Partitioning Road Networks using Density Peak Graphs: Efficiency vs. Accuracy

Tarique Anwar[a,c], Chengfei Liu[a], Hai L. Vu[a], Christopher Leckie[b,c]

[a]*Swinburne University of Technology, Melbourne, Australia*
[b]*University of Melbourne, Melbourne, Australia*
[c]*Data61, CSIRO, Melbourne, Australia*

Abstract

Road traffic networks are rapidly growing in size with increasing complexities. To simplify their analysis in order to maintain smooth traffic, a large urban road network can be considered as a set of small sub-networks, which exhibit distinctive traffic flow patterns. In this paper, we propose a robust framework for spatial partitioning of large urban road networks based on traffic measures. For a given urban road network, we aim to identify the different sub-networks or partitions that exhibit homogeneous traffic patterns internally, but heterogeneous patterns to others externally. To this end, we develop a two-stage algorithm (referred as `FaDSPa`) within our framework. It first transforms the large road graph into a well-structured and condensed density peak graph (DPG) via density based clustering and link aggregation using traffic density and adjacency connectivity, respectively. Thereafter we apply our spectral theory based graph cut (referred as α-Cut) to partition the DPG and obtain the different sub-networks. Thus the framework applies the locally distributed computations of density based clustering to improve efficiency and the centralized global computations of spectral clustering to improve accuracy. We perform extensive experiments on real as well as synthetic datasets, and compare its performance with that of an existing road network partitioning method. Our results show that the proposed method outperforms the existing normalized cut based method for small road networks and provides impressive results for much larger networks, where other methods may face serious problems of time and space complexities.

Keywords: Spatial partitioning, Road networks, Spectral clustering, Density peak graph.

1. Introduction

These days there is an increase in the frequency of traffic congestion on urban road networks, especially during the peak hours and in the city centers. This increasing congestion requires an improvement in its management by learning from its behavior to help balance the traffic flow. Usually the roads of each locality, say inside a suburb, experience a specific traffic flow pattern regardless of the global flow. For example, roads inside the city centre or any area having popular venues like a stadium or hospital, usually remain more congested than others without any such significance. Additionally, the congestion on roads connecting important places of public gatherings like airports, train stations, hospitals, and bus stops, remains comparatively higher than other locations. Thus different subnetworks of the urban road network exhibit congestion at different times. To analyze the behavior of congestion it is important for traffic management authorities to be able to partition an urban road network into different sub-networks based on the road connectivities and their congestion level, which is determined by the real traffic measures [3].

Moreover, as we move towards smart urban infrastructure, there is a growing demand for traffic-aware smart travel services, including route guidance and trip planning. These services are usually based on complex graph processing methods dealing with the road network. One way to efficiently process the execution of these services is to exploit computation in a distributed computing environment, in which the large road network is partitioned into several small sub-networks, so that queries can be focused on the relevant sub-networks [32]. Thus there exist different applications where instead of using the complete urban network, problem solving can be simplified by separately processing the smaller sub-networks that exhibit homogeneous traffic patterns inside. This leads to the important problem of traffic-based spatial partitioning of urban road networks. The application of graph partitioning on general information networks has been studied in the past [42, 17]. However, the geospatial properties of a road network associated with traffic flow patterns makes a unique kind of network [14]. The problem was recently raised in the intelligent transportation systems (ITS) community [14], where the authors proposed a normalized cut based method for partitioning road networks. While this works well for small networks, it faces serious limitations in its time and space complexity for large networks.

In our recent work [3], we proposed a method for traffic-based spatial partitioning of large road networks that outperformed existing techniques. The method comprises three different modules– road graph construction, road supergraph mining, and supergraph partitioning. The first module deals with transforming the real road network into a road graph to give it a mathematical representation. To address the problem of

Email addresses: `tAnwar@swin.edu.au` (Tarique Anwar), `cliu@swin.edu.au` (Chengfei Liu), `hvu@swin.edu.au` (Hai L. Vu), `caleckie@unimelb.edu.au` (Christopher Leckie)

large number of road segments in large urban road networks, we followed a 2-level partitioning. The second module is the first level partitioning, which mines a road supergraph from the road graph with a much reduced order following a bottom-up approach. It goes through the steps of clustering feature values using k-means and constructing the road supergraph. The last module of supergraph partitioning is the second level partitioning, which follows a top-down approach to split up the supergraph into multiple heterogeneous partitions that are homogeneous within. It is achieved by approximately optimizing a measure called α-*Cut*, by following a spectral clustering based solution. It produces supernode partitions, from which the road segment partitions are extracted. Despite obtaining good results the following issues are still outstanding, *i*) the problem of learning the right number of clusters, while applying k-means to create supernodes, is a computationally expensive task; *ii*) when the value of k in k-means is set very low, the number of supernodes is sometimes still very large, implying the relation between k and the supernodes is weak; *iii*) the connectivity among the nodes is not considered together with their feature values when applying clustering (k-means) to mine the supergraph. In this paper, we address the above issues and present a robust framework employing both density and spectral based clustering. It is known that spectral clustering based solutions provide good results but exhibit high computational complexity [39, 3]. On the other hand, density-based methods are able to discover clusters of arbitrary shapes and are very fast. Our framework combines the advantages of spectral and density based approaches simultaneously, and also overcomes the issues that existed in our previous work [3].

Our partitioning framework aims to identify the different heterogeneous regions of an urban network that internally exhibit homogeneous traffic patterns. We propose three algorithms, `FaDPa`, `FaDPa+`, and `FaDSPa`. The main scalable algorithm `FaDSPa`, which is based on the other two, mines a *density peak graph* by identifying the density peaks from the *road graph*. Then the density peak graph is subjected to our spectral theory based α-Cut to obtain the set *road network partitions*. In summary, we make the following contributions in this paper.

- We develop a fast density-based road network partitioning method `FaDPa` (extended to `FaDPa+`). It identifies the density peaks locally in the graph, and gradually grows them to form clusters. Unlike spectral clustering methods, it works very fast, and is highly suitable to large networks.
- Using `FaDPa`, we develop an efficient and effective method `FaDSPa` for partitioning small as well as large road networks. It provides an option to input a factor to control the trade-off between efficiency and partitioning quality.
- We present the complete derivation to optimize the α-Cut objective function (proposed in our previous preliminary work [3]) that is used in `FaDSPa`.
- We perform extensive experiments on real as well as synthetic datasets including road networks of different sizes to establish its efficacy.

The rest of the paper is organized as follows. Section 2 presents some preliminary theories followed by the problem definition and framework overview. Section 3 presents our density-based partitioning algorithm `FaDPa` and its extension `FaDPa+`, followed by the main algorithm `FaDSPa` in Section 4. Experimental results are shown in Section 5, followed by related work in Section 6. Section 7 concludes the paper with some future research directions.

2. Problem Definition and Framework Overview

This section presents some preliminary theories, defines the problem, and presents the framework overview.

2.1. Road Networks and their Mathematical Representation

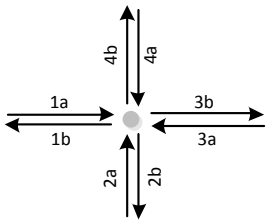
Urban roads exist in the form of a physical network spatially spread over a large urban area. To make it a machine-interpretable network, we need to give it a mathematical representation in the form of a graph, which we name as a *road graph*. The unique features associated with this kind of network, like varying spatial importance of different roads and the traffic flow being unidirectional on some roads whereas bidirectional on others, make it a challenging task to give a realistic mathematical representation. Previous works have represented it in different graph-based structures that suited the application area [15, 9].

Unlike the previously attempted problems, the focus of spatial partitioning of road networks is on the road segments, not on the intersection points. A trivial representation in the form of a graph by considering roads as links and their intersection points as nodes is not suitable as its partitioning results into subsets of intersection points, which is not the objective. To make the representation applicable to spatial partitioning, we transform the actual road network into its dual, which forms an undirected road graph. This transformation is an improved version of that used in our earlier work [3].

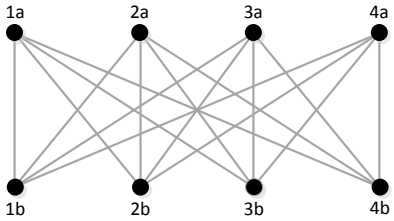
Definition 1: (**Road Network**) A real urban road network is defined as $\mathcal{N} = (\mathcal{I}, \mathcal{R})$ comprising a set of intersection points $\mathcal{I} = \{\iota_1, \iota_2, \ldots, \iota_{n_i}\}$ as nodes that are connected among themselves by the set of directed road segments $\mathcal{R} = \{r_1, r_2, \ldots, r_{n_r}\}$ as its links, where each road segment r_i associates the traffic density $r_i.d$ with itself. ■

Definition 2: (**Road Graph**) Given a road network $\mathcal{N}$, the corresponding road graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is constructed by adding each road segment $r_i \in \mathcal{N}$ as a node v_i, and establishing an undirected link e_i between each possible node pair (v_j, v_k) if there exists at least one intersection point ι_l which is a common intersection for the roads r_j and r_k, and the traffic can flow either from r_j to r_k or vice versa, as shown in Equation 1.

$$\begin{aligned} \mathcal{V} &= \{v_1, v_2, \ldots, v_{n_r}\}, \text{where } v_i = node(r_i) \\ \mathcal{E} &= \{link(v_j, v_k) : \exists\ \iota_l \text{ as the commnon intersection point for } r_j \text{ and } r_k\} \end{aligned} \quad (1)$$



(a) Intersection point



(b) Bipartite formation

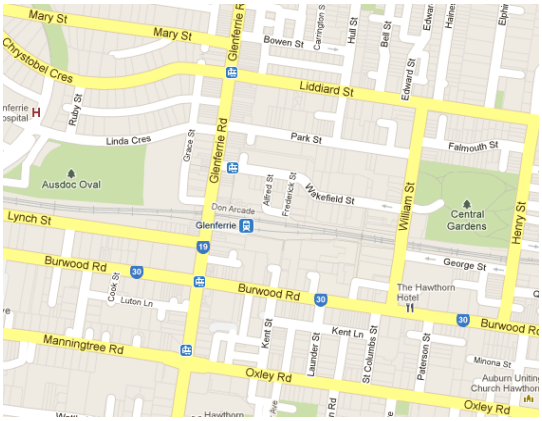
Figure 1: Star topology to bipartite formation

Thus the links stand for the adjacency relationships among the road segments. In this manner, the road network components in a star topology form bipartites in the road graph, as shown in Figure 1, where each partite stands for either incoming flow to or outgoing flow from a common intersection point. Each node v_i $(node(r_i)) \in \mathcal{V}$ associates with it a feature value $v_i.f$ which is the road traffic density $r_i.d$. ■

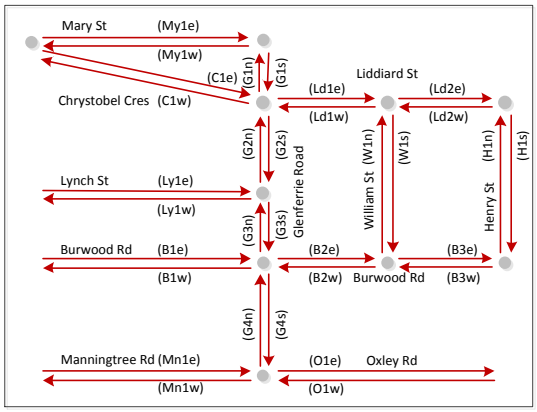
Most urban roads exist as two-way roads, which are two oppositely directed one-way parts separated from each other. Each of the two parts (directions) undergo different kinds of traffic flow patterns. For example, in the morning office hours, a road that connects outskirts with the city center would find more traffic heading towards the city center than the opposite direction. This feature of the urban network is accommodated in Definition 2 by considering the two traffic directions as separate road segments, if they share a common intersection point and thus are adjacent. Figure 2 shows an example of our road network representation in which Figure 2(a) is a sample road map, Figure 2(b) is the corresponding road network of those colored yellow in the map, and Figure 2(c) is the final representation called the road graph. When representing any kind of network in the form of a graph, normally the main objects of study in the network are considered as nodes and the links define the affinity between them. In the road network in Figure 2(b), we can see that nodes represent the intersection points of roads, which actually do not have much importance for the problem of spatial partitioning as compared to roads, which appear as links. The road graph in Figure 2(c) solves this problem as the objects of study are represented as nodes.

2.2. Problem Definition

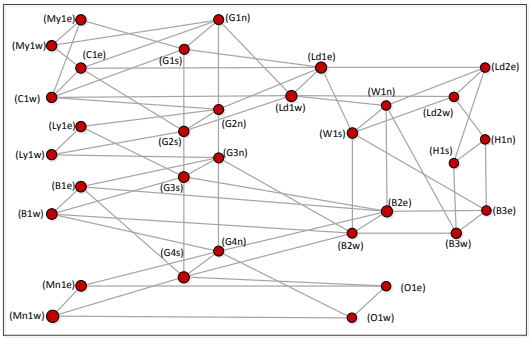
The problem of *partitioning road networks* addressed in this paper is defined as splitting a given urban road network based on traffic measures into several disjoint partitions, keeping intact the associated spatial properties. The different partitions



(a) Actual road map



(b) Road network



(c) Road graph

Figure 2: Mathematical representation of road networks

exhibit the property of intra-partition traffic *homogeneity* and inter-partition traffic *heterogeneity*. Let us suppose we have a real urban directed road network $\mathcal{N} = (\mathcal{I}, \mathcal{R})$, which is transformed into a *road graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ by following the method described in Section 2.1. Before formally stating the problem, we present four definitions.

