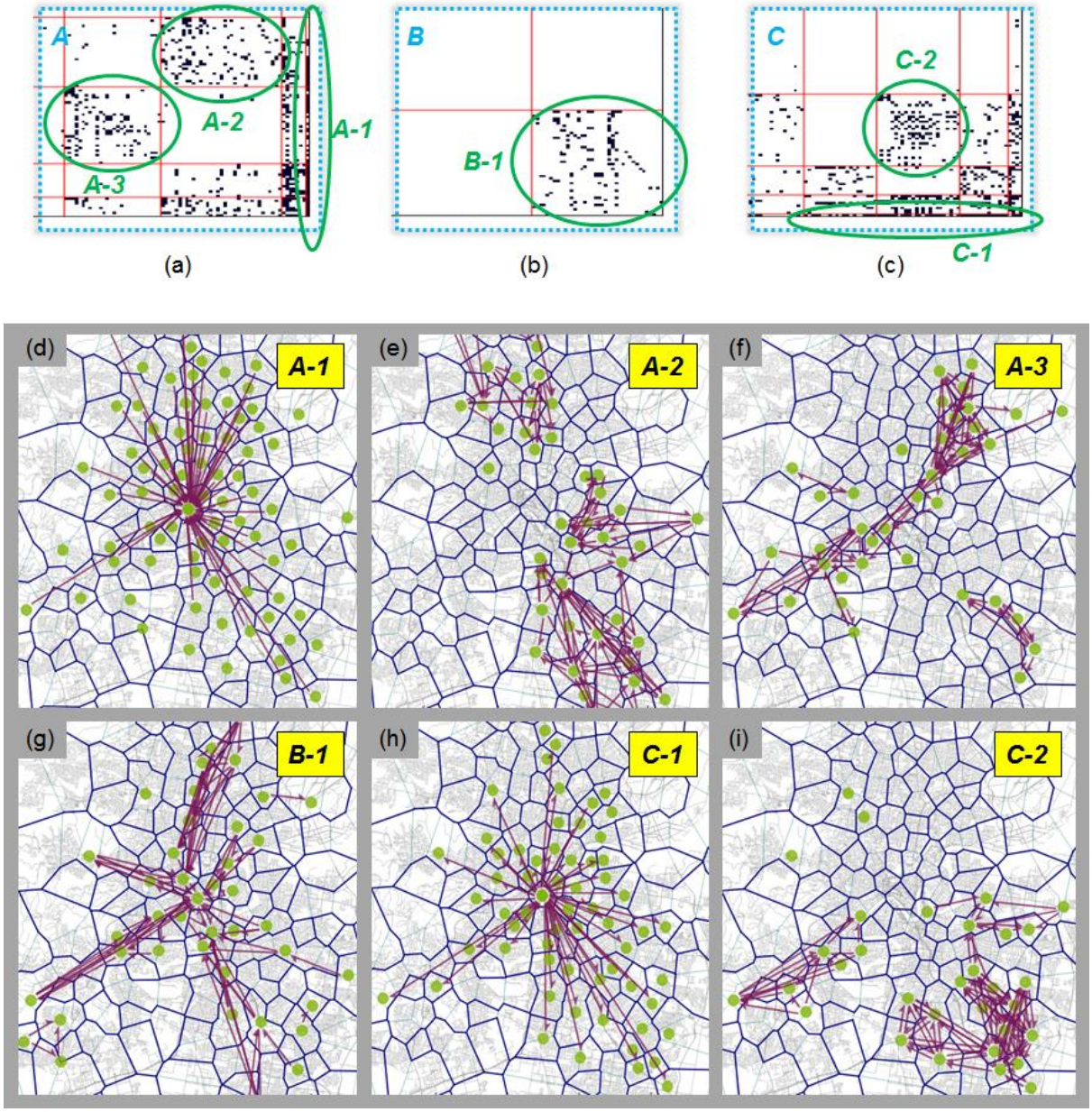


Figure 7: Selected partitions (blocks) in the adjacency matrices and the associated subgraphs



4. Conclusion

This paper proposes a graph-based analysis framework that aims to characterise spatial and temporal patterns of network-wide traffic flows. We propose the use of a *trajectory flow graph*, a dynamic graph of aggregated flows constructed from individual trajectories, to better understand and analyse city-wide mobility patterns. The proposed framework aims to apply data mining and graph mining techniques, coupled with a rich set of real-world trajectory data, to provide new ways of understanding, analysing, and visualizing traffic dynamics in a city. The paper presents approaches to partitioning the entire network into spatial cells and discusses methods to construct a trajectory flow graph at each time slice. The study applies a change point detection technique to divide the entire graph series into a set of graph. Using real-world trajectory data, this paper demonstrates the steps of spatial

partitioning (cell construction) and temporal segmentation (change-point detection). The graph-representation obtained from the case study shows a concise yet meaningful description of time-evolving patterns in the underlying trajectory data and allows us to create a robust profile of network-wide traffic flows that can be used in various applications such as anomaly detection and congestion prediction.

7. References

Adrienko, N., Adrienko, G., 2011. Spatial Generalization and Aggregation of Massive Movement Data. IEEE Transactions on Visualization and Computer Graphics 17, 205–219. doi:10.1109/TVCG.2010.44

Aurenhammer, F., 1991. Voronoi Diagrams—a Survey of a Fundamental Geometric Data Structure. ACM Comput. Surv. 23, 345–405. doi:10.1145/116873.116880

Chakrabarti, D., Papadimitriou, S., Modha, D.S., Faloutsos, C., 2004. Fully Automatic Cross-associations, in: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '04. ACM, New York, NY, USA, pp. 79–88. doi:10.1145/1014052.1014064

Ester, M., Kriegel, H., S, J., Xu, X., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. AAAI Press, pp. 226–231.

Kim, J., Mahmassani, H.S., 2015. Spatial and temporal characterization of travel patterns in a traffic network using vehicle trajectories. Transportation Research Part C: Emerging Technologies, Special Issue on International Symposium on Transportation and Traffic Theory 59, 375–390. doi:10.1016/j.trc.2015.07.010

Sun, J., Faloutsos, C., Papadimitriou, S., Yu, P.S., 2007. GraphScope: Parameter-free Mining of Large Time-evolving Graphs, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '07. ACM, New York, NY, USA, pp. 687–696. doi:10.1145/1281192.1281266

von Landesberger, T., Brodkorb, F., Roskosch, P., Andrienko, N., Andrienko, G., Kerren, A., 2016. MobilityGraphs: Visual Analysis of Mass Mobility Dynamics via Spatio-Temporal Graphs and Clustering. IEEE Transactions on Visualization and Computer Graphics 22, 11–20. doi:10.1109/TVCG.2015.2468111