Definition 3: (**Cost of Partitioning**) While partitioning the set of nodes $\mathcal{V}$ in a road graph $\mathcal{G}$ into different partitions $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k\}$, the *cost of partitioning* is defined as the aggregation of *affinity values* of all possible node pairs (v_i, v_j) for which v_i and v_j lie in different partitions in the final result,

where the affinity values are measures of traffic similarity between nodes in the pairs. ■

Definition 4: (**Partition Volume**) Given a set of road graph partitions $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k\}$, *partition volume* is defined as the aggregation of *affinity values* of all possible linked pairs (v_i, v_j) for which v_i and v_j lie in the same partition. ■

Definition 5: (**Partition Connectivity**) A partition $\mathcal{P}_l = (\mathcal{V}_l, \mathcal{E}_l)$ is said to be *connected* if for any node pair $(v_i, v_j) \in \mathcal{P}_l$ there exists a path from v_i to v_j (or vice versa), such that each node v_k in the path belongs to $\mathcal{V}_l$ (i.e., $v_k \in \mathcal{V}_l$). ■

The problem of traffic-based spatial partitioning of a road graph $\mathcal{G}$ is to split its node set $\mathcal{V}$ into k partitions (or subsets) $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k\}$ such that the following conditions hold.

C.1 $\bigcup_{i=1}^{k} \mathcal{P}_i = \mathcal{V}$ and $\mathcal{P}_i \bigcap \mathcal{P}_j = \emptyset$ for all $i \neq j$;

C.2 each $\mathcal{P}_i$ is *connected*, and all adjacency relations, except the cross-partition relations (inter-partition links), are maintained as in $\mathcal{G}$;

C.3 the *partition volume* of $\mathcal{G}$ is the maximum; and

C.4 the *cost of partitioning* $\mathcal{G}$ is the minimum;

In the above conditions, **C.1** is a general condition of grouping the set of nodes (or road segments) into k non-overlapping subsets, **C.2** introduces the spatial connectivity (or linkage) of nodes, **C.3** enforce the condition of intra-partition traffic homogeneity, and **C.4** enforces inter-partition traffic heterogeneity. A partitioning may not satisfy **C.3** and **C.4** together simultaneously. Optimizing one of them may lead to sacrificing the other condition. Therefore, our goal is to make an optimized trade-off between **C.3** and **C.4**.

2.3. Framework Overview

The task of road network partitioning is to cluster the road segments of a given road network based on their traffic measures and the associated spatial connectivities (connectivity of road segments). However, the traditional clustering algorithms, like k-means, do not take care of the connectivities directly. It requires to develop ways to incorporate the connectivities during clustering in an efficient and effective manner. In the proposed framework shown in Figure 3, our partitioning algorithm called `FaDSPa` uses a combination of an efficient density-based clustering approach and an effective spectral clustering approach. It starts with constructing a road graph from the given road network. The graph is passed to the partitioning algorithm `FaDSPa` to obtain the set of partitions. Lastly the real road network partitions are extracted from the resulting road graph partitions.

The transformation of the real road network $\mathcal{N}$ into a road graph $\mathcal{G}$ is done in the beginning to give it a mathematical representation, explained as a preliminary step in Section 2.1. Due to the large and rapidly expanding nature of urban areas, the size of an urban road network $|\mathcal{R}|$ and the order of the corresponding road graph $|\mathcal{V}|$ may become very large, which heavily affects the time and space complexity for partitioning $\mathcal{G}$. To address this problem, the framework follows a two-level partitioning (`FaDSPa`), where the first level is fast and the second level produces quality partitions. The first level follows a bottom-up approach and applies a density based algorithm called `FaDPa+` to compress the large graph $\mathcal{G}$ into a small density peak graph $\mathcal{G}^d$ (defined later) by identifying the locally dense components. The second level partitioning follows a top-down approach to split up the density peak graph $\mathcal{G}^d$ into multiple heterogeneous partitions that are internally homogeneous. It is achieved by approximately optimizing α-Cut, by following a spectral clustering based solution. It produces partitions of the density peak graph, from which the road segment partitions are extracted.

The density based `FaDPa+` is fast and thus suitable for large networks. On the other hand, the spectral based α-Cut produces quality partitions, but comes with high time and space complexity, and thus is suitable for small networks. Depending on the available computing resources and processing time, `FaDSPa` maintains a balance between the efficiency and accuracy of the partitioning task, by using an input parameter. If the urban network is small in size (manageable by the available resources), the task is done more by the α-Cut, and if it is large (beyond manageable by the available resources), the task is transferred more to `FaDPa+`. This makes `FaDSPa` effective as well as efficient in dealing with graphs of all sizes.

We propose `FaDPa` (in Section 3) as a fast density-based partitioning algorithm, which is further extended to `FaDPa+` (in Section 3.4) to partition into any desired small number of clusters, and `FaDSPa` (in Section 4) as a combined density and spectral based partitioning algorithm. `FaDSPa` is the main partitioning algorithm that is able to handle all small to large urban road networks, by following an appropriate balance between the density based (`FaDPa+`) and spectral based (α-Cut [3]) algorithms.

3. FaDPa: Fast Density-based Partitioning

The road segments inside a road sub-network or partition are linked together. Any vehicle entering into a partition through a road segment needs to go through the following segments to cross the partition or reach the destination. It makes the traffic pattern of a road segment more likely to be similar to (or dependent on) other (following or preceding) segments inside the partition. Also in each partition, locally there exist some important road segments that are spatially more closely connected to others and play a special role in the traffic movement. The road network segments including these important roads and the surrounding roads form dense components with high similarity in the traffic density, where the most important and dominating road occupies the density peak. The traffic on the surrounding roads, other than the density peak, is heavily dependent on the peak, which again have following roads that depend on these nearby roads. In this section we use this natural phenomenon of road traffic networks to propose a fast density-based network partitioning method called `FaDPa` (pronounced as *fad-paa*). It first identifies the density peaks in a network and then grows them to identify the density-based clusters.

There exist density based clustering algorithms like DBSCAN [7], which are efficient, able to detect clusters of arbitrary shapes, and able to find the suitable number of clusters

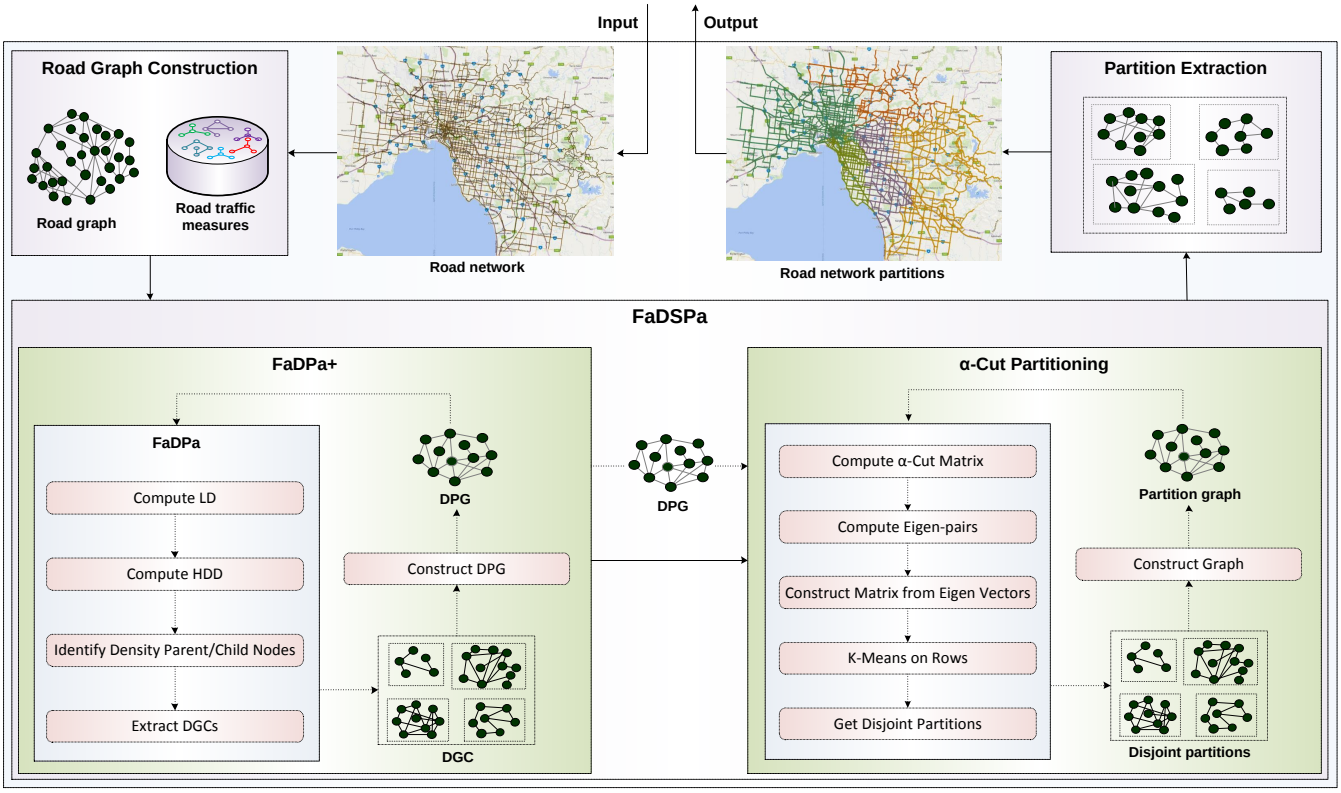


Figure 3: Architecture of the proposed framework

automatically. They identify a cluster by looking into the neighborhood of each object within a radius of a predefined threshold ϵ distance. With each *minpts* (predefined) objects in the ϵ-neighborhood, a new cluster is formed. The process is carried out to find all density-connected clusters, where a density-connected cluster is defined as the maximal set of density-connected objects. The main drawback of this method is the predetermination of ϵ and *minpts* thresholds, and their high sensitivity to cluster formation [28]. The method we propose here is free from these requirements. We start with presenting the main concepts and terminology, which is followed by the algorithm.

3.1. Concepts and Terminology

We use some of the ideas of [28] in Definitions 6 and 7 to find the density peaks in unlinked data, and then extend them to graph data.

Definition 6: (**Local Density (LD)**) Given a set of data objects $\mathcal{D} = \{d_1, d_2, ..., d_{(n_d)}\}$, the local density $\rho(d_i)$ of an object d_i is defined as the number of objects closer than a predefined distance threshold ϵ^d to d_i. It is formulated in Equation 2, where $dist(d_i, d_j)$ gives the distance[1] between d_i and d_j in terms of their feature values, and $\chi(.)$ is a binary function defined in Equation 3. ■

$$\rho(d_i) = \sum_j \chi(dist(d_i, d_j) - \epsilon^d) \tag{2}$$

$$\chi(x) = \begin{cases} 1 & \text{if } x < 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

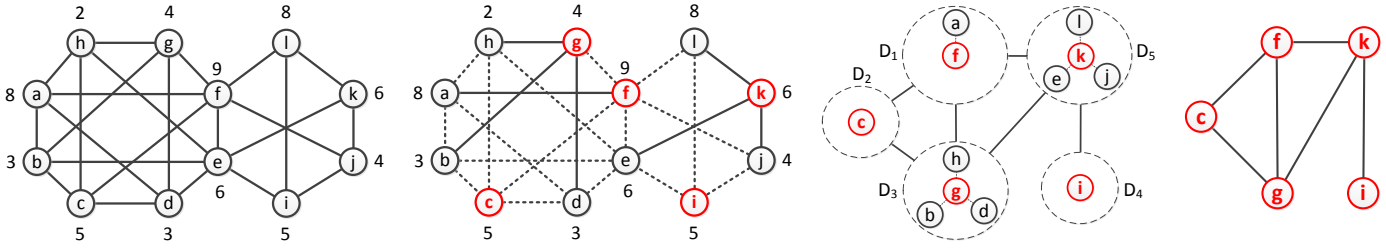
Definition 7: (**Higher Density Distance (HDD)**) Given a set of data objects $\mathcal{D} = \{d_1, d_2, ..., d_{(n_d)}\}$, the higher density distance $\delta(d_i)$ of an object d_i is defined as the distance from d_i to the closest object d_j of higher *local density*. It is formulated in Equation 4 as the minimum distance between d_i and any other object d_j with higher density. ■

$$\delta(d_i) = \min_{\forall d_j : \rho(d_j) > \rho(d_i)} dist(d_i, d_j) \tag{4}$$

Definition 8: (**LD in Graph**) Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the LD $\rho^g(v_i)$ of a node v_i is defined as the number of nodes that are directly linked to v_i and closer than a predefined distance threshold ϵ^d. It is formulated in Equation 5, where $neigh(v_i)$ returns all the neighboring or linked nodes to v_i, $dist(v_i, v_j)$ returns the distance between v_i and v_j in terms of their feature values, and $\chi(.)$ is the same binary function defined in Equation 3. ■

$$\rho^g(v_i) = \sum_{\forall v_j \in neigh(v_i)} \chi(dist(v_i, v_j) - \epsilon^d) \tag{5}$$

[1] We use Gaussian based distance measure defined later in Section 3.2



(a) Sample road graph (b) Identification of DPNs (c) DGCs (d) DPG

Figure 4: Illustration of DPG construction from a graph

Table 1: Distance measures in the sample road graph

	Distance measures												LD
	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	
a	-	0.93	-	0.93	-	**0.10**	-	0.98	-	-	-	-	1
b	0.93	-	**0.35**	-	0.62	-	**0.10**	-	-	-	-	-	2
c	-	**0.35**	-	**0.35**	-	0.82	-	0.62	-	-	-	-	2
d	0.93	-	**0.35**	-	0.62	-	**0.10**	-	-	-	-	-	2
e	-	0.62	-	0.62	-	0.62	-	0.82	**0.10**	-	**0.00**	-	2
f	**0.10**	-	0.82	-	0.62	-	0.93	-	-	0.93	-	**0.10**	2
g	-	**0.10**	-	**0.10**	-	0.93	-	**0.35**	-	-	-	-	3
h	0.98	-	0.62	-	0.82	-	**0.35**	-	-	-	-	-	1
i	-	-	-	-	**0.10**	-	-	-	-	**0.10**	-	0.62	2
j	-	-	-	-	-	0.93	-	-	**0.10**	-	**0.35**	-	2
k	-	-	-	-	**0.00**	-	-	-	-	**0.35**	-	**0.35**	3
l	-	-	-	-	-	**0.10**	-	-	0.62	-	**0.35**	-	2

Example 1: Figure 4(a) shows an example of a road graph constructed from a small road network, in which exemplary node feature values are shown beside the nodes, and Table 1 shows the distance measures computed for each pair of nodes. Setting the distance threshold ϵ^d to 0.5, the distances lower than this threshold are highlighted (bold) in the table, and the rightmost column shows the node *local density* as the count of these highlighted entries in each row. ■

Definition 9: (**Density Parent**) Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the *density parent* of a node v_i is defined as the linked node v_j having the closest higher *local density*, such that $\chi(dist(v_i, v_j) - \epsilon^d) = 1$. If there are multiple nodes equally close to v_i in terms of LD, then the one with the lowest $dist(v_i, v_j)$ is chosen as the parent. ■

Definition 10: (**Density Child**) Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the *density children* of a node v_i is defined as the set of linked nodes $\{v_j\}$ that have v_i as their *density parent*. A node can have multiple density children. ■

Definition 11: (**HDD in Graph**) Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the HDD $\delta^g(v_i)$ of a node v_i is defined as the distance from v_i to its *density parent* v_j if v_j exists (Equation 6), otherwise it is the maximum of all distances between any two linked nodes (Equation 7). In the equations, $neigh(v_i)$ returns all the neighboring or linked nodes to v_i, $dist(v_i, v_j)$ returns the distance between v_i and v_j in terms of their feature values, and $v_k \Leftrightarrow v_l$ denotes that v_k is linked to v_l. ■

$$\delta^g(v_i) = \min_{\forall v_j} \left\{ dist(v_i, v_j) \right\} \tag{6}$$

$$\delta^g(v_i) = \max_{\forall v_k, v_l, v_k \Leftrightarrow v_l} \{ dist(v_k, v_l) \} \tag{7}$$

Definition 12: (**Density Peak Node (DPN)**) Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a node v_j is called a *density peak node* ς_i, if v_j does not have any *density parent*. Like nodes, the DPNs also associate a feature value $\varsigma_i.f (= v_j.f)$ with them. ■

Definition 13: (**Density Similar**) Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, two nodes v_i and v_j are said to be *density similar*, if they have a *density parent* and *density child* relationship, or if there is another node v_k such that v_i is *density similar* to v_k and v_k is *density similar* to v_j. Hence this relationship is both *reflexive* and *transitive*. ■

Definition 14: (**Graph Component (GC)**) Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a *graph component* is defined as a subgraph in which there exists a path between any two nodes $v_i, v_j \in \mathcal{V}$ in such a way that each node v_k in the path belongs to $\mathcal{V}$. ■

Definition 15: (**Dense Graph Component (DGC)**) Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a *dense graph component* D_i is defined as a *graph component* in which each pair of nodes (v_i, v_j) are *density-similar*. Every DGC must have exactly 1 DPN, which will not have any density parent, whereas all other nodes in the DGC must have. ■

Definition 16: (**Density Peak Graph (DPG)**) Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a *density peak graph* $\mathcal{G}^d$ is defined as a 3-tuple $\left(\mathcal{V}^d, \mathcal{E}^d, \mathcal{W}^d\right)$, where $\mathcal{V}^d = \left\{\varsigma_1, \varsigma_2, \ldots, \varsigma_{n_\varsigma}\right\}$ is the set of DPNs, $\mathcal{E}^d = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{n_\varepsilon}\}$ is the set of links connecting the DPNs, and $\mathcal{W}^d = \{\omega_1, \omega_2, \ldots, \omega_{n_e}\}$ is the set of weights associated with each of the corresponding links. The links between the DPNs are established by looking into the neighborhood relationships between the corresponding DGCs. For each pair of DPNs $(\varsigma_i, \varsigma_j)$ in the DPG, if there exists a link e_k between a pair of nodes (v_p, v_q), such that ($v_p \in D_i$ and $v_q \in D_j$) or ($v_q \in D_i$ and $v_p \in D_j$), then a link ε_l is established between them. The weight of this link is set as a measure of similarity between ς_i and ς_j, i.e., $\omega_l = sim(\varsigma_i, \varsigma_j)$ (defined later in Equation 11). ■

Example 2: In Table 1, the LD of node a is 1. To find its *density parent*, we look into the LD of all the linked nodes (i.e., b, d, f, h) that have their distance less than $\epsilon^d (= 0.5)$ (i.e., f), and select the node having the closest higher LD, which is f. Thus a becomes *density child* of f, and f becomes the *density*

parent of a. After establishing this relationship, a and f are called to be *density similar*. In the set $\{b, d, g, h\}$, b, d, and h are children of g, which makes all the nodes *density similar* to each other. Therefore the set forms a *dense graph component*. For a node, if there does not exist any linked node with higher LD, then it forms the *density peak*. As shown in the table, g has b, d, and h as the linked nodes satisfying the ϵ^d condition, but none of them have their LD higher than g. Therefore, g becomes a *density peak node*. Figure 4(b) shows the DPNs (colored) found in the sample graph. The solid lines represent a parent-child relationship and the dotted lines represents a link in the road graph. Figure 4(c) shows the identified *dense graph components* enclosed in the circles with solid lines, where the colored nodes are the DPNs, and the links with solid lines represent the neighborhood relationship between the DGCs. Figure 4(d) shows the *density peak graph*, where the nodes are the identified DPNs linked by the neighborhood relationship. ∎

For a given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the density *parent-child* relationships among the nodes can be easily established after computing their LD and HDD. According to Definitions 9 and 12, all nodes must have a density parent, unless they are DPNs. It means that except the DPNs, all other nodes in $\mathcal{V}$ can be accessed by traversing through the children of DPNs, followed by their children, and so on, until the nodes do not have any children. This traversal from a single DPN results into accessing a complete DGC, and doing this for all the DPNs, gives the complete set of DGCs, which include all the nodes in $\mathcal{V}$. It leads to the conclusion that any given $\mathcal{G}$ can be decomposed into a set of DGCs, where each of them have one DPN. These DPNs are the density peaks, which form the center of attention in a surrounding. They become nodes in the DPG $\mathcal{G}^d = (\mathcal{V}^d, \mathcal{E}^d, \mathcal{W}^d)$, while the remaining surrounding nodes in $\mathcal{V}$ disappear, as shown in Figure 4. Each DPN represents its corresponding DGC, and thus constructing a DPG from a road graph condenses the graph using the density peaks.

3.2. Algorithm

The algorithm (shown in Algorithm 1) starts after transforming the given road network $\mathcal{N} = (\mathcal{I}, \mathcal{R})$ into the road graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as explained in Section 2.1. For all the nodes $\mathcal{V}$ in $\mathcal{G}$, the LD (lines 3–4), and the HDD with density parent/child nodes (lines 5–19) are computed. We assume that the feature values are in Gaussian distribution[2] and define the distance measure for computing LD and HDD based on the Gaussian similarity. Equation 8 formulates the Gaussian similarity between two linked nodes v_i and v_j, where $\sigma^2(v) = \frac{1}{n_v} \times \sum_{i=1}^{n_v} (v_i.f - \mu^v)^2$ is the variance of node feature values with respect to the node mean μ^v. It is a direct similarity with path length[3] 1.

$$gsim^1(v_i, v_j) = \exp\left(\frac{-\left(v_i.f - v_j.f\right)^2}{2 \times \sigma^2(v)}\right) \tag{8}$$

[2] In [14], the authors have used the Gaussian function in road networks, and we follow them.

[3] It refers to the number of links in the path connecting the two nodes.

Equation 9 shows the similarity with path length 2 where we multiply the $gsim^1(.)$ of intermediate links together for each different path between v_i and v_j and get the average of all such paths. As the value of $gsim^1(.)$ ranges from 0 to 1, its product of intermediate links also lies in the same range, and thus it also follows to $gsim^2(.)$. This equation is generalized for path length n in Equation 10.

$$gsim^2(v_i, v_j) = \frac{1}{|\mathcal{V}^2_{(v_i,v_j)}|} \times \left(\sum_{v_k \in \mathcal{V}_{(v_i,v_j)}} \left(gsim^1(v_i, v_k) \times gsim^1(v_k, v_j)\right)\right) \tag{9}$$

The final similarity measure is defined in Equation 11, by considering all the possible path lengths up to n, where we weight the similarity terms with the harmonic series members and divide their summation by the harmonic series. Generally the shorter paths between two nodes define their associativity (or relationship) strength more accurately than the longer paths. The rationality behind using the harmonic series to define the aggregated similarity is to make the effect of shorter paths more than longer paths, proportional to the path length. All the measures $gsim^1(v_i, v_j)$, $gsim^2(v_i, v_j)$, ..., $gsim^n(v_i, v_j)$, range between 0 and 1, and so does the $sim(v_i, v_j)$.

$$sim(v_i, v_j) = \frac{gsim^1(v_i, v_j) + \frac{gsim^2(v_i,v_j)}{2} + \ldots + \frac{gsim^n(v_i,v_j)}{n}}{1 + \frac{1}{2} + \ldots + \frac{1}{n}} \tag{11}$$

Based on this, the distance between a pair of nodes (v_i, v_j) is defined in Equation 12, which again makes it range between 0 and 1.

$$dist(v_i, v_j) = 1 - sim(v_i, v_j) \tag{12}$$

All those nodes having their HDD value $\delta^g(v_i)$ as the maximum of all distances between a pair of linked nodes $\max_{\forall v_k, v_l, v_k \Leftrightarrow v_l} \{dist(v_k, v_l)\}$ are designated as a DPN (line 17). For each DPN, a search is then started for the density children, which are combined with the DPNs to form a DGC (lines 20–29). This component is grown further by looking into the density children of the children of each DPN, and so on, until they return null. Thus a DGC is the largest component that could be grown from a DPN by exploring the density children. The number of DGCs in $\mathcal{G}$ is equal to the number of DPNs, $|D| = |\mathcal{V}^d|$, and the union of all the DGCs equals to the whole graph, $\{\cup_{\forall i} D_i\} = \mathcal{G}$. These DGCs are finally accepted as the different partitions of the road graph $\mathcal{G}$ (lines 30–31).

3.3. Determining Distance Threshold

As mentioned earlier, the main drawback of DBSCAN is the requirement of predetermined constants, ϵ and $minpts$. The cluster formation is highly sensitive to these parameters; a bad value for these parameters will lead to poor results. In `FaDPa`, one of our objectives is to make the algorithm more robust

$$gsim^n(v_i, v_j) = \frac{1}{|\mathcal{V}^n_{(v_i,v_j)}|} \times \left(\sum_{\langle v_{k1}, v_{k2}, \ldots, v_{k(n-1)} \rangle \in \mathcal{V}^n_{(v_i,v_j)}} \left(gsim^1(v_i, v_{k1}) \times gsim^1(v_{k1}, v_{k2}) \times \ldots \times gsim^1(v_{k(n-1),v_j}) \right) \right) \quad (10)$$

against these pre-determined parameters. We have only one constant, which is the distance threshold ϵ^d used in Equation 5. We consider this as a vector $\langle \epsilon^d_1, \epsilon^d_2, \ldots, \epsilon^d_{n_v} \rangle$ of dimension n_v, instead of a single constant value, where each ϵ^d_i corresponds to the distance threshold for node v_i. The value of ϵ^d_i is computed by looking into the neighborhood of v_i locally using Equation 13, where $\mathcal{V}^1_{(v_i)}$ denotes the set of nodes directly linked to v_i (with path length 1).

$$\epsilon^d_i = 1 - \frac{1}{|\mathcal{V}^1_{(v_i)}|} \times \sum_{v_j \in \mathcal{V}^1_{(v_i)}} sim(v_i, v_j) \quad (13)$$

3.4. FaDPa+: Reducing the Number of Partitions Further

In a graph where nodes are linked among themselves, each node is exposed only to its neighboring nodes. While computing the DPNs in $\mathcal{G}$ in Section 3.2, the *density parents* and *density children* relationships are established by looking into only the neighboring nodes. The DPNs obtained in this manner are based on the local connections (not on the complete node set globally). This leads to a large number of DPNs locally, and in turn a large number of DGCs. But in real situations, we may sometimes need to cluster the graph into a small number of partitions to know the global partitioning pattern. For example, in our experimental dataset $\mathtt{M}_2$ that has a graph of 53,494 nodes, the number of partitions produced by `FaDPa` is 22670; it generally depends on the number of nodes, links, and their distance weights. This number is still large. A manual analysis of these partitions would be very difficult, and the user may want to have far fewer partitions numbering less than 100 or even 10.

To further reduce the number of partitions generated by `FaDPa` as per the user requirements, we propose an extended algorithm named `FaDPa+` (shown in Algorithm 2). In this extension, the DPNs obtained from $\mathcal{G}$ by `FaDPa` are used to construct a DPG $\mathcal{G}^d = \left(\mathcal{V}^d, \mathcal{E}^d, \mathcal{W}^d\right)$ (defined in Definition 16). The new graph $\mathcal{G}^d$ becomes a condensed form of the original graph $\mathcal{G}$ where some local information is merged together. Considering $\mathcal{G}^d$ as the main graph now, the DPNs are further obtained using `FaDPa`. This time the number of DPNs would reduce further, and so would the number of partitions. These steps of constructing the DPG and identifying the DPNs are repeated alternatively until the number of DPNs becomes lower than the predefined number of partitions ϵ_p (lines 2–4). Thereafter the DGCs are obtained in the same way as explained earlier and accepted as the different partitions of the road graph (line 5).

4. FaDSPa: Fast Density and Spectral based Partitioning

`FaDPa+`, proposed in the previous section, is a complete road network partitioning algorithm in itself. It grows the clusters in arbitrary shapes by first identifying the dense components, and is able to work efficiently. In contrast, spectral

Algorithm 1: `FaDPa` (Road graph $\mathcal{G}$, Distance threshold ϵ^d)

1 $\mathcal{V}^d \leftarrow$ instantiate an empty set of DPNs;
2 *stack* $\leftarrow$ initialize a stack;
`// compute LD`
3 **for** $i \leftarrow 1$ **to** (n_r) **do**
4 $\quad \rho^g(v_i) = \sum_{\forall v_j \in neigh(v_i)} \chi(dist(v_i, v_j) - \epsilon^d)$;
`// compute HDD, and density parent/child nodes`
5 **for** $i \leftarrow 1$ **to** (n_r) **do**
6 $\quad \delta^g(v_i) \leftarrow -1$; `// initialize with null`
7 $\quad$ **forall the** $v_j \in neigh(v_i)$ **do**
8 $\quad\quad$ **if** $\rho^g(v_j) > \rho^g(v_i)$ **then**
9 $\quad\quad\quad$ **if** $\delta^g(v_i) = -1$ **then**
$\quad\quad\quad\quad$ `// assign distance`
10 $\quad\quad\quad\quad$ $\delta^g(v_i) \leftarrow dist(v_i, v_j)$;
11 $\quad\quad\quad\quad$ *parentnode* $\leftarrow v_j$;
12 $\quad\quad\quad$ **else if** $\delta^g(v_i) > dist(v_i, v_j)$ **then**
$\quad\quad\quad\quad$ `// overwrite HDD with the minimum distance`
13 $\quad\quad\quad\quad$ $\delta^g(v_i) \leftarrow dist(v_i, v_j)$;
14 $\quad\quad\quad\quad$ update, *parentnode* $\leftarrow v_j$;
15 $\quad$ **if** $\delta^g(v_i) = -1$ **then**
$\quad\quad$ `// Equation 7`
16 $\quad\quad$ $\delta^g(v_i) = \max_{\forall v_k, v_l, v_k \Leftrightarrow v_l} \{dist(v_k, v_l)\}$;
17 $\quad\quad$ $\mathcal{V}^d \leftarrow \mathcal{V}^d \cup \{v_i\}$; `//` v_i `found as DPN` ς_i
18 $\quad$ **else**
19 $\quad\quad$ Set v_i as child of *parentnode*, and *parentnode* as parent of v_i;
`// extract DGCs`
20 $D \leftarrow$ instantiate an empty set of DGCs;
21 **forall the** $\varsigma_i \in \mathcal{V}^d$ **do**
22 $\quad$ push ς_i into *stack*;
23 $\quad$ $d \leftarrow \{\phi\}$; `// instantiate an empty DGC`
24 $\quad$ **while** *stack is not empty* **do**
25 $\quad\quad$ *node* $\leftarrow$ pop out from *stack*;
26 $\quad\quad$ $d \leftarrow d \cup \{node\}$; `// add density-similar nodes to the DGC`
27 $\quad\quad$ **forall the** *childnode* $\in$ *child*(*node*) **do**
28 $\quad\quad\quad$ push *childnode* into *stack*;
29 $\quad$ $D \leftarrow D \cup \{d\}$;
`// extract partitions from DGCs`
30 $P \leftarrow$ extract partitions from D;
31 **return** P;

Algorithm 2: FaDPa+ (Road graph $\mathcal{G}$, Distance threshold ϵ^d, Number of partitions threshold ϵ_p)

```
1 G^d = (V^d, E^d, W^d) ← G = (V, E, W);
2 while |V^d| > ε_p do
3 |   partition set P ← FaDPa(G^d, ε^d);
4 |   G^d ← construct DPG from P;
5 return P;
```

clustering methods have been a major focus in the literature due to their ability to produce high quality results. Due to its high computational complexity, spectral clustering is often not used directly in large-scale data mining problems. However, attempts are being made to improve the efficiency of spectral clustering [39]. In this section, we propose FaDSPa (pronounced as *fad-spaa* and shown in Algorithm 3) as an efficient as well as effective road network partitioning algorithm that employs both density-based (FaDPa) (lines 2–4) and spectral-based (α-Cut) (line 5) theories.

Algorithm 3: FaDSPa (Road graph $\mathcal{G}$, Distance threshold ϵ^d, Compression threshold ϵ_c, Number of desired partitions k)

```
1 G^d = (V^d, E^d, W^d) ← G = (V, E, W);
  // density based clustering
2 while |V^d| > ε_c do
3 |   partition set P ← FaDPa(G^d, ε^d) ; // Algorithm 1
4 |   G^d ← construct DPG from P;
  // spectral based clustering
5 partition set P ← α-Cut Partitioning(G^d, k) ;
  // Algorithm 4
6 return P;
```

4.1. Mining DPG

FaDSPa starts by mining a road DPG $\mathcal{G}^d = \left(\mathcal{V}^d, \mathcal{E}^d, \mathcal{W}^d\right)$ from the road graph $\mathcal{G}$. It uses FaDPa+ to mine this DPG, in which ϵ_c is a compression threshold that determines the number of DPNs ($|\mathcal{V}^d|$). The value of ϵ_c is pre-defined depending on the available computing resources and the time that we can afford to spend in order to obtain good partitioning results. FaDPa+ compresses the graph until $|\mathcal{V}^d|$ becomes lower than or equal to ϵ_c. $\mathcal{G}$ is normally a sparse graph in nature. $\mathcal{G}^d$ is mined by identifying the dense components in $\mathcal{G}$ in arbitrary shapes, which reduces the sparsity of the graph as well as the overhead in dealing with that sparsity. The resulting graph $\mathcal{G}^d$ becomes a condensed form of the road graph $\mathcal{G}$, which is much smaller in order. As the level of compression of $\mathcal{G}$ is controlled by ϵ_c, there exist two extremes. At one end, ϵ_c could be set to $|\mathcal{V}|$, and on the other end, it could be the number of required partitions k. The first case makes it FaDPa+, whereas the second case makes it the α-Cut spectral clustering algorithm. Thus FaDSPa provides a good balance of FaDPa+ and α-Cut, and is a generalization of these two algorithms. After mining the DPG, a preliminary level of grouping of road segments has already happened in the form of DGCs in the DPG ($\mathcal{G}^d$) in a bottom-up manner. Thereafter the spectral based partitioning algorithm α-Cut is applied on the compressed graph $\mathcal{G}^d$, instead of the large graph $\mathcal{G}$, in a top-down manner.

4.2. Spectral Clustering for DPG Partitioning

Spectral clustering treats clustering as a graph partitioning problem. Among the existing graph cuts, normalized cut has been found to be comparatively effective for graph partitioning [29, 14]. Its objective function $\min_{\mathcal{P}} \sum_{i=0}^{k} \frac{W(\mathcal{P}_i, \overline{\mathcal{P}_i})}{W(\mathcal{P}_i, \mathcal{P})}$ is a minimization of the normalized summation of the cross-partition weighted links, where the normalization is done by all the weighted links having at least one end in the corresponding partition. Both the numerator and denominator take into account just the weighted links, and no consideration is made for the node groupings (or node counts) inside the resulting partitions. The links in our road graph are established only if they are adjacent in the road network, and thus the DPG links too are based on adjacency relationships. To partition the graph based on both weighted links and node counts in resulting partitions, in the next section we present the k-way graph cut developed in our recent work [3]. Instead of repeated bipartitioning of the whole graph, it produces $k'(> k)$ partitions in just a single iteration, and then applies repeated partitioning to produce k partitions, which significantly improves its efficiency.

4.3. The k-way α-Cut

For a given weighted graph, which in our case is the DPG[4] $\mathcal{G}^d$, let us suppose its DPN set is partitioned into k disjoint subsets or clusters as $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k\}$. The *adjacency matrix* of $\mathcal{G}^d$ is denoted by A, the *degree matrix* is denoted by D, which is a diagonal matrix having row sums of A at the diagonal as shown in Equation 14, and the *Laplacian matrix* $(D - A)$ is denoted by L.

$$D = \begin{pmatrix} \sum_{i=1}^{n_\varsigma} a_{1i} & 0 & \cdots & 0 \\ 0 & \sum_{i=1}^{n_\varsigma} a_{2i} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sum_{i=1}^{n_\varsigma} a_{n_\varsigma i} \end{pmatrix} \tag{14}$$

A function $W(\mathcal{P}_i, \mathcal{P}_j)$ is defined in Equation 15 as the sum of weights associated with all the links having their DPN at one end in $\mathcal{P}_i$ and the DPN at the other end in $\mathcal{P}_j$.

$$W(\mathcal{P}_i, \mathcal{P}_j) = \sum_{\varepsilon_r \in \{links(\mathcal{P}_i, \mathcal{P}_j)\}} \omega_r = \sum_{\varsigma_p \in \mathcal{P}_i, \varsigma_q \in \mathcal{P}_j} A(p, q) \tag{15}$$

[4] In this section, the DPG can be treated just like a weighted graph, and the terms DPG and DPN can be read synonymously as graph and node respectively for the application of α-Cut in graph partitioning.

Definition 17: (**Cut**) For a given partition set $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k\}$ the *cut* of a partition $\mathcal{P}_i$ is defined as the summation of weights associated with all the links having their DPNs at one end in $\mathcal{P}_i$ and DPNs at other end in any partition other than $\mathcal{P}_i$, i.e., $W(\mathcal{P}_i, \overline{\mathcal{P}_i})$. ■

Definition 18: (**Association**) For a given partition set $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k\}$ the *association* of a partition $\mathcal{P}_i$ is defined as the summation of weights associated with all the links having DPNs at both ends in $\mathcal{P}_i$, i.e. $W(\mathcal{P}_i, \mathcal{P}_i)$. ■

The *cut* value of a partition $\mathcal{P}_i$ gives a measure of connectivity strength between $\mathcal{P}_i$ and the rest of the partitions, and thus quantifies the loss incurred in cutting those link connections while partitioning the graph. When this value is divided by the number of DPNs in $\mathcal{P}_i$, it gives the average contribution of each DPN in the overall cut of $\mathcal{P}_i$. It represents the inter-partition similarity. Similarly, the *association* value of a partition $\mathcal{P}_i$ gives a measure of connectivity strength within $\mathcal{P}_i$ that binds it as a unit, and thus quantifies the retained association of $\mathcal{P}_i$ after partitioning the graph. When this value is divided by the number of DPNs in $\mathcal{P}_i$, it gives the average contribution of each DPN in the overall association of $\mathcal{P}_i$. It represents the intra-partition similarity. A good partitioning is achieved by minimizing the summation of average cut values and simultaneously maximizing the summation of average association values of each partition [29]. However, optimizing any one of these objectives does not guarantee the other. One possible approach is that of normalized cut [29, 14]. It minimizes inter-partition similarity and maximizes intra-partition similarity simultaneously. But the optimization is based on normalized values of cut and association, where the normalization considers the link connectivities between nodes, instead of the nodes directly. It does not guarantee the optimization of their average cut and association.

Our k-way graph cut called α-Cut aims to achieve a well balanced optimization of average cut and average association. We optimize the objective function $\min_{\mathcal{P}} \alpha\text{-}Cut(\mathcal{P})$, where $\alpha\text{-}Cut(\mathcal{P})$ is shown in Equation 16.

$$\alpha\text{-}Cut(\mathcal{P}) = \sum_{i=1}^{k} \left(\alpha \times \frac{W(\mathcal{P}_i, \overline{\mathcal{P}_i})}{|\mathcal{P}_i|} - (1-\alpha) \times \frac{W(\mathcal{P}_i, \mathcal{P}_i)}{|\mathcal{P}_i|} \right) \quad (16)$$

It minimizes a combination of two components, which separately are the minimization of the average *cut* representing the inter-partition similarity, and the maximization of the average *association* representing the intra-partition similarity. The $\alpha \in [0, 1]$ acts as a balance between the two components. Its value is crucial to obtain the best possible optimized partitions. An advantage of α-Cut over normalized cut is that α-Cut normalizes the cut and association by the partition size, whereas normalized cut normalizes the cut by the association.

4.4. Determining $\boldsymbol{\alpha}$ in $\boldsymbol{\alpha}$-Cut

Instead of considering α as a single constant value for all the partitions, we consider it as a vector $\alpha = \langle \alpha_1, \alpha_2, \ldots, \alpha_k \rangle$, where each α_i corresponds to the partition $\mathcal{P}_i$. The advantage in considering it as a vector over a single scalar value is its non-uniformly defined value depending on the nature of the respective partition. We consider this factor α_i as the portion of the connectivity weight contributed by $\mathcal{P}_i$ in the whole DPG (including intra-connections as well as inter-connections), and define it as the ratio of the summation of its link connection weights to the summation of all link connection weights in the DPG, i.e., $\alpha_i = \frac{W(\mathcal{P}_i, \mathcal{V}^d)}{W(\mathcal{V}^d, \mathcal{V}^d)}$. Its value ranges from 0 to 1. In contrast, $(1 - \alpha_i)$ gives the portion of the connectivity weight contributed by all partitions other than $\mathcal{P}_i$. Putting this value of α_i in Equation 16, α-Cut simplifies as shown in Equation 17.

$$\begin{aligned}
\alpha\text{-}Cut(\mathcal{P}) &= \sum_{i=1}^{k} \left(\frac{W(\mathcal{P}_i, \mathcal{V}^d)}{W(\mathcal{V}^d, \mathcal{V}^d)} \times \frac{W(\mathcal{P}_i, \overline{\mathcal{P}_i})}{|\mathcal{P}_i|} - \frac{W(\mathcal{P}_i, \mathcal{P}_i)}{|\mathcal{P}_i|} \right. \\
&\quad \left. + \frac{W(\mathcal{P}_i, \mathcal{V}^d)}{W(\mathcal{V}^d, \mathcal{V}^d)} \times \frac{W(\mathcal{P}_i, \mathcal{P}_i)}{|\mathcal{P}_i|} \right) \\
&= \sum_{i=1}^{k} \left(\frac{W(\mathcal{P}_i, \mathcal{V}^d)}{W(\mathcal{V}^d, \mathcal{V}^d)} \times \left(\frac{W(\mathcal{P}_i, \overline{\mathcal{P}_i})}{|\mathcal{P}_i|} + \frac{W(\mathcal{P}_i, \mathcal{P}_i)}{|\mathcal{P}_i|} \right) \right. \\
&\quad \left. - \frac{W(\mathcal{P}_i, \mathcal{P}_i)}{|\mathcal{P}_i|} \right) \\
&= \sum_{i=1}^{k} \left(\frac{W(\mathcal{P}_i, \mathcal{V}^d)}{W(\mathcal{V}^d, \mathcal{V}^d)} \times \frac{W(\mathcal{P}_i, \mathcal{V}^d)}{|\mathcal{P}_i|} - \frac{W(\mathcal{P}_i, \mathcal{P}_i)}{|\mathcal{P}_i|} \right)
\end{aligned} \quad (17)$$

Like normalized cut [29], the problem of achieving a partitioning configuration that minimizes this cost is an NP-complete problem. To solve it in a time-bound and computationally efficient manner, we follow a spectral clustering approach described in the following subsection.

4.5. Spectral Clustering Approach to $\boldsymbol{\alpha}$-Cut

If $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k\}$ is the set of k disjoint partitions of $\mathcal{G}^d$, let $\mathbf{1} \in \mathbb{R}^{n_s}$ be a vector with each of its values as 1, and $c_i \in \mathbb{R}^{n_s}$ be the cluster indicator vector of $\mathcal{P}_i$ such that its jth value $c_i(j) = 1$, if $\varsigma_j \in \mathcal{P}_i$, and $c_i(j) = 0$ otherwise, as shown in Equation 18.

$$c_i(j) = \begin{cases} 1, & \text{if } (\varsigma_j \in \mathcal{P}_i) \\ 0, & \text{if } (\varsigma_j \notin \mathcal{P}_i) \end{cases} \quad (18)$$

The spectral clustering approach to minimize the cost of α-Cut partitioning follows a relaxed approach based on eigenvectors and eigenvalues. The relaxation lies in the cluster indicator vectors, which are allowed to take on any real value, instead of restricting them only to discrete values. Using the cluster indicator vectors, the α-Cut formulation can be simplified by substituting $W(\mathcal{P}_i, \mathcal{V}_d)$ by $\mathbf{1}^T D c_i$, $W(\mathcal{P}_i, \mathcal{P}_i)$ by $c_i^T A c_i$, $W(\mathcal{V}_d, \mathcal{V}_d)$ by $\mathbf{1}^T D \mathbf{1}$, and $|\mathcal{P}_i|$ by $c_i^T c_i$ in Equation 17. The simplification steps are shown in Equation 19.

Algorithm 4: α-Cut Partitioning (DPG $\mathcal{G}^d$, number of desired partitions k)

1 $A \leftarrow$ adjacency matrix of $\mathcal{G}^d$;
2 $D \leftarrow$ degree matrix of $\mathcal{G}^d$;
// repeated partitioning to obtain k partitions
3 **repeat**
4 $M \leftarrow \left(\frac{(\mathbf{1}^T D)^T(\mathbf{1}^T D)}{\mathbf{1}^T D\mathbf{1}} - A\right)$; // get the α-Cut matrix
5 $\bigcup_{i=1}^{n_\varsigma}\{(y_i, \lambda_i)\} \leftarrow$ get eigenvector and eigenvalue pairs of M;
6 sort eigenvalues λ_i to have $\lambda_{n_\varsigma} \leq \lambda_{n_\varsigma - 1} \leq \ldots \leq \lambda_1$;
7 select $\{\lambda_{n_\varsigma}, \lambda_{n_\varsigma-1}, \ldots, \lambda_{n_\varsigma-k+1}\}$ eigenvalues and corresponding eigenvectors $\{y_{n_\varsigma}, y_{n_\varsigma-1}, \ldots, y_{n_\varsigma-k+1}\}$;
8 generate matrix $Y_{n_\varsigma \times k} = \begin{pmatrix} y_1 & y_2 & \ldots & y_k \end{pmatrix}$;
9 $Z \leftarrow$ row normalize Y;
10 $\left\{z_1, z_2, \ldots, z_{n_\varsigma}\right\} \leftarrow$ get row vectors of Z;
11 $C = \{C_1, C_2, \ldots, C_k\} \leftarrow k\text{-means}\left(\left\{z_1, z_2, \ldots, z_{n_\varsigma}\right\}, k\right)$;
12 $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_{k'}\} \leftarrow$ get disjoint partitions from C; // resulting set of partitions
13 **if** *k' is not equal to k* **then**
// construct a graph from the partitions and consider this as the new graph for partitioning
14 $n_\varsigma \leftarrow k'$;
15 $\mathcal{G}^p \leftarrow$ construct partition graph from $\mathcal{P}$;
16 $A'_{n_\varsigma \times n_\varsigma} \leftarrow$ adjacency matrix of $\mathcal{G}^p$;
17 $D'_{n_\varsigma \times n_\varsigma} \leftarrow$ degree matrix of $\mathcal{G}^p$;
18 $A \leftarrow A'$;
19 $D \leftarrow D'$;
20 **until** *number of partitions in $\mathcal{P}$ equals to k*;
21 **return** $\mathcal{P}$; // return the partitions when their number equals to k

$$
\begin{aligned}
\alpha\text{-}Cut(\mathcal{P}) &= \sum_{i=1}^{k}\left(\frac{\mathbf{1}^T D c_i}{\mathbf{1}^T D\mathbf{1}} \times \frac{\mathbf{1}^T D c_i}{c_i^T c_i} - \frac{c_i^T A c_i}{c_i^T c_i}\right) \\
&= \sum_{i=1}^{k}\frac{1}{c_i^T c_i} \times \left(\frac{\left(\mathbf{1}^T D c_i\right)^2}{\mathbf{1}^T D\mathbf{1}} - c_i^T A c_i\right) \\
&= \sum_{i=1}^{k}\frac{1}{c_i^T c_i} \times \left(\frac{c_i^T\left(\mathbf{1}^T D\right)^T\left(\mathbf{1}^T D\right)c_i}{\mathbf{1}^T D\mathbf{1}} - c_i^T A c_i\right) \\
&= \sum_{i=1}^{k}\frac{1}{c_i^T c_i} \times c_i^T\left(\frac{\left(\mathbf{1}^T D\right)^T\left(\mathbf{1}^T D\right)}{\mathbf{1}^T D\mathbf{1}} - A\right)c_i \\
&= \sum_{i=1}^{k}\frac{c_i^T M c_i}{c_i^T c_i} \\
&\text{where } M = \left(\frac{\left(\mathbf{1}^T D\right)^T\left(\mathbf{1}^T D\right)}{\mathbf{1}^T D\mathbf{1}} - A\right)
\end{aligned}
\tag{19}
$$

The derived matrix M is called the α-Cut matrix for $\alpha_i = \frac{W(\mathcal{P}_i, \mathcal{V}^d)}{W(\mathcal{V}^d, \mathcal{V}^d)}$, and the spectral clustering algorithm works on this matrix. Equation 19 is further simplified as follows.

$$
\sum_{i=1}^{k}\frac{c_i^T M c_i}{\|c_i\|^2} = \sum_{i=1}^{k}\left(\frac{c_i}{|c_i|}\right)^T M\left(\frac{c_i}{|c_i|}\right) = \sum_{i=1}^{k} y_i^T M y_i
$$

where y_i is a unit vector in the direction of c_i, such that $y_i^T y_i = 1$. Hence the optimization function becomes

$$
\min_{\mathcal{P}} \sum_{i=1}^{k} y_i^T M y_i \text{ subject to } y_i^T y_i = 1 \tag{20}
$$

This is solved by setting its derivative with respect to y_i to zero and introducing a Lagrange multiplier λ_i for each $\mathcal{P}_i$ to incorporate the associated constraint [40], as shown in Equation 21.

$$
\begin{aligned}
\frac{\partial}{\partial y_i}\left(\sum_{i=1}^{k} y_i^T M y_i + \sum_{i=1}^{n}\lambda_i\left(1 - y_i^T y_i\right)\right) &= 0 \\
M y_i - \lambda_i y_i &= 0 \\
M y_i &= \lambda_i y_i
\end{aligned}
\tag{21}
$$

It implies that y_i is one of the eigenvectors of M corresponding to the eigenvalue λ_i, and $y_i^T M y_i = y_i^T \lambda_i y_i = \lambda_i$. As the objective is minimization, we select k smallest eigenvalues from the total of n_ς eigenvalues as $\lambda_{n_\varsigma} \leq \lambda_{n_\varsigma-1} \leq \cdots \leq \lambda_{n_\varsigma-k+1}$ and corresponding eigenvectors $y_{n_\varsigma}, y_{n_\varsigma-1}, \ldots, y_{n_\varsigma-k+1}$ which represent the relaxed cluster indicator vectors. Thus, it leads to Equation 22.

$$
\begin{aligned}
\min_{\mathcal{P}} \alpha\text{-}Cut(\mathcal{P}) &= y_{n_\varsigma}^T M y_{n_\varsigma} + \cdots + y_{n_\varsigma-k+1}^T M y_{n_\varsigma-k+1} \\
&= \lambda_{n_\varsigma} + \cdots + \lambda_{n_\varsigma-k+1}
\end{aligned}
\tag{22}
$$

Algorithm 4 presents the complete partitioning method, which starts with getting the adjacency and degree matrices in line 1 and 2. The steps 4–19 are repeatedly performed until the resulting number of partitions equals to k. The α-Cut matrix is computed from the adjacency and degree matrices in line 4 and eigen-decomposed in line 5. Lines 6–7 select the k smallest eigenvalues and corresponding eigenvectors. Ideally the indicator vectors should have only binary values, but the actually obtained indicator vectors are in fact the relaxed vectors and do not follow a binary pattern. Due to the lack of concrete information about clusters, it becomes another problem to separate the k clusters. We assume that the clusters are well-separated in the k-dimensional eigenspace, which is a general assumption in spectral clustering [40], and use the eigenvectors (or indicator vectors) to generate a matrix Y of $n_\varsigma \times k$ dimensions (line 8). It is then row-normalized using Equation 23 to have row-vectors z_i of unit length giving the final matrix Z (line 9). Each row-vector z_i represents a DPN ς_i. The set of row-vectors are used to cluster the DPNs by applying k-means to find a set of k clusters

$C = \{C_1, C_2, \ldots, C_k\}$ (lines 10–11), where each cluster C_i comprises one or more row vectors (DPNs) in Z. The DPNs inside each cluster are linked together as they exist in the DPG. Upon linking them, sometimes more than one connected component may be found inside a single cluster. As these multiple connected components within a single cluster are disjoint, they can not become part of the same partition. These connected components are extracted from each cluster to form disjoint partitions (line 12).

$$Y = \begin{pmatrix} y_{11} & y_{21} & \cdots & y_{k1} \\ y_{12} & y_{22} & \cdots & y_{k2} \\ | & | & & | \\ y_{1n} & y_{2n} & \cdots & y_{mn} \end{pmatrix} = \begin{pmatrix} — & z_1^T & — \\ — & z_2^T & — \\ & \vdots & \\ — & z_n^T & — \end{pmatrix} = Z \qquad (23)$$

$$\text{where, } z_i = \frac{1}{\sqrt{\sum_{j=1}^{k} y_{ji}^2}} (y_{1i}, y_{2i}, \ldots, y_{mi})^T$$

Depending on the data, the number of disjoint partitions may sometimes be large, which would yield the partition set from C as $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_{k'}\}$, where $k' \geq k$. These k' partitions may be accepted as the final result. However, if the requirement to have exactly k partitions is strict, Shi and Malik [29] described two approaches to achieve this, which are greedy pruning and global recursive bipartitioning. The greedy pruning approach iteratively merges the two nearest partitions optimizing the defined graph cut, until it results in a total of k partitions. In contrast, the global recursive bipartitioning approach generates a condensed graph where each partition forms a node and adjacent partitions are connected by weighted links with $W(\mathcal{P}_i, \mathcal{P}_j)$ as the weight, and is recursively bipartitioned until it results in a total of k partitions. For large k' values, the greedy pruning approach is computationally intensive. On the other hand, as the global recursive approach bipartitions each time, it would yield a balanced set of partitions only when k follows the pattern 2^i for any value of i. For example, if the value of k is 3, firstly the graph is bipartitioned to get 2 partitions, and then only one of them has to be bipartitioned to get a total of 3 partitions, whereas the other partition remains as it is. In this way it generates one large partition and two small partitions, where the large partition is approximately double in size than the small ones. Moreover, the selection of the large partition that is to be bipartitioned first, is either arbitrary or some additional condition has to be applied to decide this.

To avoid these complexities and make the method efficient, we follow a repeated partitioning approach where the interleaved steps of partitioning and constructing a new graph each time from the obtained partitions are repeated until we obtain exactly k partitions (lines 4–12). In each repetition, we construct a new graph from the set of partitions, by considering each partition as a node and their connectivity via the nodes belonging to them as links (lines 14–15). The feature value of the nodes (formed from the partitions obtained in the last iteration) is assigned as the average of feature values of all nodes belonging to the respective partitions (old partitions), based on which the link weights are assigned. In lines 16–17, we get its adjacency and degree matrices. These matrices are considered to compute the α-Cut matrix for the next iteration of partitioning. Finally the k partitions are returned at the end (line 21).

4.6. Computational Complexity

The algorithm `FaDSPa` comprises successive applications of `FaDPa` to mine the DPG of desired order, followed by α-Cut. The computational complexity of `FaDPa` is $O(n^2)$ for computation of $\rho^g(v_i)$ and $\delta^g(v_i)$, after which the partitions are extracted using a stack. Thus the overall computational complexity of `FaDPa` becomes $O(n^2)$, which is applied multiple times but still much less than n, thereby making it $\approx O(n^2)$. In α-Cut, the eigen-decomposition task is done in $\mathbf{O}(n^3)$ time in general and $\mathbf{O}(n^2)$ time for sparse matrices. The application of k-means on row-vectors to find the clusters costs $\mathbf{O}(tnk^2)$, where t is the number of iterations required to reach the convergence. In these costs, $n = n_\varsigma$ when the spectral clustering is applied on the DPG, and $n = n_r$ when it is applied directly on the road graph.

4.7. Relation with Modularity

Definition 19: (**Modularity**) The modularity of a set of graph partitions $Q(\mathcal{P})$ [35] is defined as the difference between the observed and expected fraction of links within a partition, and is formulated as Equation 24. ■

$$Q(\mathcal{P}) = \sum_{i=1}^{k} \left[\frac{W(\mathcal{P}_i, \mathcal{P}_i)}{W(\mathcal{V}, \mathcal{V})} - \left(\frac{W(\mathcal{P}_i, \mathcal{V})}{W(\mathcal{V}, \mathcal{V})} \right)^2 \right] \qquad (24)$$

Larger modularity values are correlated with better graph partitioning. To maximize modularity while partitioning a graph, in [35] the authors presented a spectral clustering solution. They showed that the partitioning can be obtained using the k largest eigenvalues and corresponding eigenvectors obtained after eigen-decomposition of a derived matrix called Q-Laplacian [35]. This matrix actually equals to the negative of our α-Cut matrix derived in Equation 19. As we obtain the partitioning by selecting the k smallest eigenvalues and corresponding eigenvectors, both the techniques result into the same set of eigenvalues and eigenvectors, and thus the same partitioning. It means that the minimization of α-Cut approximately maximizes the modularity.

5. Experimental Evaluation

Although there exist many works on general graph partitioning, we compare our results to a recent work [14] on the same problem, for a specific comparison. In addition we also compare with the results obtained by replacing α-Cut by normalized cut in `FaDSPa` to show the effectiveness of α-Cut. Section 5.3 presents our experiments on small road networks, where we compare the results obtained by α-Cut, normalized cut, `FaDSPa`, [14], and `FaDPa+`. Section 5.4 presents our experiments on the real SCATS dataset, where we compare the results obtained by the proposed `FaDSPa` and a modified version of `FaDSPa` (by replacing α-Cut by normalized cut in `FaDSPa`).

Section 5.5 presents our experiments on large networks, where we show the performance of FaDPa+ and FaDSPa for varying values of the compression threshold ϵ_c to understand the trade-off between efficiency and accuracy.

5.1. Datasets

We perform experiments on five datasets of different sizes including both real data and synthetic data generated on real road networks. Table 2 shows the statistics of all these datasets. The real data (M_s) is recorded by the Sydney Coordinated Adaptive Traffic System (SCATS)[5] from the Melbourne road networks provided to us by VicRoads[6]. This dataset is an accumulation of the traffic records of individual road segments for each signal cycle from 1st Jan 2011 to 1st Jan 2013. The considered Melbourne network consists of 7245 road segments and 2928 intersection points, where the traffic measures are logged by the installed sensors, respective to each lane of road segments at the SCATS sites. The traffic measures include traffic volume (number of vehicles crossing a road segment during the green time) and degree of saturation (the ratio of the effectively used green time to the total available green time). In this dataset we consider the degree of saturation as feature value of the road segments, as it gives an indication of the traffic density. The degree of saturation measure for each road segment is computed by taking the average of this measure of all the different lanes that are part of the referred road segment.

The other datasets include synthetic data generated on real small and large road networks. The traffic on the small network (D_1), shared by the authors of [14], is based on a micro-simulation performed for 4 hours at 120 time intervals of 2 minutes. At each time point t, the traffic density on each road segment is computed in terms of number of vehicles per meter. For large road network, we consider the city of Melbourne with three sets of data, M_1, M_2, and M_3. M_1 is the road network of the 6.6 sq. miles CBD area consisting of 10,096 intersection points and 17,206 directed road segments. M_2 and M_3, larger than M_1, is the road network of the CBD and adjoining areas in a total of 31.5 and 42.03 sq. miles, consisting of 28,465 and 42,321 intersection points, and 53,494 and 79,487 directed road segments respectively. These road segments are obtained by considering all the two-way road segments as two different one-way road segments. The traffic data for the large networks is generated by a web-based[7] random road traffic generator MNTG [25, 26]. It populates vehicles on a selected real road network, which keep on moving for a time duration on the roads. We populate M_1, M_2, and M_3 by 25,246, 62,300, and 84,999 vehicles respectively, and obtain their trajectories for 100 continuous timestamps. The trajectories are sequences of 100 or less ⟨latitude,longitude⟩ pairs corresponding to vehicle positions at each timestamp. A self-designed program is used to map their positions to corresponding road segments, and compute the traffic density of road segments (in terms of vehicles/meter) at each point of time. While doing this, after each interval of 10 timestamps, each vehicle is considered as a different one and its updated position is recounted to compute the density. Thus it makes t range from 1 to 10, and in this work we experiment with $t = 1$. It is done to make the network more dense, and reflect the flow speed on corresponding road segments. The count is then divided by the road segment length to get the average traffic density in terms of vehicles per meter.

5.2. Evaluation Metrics

The partitioning framework is evaluated using metrics that quantify the quality of the results from different perspectives. The problem defined in Section 2.2 intends to achieve four different conditions. As we obtain results in the form of disjoint and connected road network partitions (connected components), **C.1** and **C.2** are automatically fulfilled. **C.3** which enforces intra-partition homogeneity is evaluated by the *intra* metric defined in Equation 25. For each partition, it computes the intra-partition distance as the average absolute distance between the pair of nodes, and then takes the average of that computed for all the partitions. Lower values of *intra* indicate better partitioning.

$$Intra(\mathcal{P}) = \frac{1}{|\mathcal{P}|} \times \sum_{\mathcal{P}_i \in \mathcal{P}} \frac{\sum\limits_{\substack{v_p, v_q \in \mathcal{P}_i \\ p \neq q}} \text{abs}(v_p.f - v_q.f)}{|\mathcal{P}_i| \cdot (|\mathcal{P}_i| - 1)} \tag{25}$$

C.4 which enforces inter-partition heterogeneity is evaluated by the *inter* metric defined in Equation 26, where $\mathcal{P}_i \overset{\text{adj}}{\longleftrightarrow} \mathcal{P}_j$ denotes the set of adjacency relationships[8]. It is the average of inter-partition distances between each pair of spatially adjacent partitions, where the inter-partition distance is the average absolute distance between nodes from the respective pair of adjacent partitions. Higher values of *inter* indicate better partitioning.

$$Inter(\mathcal{P}) = \frac{1}{\left|\mathcal{P}_i \overset{\text{adj}}{\longleftrightarrow} \mathcal{P}_j\right|} \times \sum_{\substack{\mathcal{P}_i, \mathcal{P}_j \in \mathcal{P} \\ \mathcal{P}_i \overset{\text{adj}}{\longleftrightarrow} \mathcal{P}_j}} \frac{\sum\limits_{v_p \in \mathcal{P}_i} \sum\limits_{v_q \in \mathcal{P}_j} \text{abs}(v_p.f - v_q.f)}{|\mathcal{P}_i| \cdot \left|\mathcal{P}_j\right|} \tag{26}$$

We also evaluate the overall partitioning using average NcutSilhouette (ANS) measure defined in [14] especially for partition evaluation. It is derived from the standard Silhouette measure used for cluster evaluation. NS between

[5] SCATS is a fully adaptive urban traffic control system developed in Australia in 1970. It manages the signal phases (cycle times, phase splits and offsets) of the traffic signals dynamically in real-time, based on the traffic data collected by the vehicle sensors (inductive loops) installed within road pavements of each traffic signal.

[6] https://www.vicroads.vic.gov.au

[7] It can be accessed through `http://mntg.cs.umn.edu/tg/`

[8] A pair of partitions $\mathcal{P}_i$ and $\mathcal{P}_j$ are said to be *adjacent*, if there exists at least one link connecting nodes v_p and v_q such that $v_p \in \mathcal{P}_i$ and $v_q \in \mathcal{P}_j$

Table 2: Dataset statistics

Dataset	Place	Area (sq ml)	# Road segments	# Intersection points
Real SCATS data on real road network				
M_s	Melbourne	627.5	7245	2928
Synthetic data generated on real road network				
D_1	Downtown San Francisco	2.5	420	237
M_1	CBD Melbourne	6.6	17,206	10,096
M_2	CBD(+) Melbourne	31.5	53,494	28,465
M_3	Melbourne	42.03	79,487	42,321

a pair of partitions $(\mathcal{P}_i, \mathcal{P}_j)$ is calculated using Equation 27, and the quality of each individual partition is evaluated using $NS(\mathcal{P}_i)$ defined in Equation 28, where $NSN(\mathcal{P}_i, \mathcal{P}_j) = \min\left\{NS(\mathcal{P}_i, \mathcal{P}_x) | \mathcal{P}_x \in neighbor(\mathcal{P}_j)\right\}$.

$$NS(\mathcal{P}_i, \mathcal{P}_j) = \frac{\sum_{v_p \in \mathcal{P}_i} \sum_{v_q \in \mathcal{P}_j} \left(v_p.f - v_q.f\right)^2}{|\mathcal{P}_i| \times \left|\mathcal{P}_j\right|} \quad (27)$$

$$NS(\mathcal{P}_i) = \frac{NS(\mathcal{P}_i, \mathcal{P}_i)}{NSN(\mathcal{P}_i, \mathcal{P}_j)} \quad (28)$$

Average NS (ANS) is computed as the average of $NS(\mathcal{P}_i)$ for all $\mathcal{P}_i \in \mathcal{P}$. A value less than 1 indicates a good partitioning, and lower values indicate better partitioning.

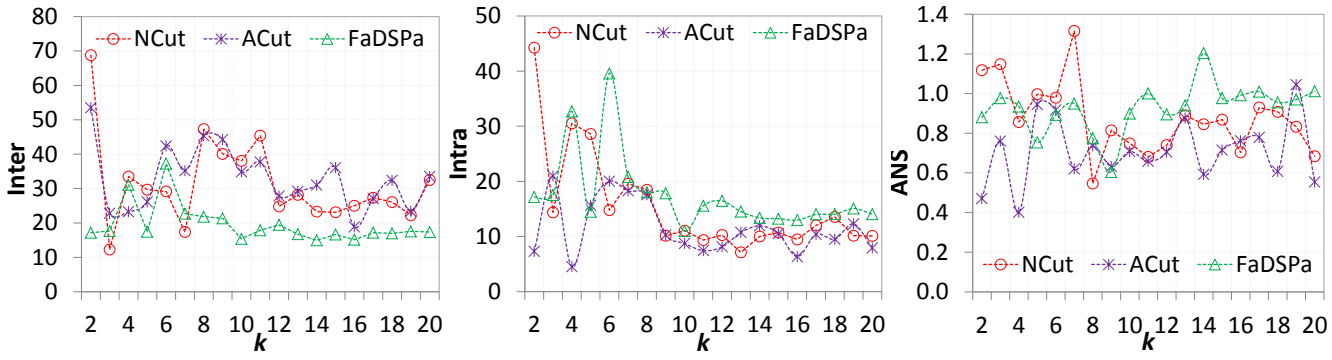
5.3. Experimental Results on Small Networks

We perform experiments on the small road network D_1 to compare the partitioning quality of our α-Cut, FaDPa+ and FaDSPa, with other state-of-the-art techniques (normalized cut and [14]) using performance evaluation metrics listed in Section 5.2, and demonstrate their effectiveness.

Quality comparison of NCut, α-Cut, and FaDSPa: We consider normalized cut as the baseline, and show our comparative results. Figure 5 shows the complete results obtained by α-Cut (ACut) and FaDSPa in comparison to normalized cut (NCut). We present the results in terms of *inter*, *intra* and ANS in Figures 5(a), 5(b) and 5(c) respectively, for the number of partitions k ranging from 2 to 20. The ANS measure quantifies the overall partitioning quality. In terms of ANS, ACut of our framework outperforms NCut for most of the values of k, whereas NCut outperforms FaDSPa for most of the values. This is expected because FaDSPa is of lower complexity and suitable for large networks. In terms of *inter*, which quantifies inter-partition heterogeneity, we observe that ACut performs similar to NCut. However, both of them outperform FaDSPa for all $k \geq 8$, and for $k \leq 7$ sometimes FaDSPa outperforms NCut and ACut. In terms of *intra*, which quantifies intra-partition homogeneity, ACut outperforms NCut for most of the values except $k = 3$, 6, 13, 14, and 19, whereas both of them outperform FaDSPa for all $k \geq 6$. These results of FaDSPa are obtained by setting the compression threshold ϵ_c to the number of DPNs obtained after the first round of FaDPa+, which is 109.

Experiments with FaDPa+: FaDPa+, as proposed in this paper, does not provide the option to input the value of k. It has a parameter called the number of partitions threshold ϵ_p. The algorithm merges the partitions on the basis of their local densities, until k is lower than or equal to ϵ_p for the first time. Thus its k can be any value closest to and lower than or equal to ϵ_p. In our experiment on this dataset it started with 420 nodes, and after the first round of FaDPa it gave 109 partitions with 14.61, 7.85, and 0.74, as their *inter*, *intra* and ANS measures respectively. After the second round, it gave 13 partitions with the values as 20.69, 16.11, and 1.00, respectively, and after the next round all the partitions were merged to a single partition. Looking into the ANS values, it shows that the quality of 109 partitions are better than the 13 partitions obtained after the second round. However, Figure 5 shows that the best clustering is obtained at lower values (e.g., $k = 5$, 6 and 8, by different methods). To know how FaDPa+ behaves for these lower k, we merged the density-closest partitions one by one, and found the best partitioning at $k = 5$. The performance metric values are found as 42.16, 16.53, and 0.68, respectively.

Summary of comparisons: The overall partitioning quality is evaluated by ANS, which considers both the inter-partition heterogeneity and intra-partition homogeneity simultaneously. Lower values indicate a better partitioning. In Figure 5(c), ACut is lower than NCut at most values of k, whereas FaDSPa is mostly above both of them. In [14], the authors used the ANS measure to learn the number of optimal partitions. They accept the value of k that leads to the ANS minimum as the optimal number of partitions, which in this case is 8 for NCut, 4 for ACut, and 9 for FaDSPa. We observe in the figure that the minimum of ACut is the lowest followed by NCut, and that of FaDSPa is the highest. It shows the accuracy of the partitioning task by these three methods. ACut performs the best, followed by NCut, and both of them are better than FaDSPa. Figure 6 and Table 3 show the obtained ANS measures by ACut, FaDSPa, FaDPa+, NCut, in comparison to the existing work [14]. Our methods ACut and FaDSPa are lower (and thus perform better) than [14]. In the methods, we determine the optimal value of k by repeatedly obtaining the results for a range of k and comparing their ANS measure. It can also be an application dependent issue, as a small k produces partitions of coarse granularity and this granularity becomes finer with an increasing k.



(a) Inter (b) Intra (c) ANS

Figure 5: Road graph and DPG partitioning results in small networks

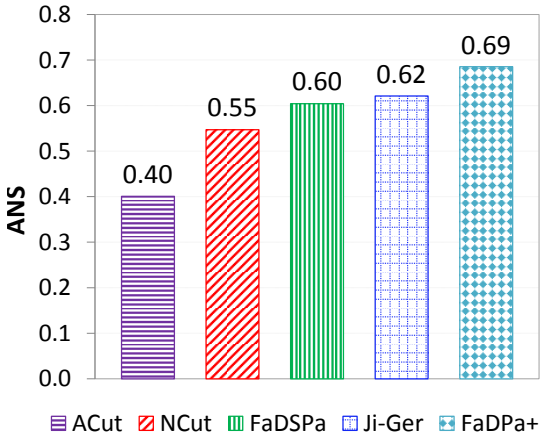


Figure 6: Overall comparison of partitioning results in small networks

Table 3: Overall quality of partitioning

	ACut	NCut	FaDSPa	Ji-Ger	FaDPa+
ANS	0.4009	0.5470	0.6041	0.6210	0.6853
k	4	8	9	3	5

Even though the results of FaDSPa do not look very impressive (in comparison to α-Cut or NCut), its advantage is that it can efficiently handle large networks while simultaneously maintaining the quality. The spectral based algorithms face time and space complexity issues, whereas the density based algorithms compromise the partitioning accuracy.

5.4. Experimental Results on Real Data

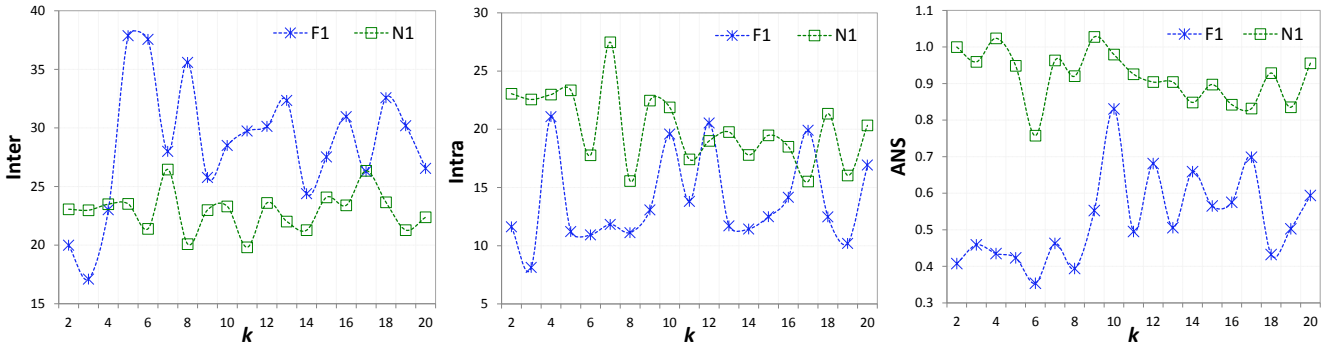
We perform experiments on real data to see the applicability of the proposed method in real environments. For this we consider the SCATS data M_s, described in Section 5.1. Through our experiments on this dataset, we show the results obtained by the proposed FaDSPa algorithm and also compare them with those obtained by replacing α-Cut by normalized cut in FaDSPa.

Different schemes of FaDSPa: Before going further, we explain the different schemes we have used to present our result insights. We applied the proposed method FaDSPa in different ways, which are denoted by F⟨*number*⟩. This stands for the method when FaDSPa is applied by repeatedly running FaDPa+ ⟨*number*⟩ number of times forming hierarchical groups, before passing the control to α-Cut. Thus F1 applies one *round*[9] of FaDPa+, and the generated DPG after that is treated by α-Cut to obtain the k partitions. F2, F3, F4, and F5 work similarly with two, three, four and five rounds of FaDPa+ followed by α-Cut. One alternative to α-Cut in the proposed FaDSPa is to replace α-Cut by normalized cut and keep the remaining method same. We denote these schemes by N⟨*number*⟩. This stands for the method when FaDSPa is applied by repeatedly running FaDPa+ ⟨*number*⟩ number of times before passing the control to normalized cut. Thus N1 applies one *round* of FaDPa+, and the generated DPG after that is treated by normalized cut to obtain the k partitions.

Quality comparison of F1 and N1: Figure 7 shows the quality of partitioning obtained by the F1 and N1 schemes of FaDSPa. It presents a clear comparison of α-Cut and normalized cut when they are embedded in FaDSPa. Figures 7(a), 7(b), and 7(c) show the quality in terms of *inter*, *intra*, and ANS respectively (shown in Y-axis) for the number of partitions k varying from 2 to 20 (shown in X-axis) using two curves. The overall partitioning quality is shown in terms of ANS in Figure 7(c). We observe that at all the values of k, F1 is lower (better in quality) than N1. As this measure considers both the inter-partition and intra-partition distances, it very clearly shows that our proposed FaDSPa algorithm (using α-Cut) outperforms the other method.

We also look into the inter-partition heterogeneity and intra-partition homogeneity individually. In Figure 7(a), we observe that except at $k = 2$, 3 and 4, the *inter* measure of F1 is always greater than or equal to that of N1. As a higher *inter* indicates a better partitioning in terms of inter partition distances, it means that most of the times F1 performs better than N1 in terms of this measure. In Figure 7(b), we observe that except at $k = 12$ and 17, the *intra* measure of F1 is always smaller than that of N1. As a lower *intra* indicates a better partitioning in terms of

[9] All subsequent usage of this term refer to the schemes F⟨*number*⟩



(a) Inter (b) Intra (c) ANS

Figure 7: Comparison of proposed FaDSPa and normalized cut based FaDSPa on real data

intra partition distances, it means that most of the times F1 performs better than N1 in terms of this measure. Thus we see that sometimes F1 performs inferior to N1 in terms of either *inter* or *intra* individually, but as the overall partition quality considers both the factors simultaneously, F1 always outperforms N1.

Quality comparison of F1 and F2: After we are known about the good performance of our proposed FaDSPa, it is important to look into the variation in quality as we increase the number of *rounds*. Figure 8 shows the quality of partitioning obtained by the F1 and F2 schemes of FaDSPa. Both of them use α-Cut. It presents a comparison of one and two rounds of the density-based FaDPa+ (both followed by α-Cut). The overall partitioning quality is shown in terms of ANS in Figure 8(c). We observe that at all the values of k except 10, F1 is lower (better in quality) than F2. It shows that generally fewer rounds of FaDPa+ followed by α-Cut produces results in better quality. Also in terms of inter-partition and intra-partition distances individually, we found that most of the times F1 performs better than F2. In Figure 8(a), we observe that except at $k = 2$ and 3, the *inter* measure of F1 is always greater than that of N1. It means that most of the times F1 performs better than F2 in terms of inter-partition distances. In Figure 8(b), we observe that except at $k = 4$, 12 and 20, the *intra* measure of F1 is always smaller than that of F2. It means that most of the times F1 performs better than F2 in terms of intra-partition distances.

Summary of comparisons: Figure 9 presents the overall results summary obtained on the real M_s dataset. It shows the final comparison of the F1, F2, and N1 schemes (in the X-axis) in terms of ANS (in the Y-sxis). As mentioned in Section 5.3, the ANS measure also gives the information to identify the optimal number of partitions by selecting the k where the its minimum is found. We see in Figures 7(c) and 8(c) that the minima of F1, F2 and N1 occur at $k = 6$, 2, and 6 respectively. These values of k become the optimal number of partitions for the respective schemes. We compare the quality of partitioning in terms of ANS obtained at these optimal values of k in the figure. The lowest value of F1 shows itself as the best performer, followed by F2 and N1. Looking into the running times of F1 and F2, we found that they take 173.56 and 17.41 seconds respectively to complete the execution. Thus F1 produces better results than F2 in terms of effectiveness, but takes longer execution time, thus sacrificing the efficiency. The running time of FaDSPa on all the datasets is discussed in detail in Section 5.5.

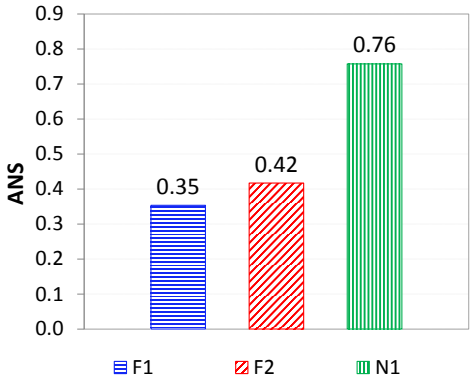
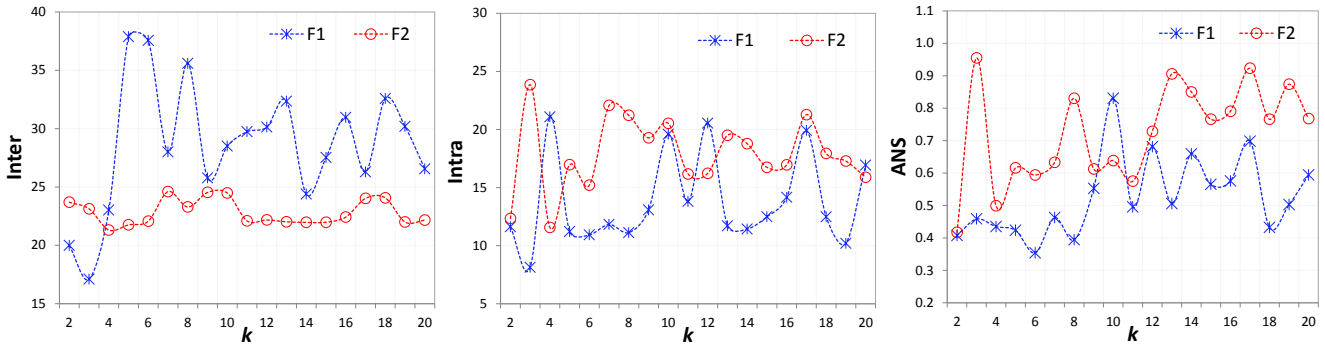


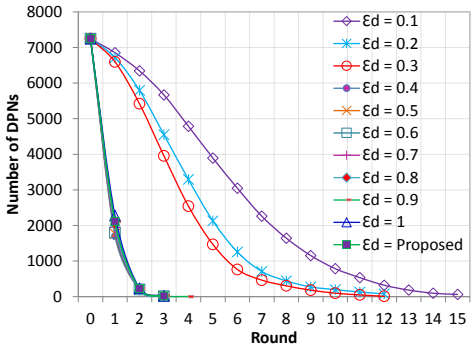
Figure 9: Partitioning results summary on SCATS data

Impact of ϵ^d in FaDPa+: The distance threshold ϵ^d is a fundamental parameter of FaDPa+ and has an impact on the quality of partitioning. As mentioned in Section 3.3, we consider ϵ^d as a vector of values, where each of those values individually refer to a specific node in the graph. It means that the distance threshold for each node is customized according to the kind of links of the referred node, which helps in clustering the nodes relatively. As it is an external parameter, it can also be set to some fixed value ($\in [0, 1]$) that is same for all the nodes. When it is set to low values (less than 0.4), due to the strict condition the number of established *density parent-child* relationships is found to be low. As shown in Figure 10(a), the DPG is compressed at a slow rate in each round of FaDPa+ for $\epsilon^d = 0.1, 0.2, \& 0.3$, and therefore requires more *rounds* to produce the final partitions. During this gradual compression, most of the nodes accumulate as children of one (or very few) density peak. It results into an imbalanced set constituting one (or very few) big partition and several small partitions, which is not good. For example, setting $\epsilon^d = 0.1$ results into one big partition of 7179 nodes and the remaining 66 nodes are distributed into 61 other partitions. The results gradually improve as this

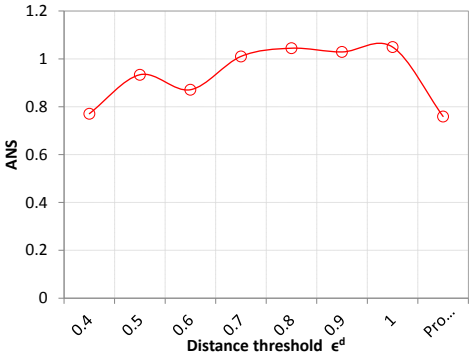


(a) Inter (b) Intra (c) ANS

Figure 8: Partitioning results of FaDSPa on real data



(a) DPG compression



(b) Quality of Partitions

Figure 10: Impact of the distance threshold ϵ^d

threshold is increased, and become satisfactory only after 0.4. Therefore, ignoring $\epsilon^d = 0.1, 0.2, \& \ 0.3$, Figure 10(b) shows the quality of partitions obtained by varying ϵ^d from 0.4 to 1, and compares it with the proposed method of determining its value. Observe that the ANS measure for the proposed method is the lowest of all, and thus leads to the best quality results.

Visualization of obtained partitions: Figure 11 presents a snapshot of the different partitions obtained by F1 setting k = 6 (optimal number of partitions) at 08:00 AM on 03-12-2012 (Monday). The road segments in each different color represent a partition. The displayed road network includes only those road segments that are operated by traffic signals and have the traffic sensors installed. Therefore even though the area is large, the number of road segments is relatively small. In the Section 5.5, we consider all the road segments that exist on digital maps, which results into large number of road segments in small areas.

5.5. *Experimental Results on Large Networks*

Through our experiments on large urban road networks, we show the performance of FaDPa+ and FaDSPa at different values of the thresholds ϵ_p and ϵ_c respectively. As FaDSPa provides the flexibility to handle the complexity of large networks, we also show the trade-off between efficiency and accuracy by varying the external parameter ϵ_c.

Impact of ϵ_p in FaDPa+: Table 4 shows the results obtained using FaDPa+ on the M_1 and M_2 datasets. In this algorithm the number of desired partitions is controlled by ϵ_p, shown in the left column, and the actual number of partitions denoted by k could be any value less than or equal to that number. The results in terms of *intra*, *inter*, and ANS are shown for ϵ_p = 25, 50, 75, and 100. The optimal ϵ_p, optimal k, and optimal partitioning are determined by looking into the minimum ANS. The minimum values of 0.75 in M_1 and 0.78 in M_2 at ϵ_p = **75** (k = 69 and 63 respectively) indicate that 75 is the optimal ϵ_p for both the datasets. We also observe that the same value of ϵ_p leads to different values of k in different datasets.

Table 4: FaDPa+: quality of partitioning

	M_1				M_2			
ϵ_p	k	Intra	Inter	ANS	k	Intra	Inter	ANS
25	23	0.4554	1.2805	1.1375	24	0.1691	0.2904	1.1857
50	47	0.4384	1.2886	0.8071	46	0.2274	0.3735	0.8632
75	69	0.3878	1.2934	**0.7463**	63	0.2947	0.5810	**0.7826**
100	93	0.4849	1.1628	0.8322	89	0.3326	0.6113	0.8146

Accuracy in FaDSPa: Figure 12 shows the partitioning quality of FaDSPa in large road network datasets M_1, M_2 and M_3. Figure 12(a) shows the ANS measures obtained for k = 2 to 10 for M_1. There are four curves for F2, F3, F4 and F5. As explained earlier in Section 5.4, F⟨*number*⟩ denotes the method when FaDSPa is applied by repeatedly running FaDPa+

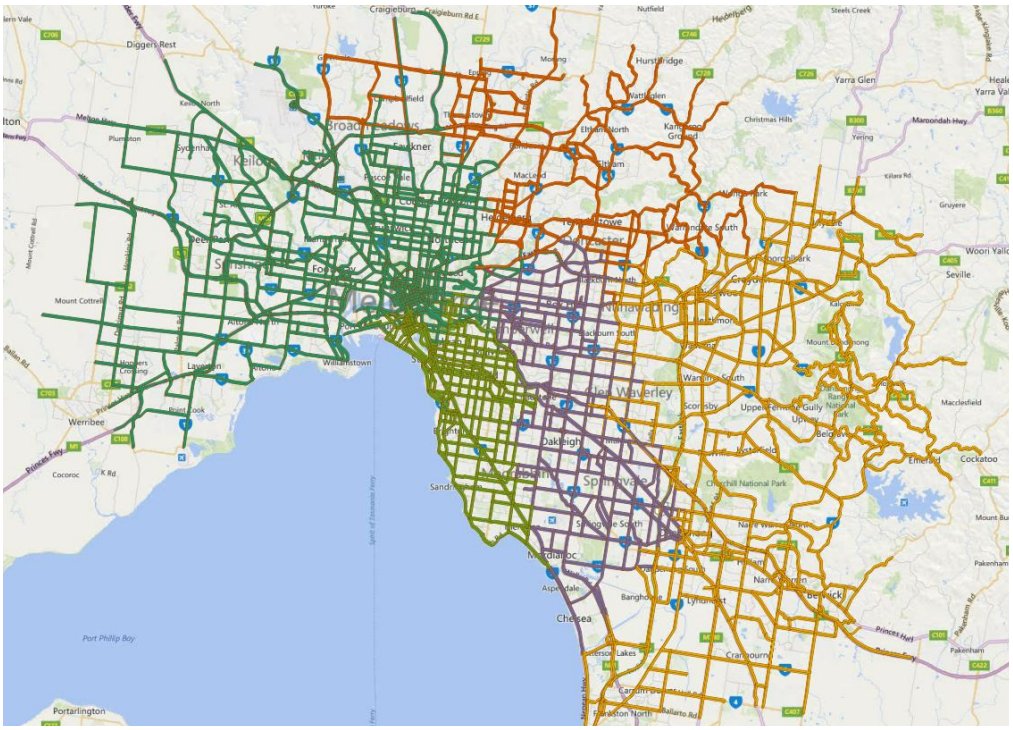
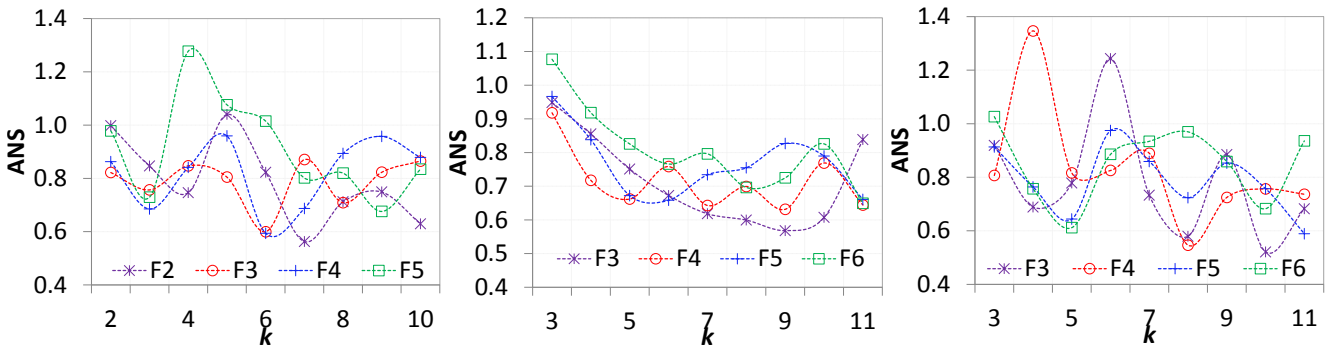


Figure 11: Partitions obtained from the Melbourne network at 08:00 AM on 03-12-2012 (Monday)



(a) ANS in $\mathtt{M_1}$ (b) ANS in $\mathtt{M_2}$ (c) ANS in $\mathtt{M_3}$

Figure 12: Partitioning results in large networks

$\langle$***number***$\rangle$ number of times before passing the control to α-Cut. Thus F2, F3, F4, and F5 apply two, three, four, and five *rounds* of FaDPa+, and the generated DPG after that is treated by α-Cut to obtain the k partitions. Fewer rounds of FaDPa+ produces a large DPG and puts more work on α-Cut, and vice versa. Thus the figure shows the quality of results obtained by varying the combination of our density and spectral based methods. We observe that there is no such clear trend that for all the values of k, one setting outperforms the other. The reason is that the partitioning quality is also highly dependent on the selected value of k, which could vary for the different schemes. To know their relative performance, we find the most suitable k for each of them. The scheme F2 has its minimum at $k = 7$, which shows 7 as its most suitable k. Similarly F3, F4, and F5, have their most suitable k as 6, 6 and 9 respectively. Their ANS values at the minimum are 0.56, 0.60, 0.59 and 0.68 respectively. Comparing the depth of their minimum we observe that the order of their performance is $F2 > F3 \simeq F4 > F5$. Figures 12(b) and 12(c) show the ANS measures for the larger datasets $\mathtt{M_2}$ and $\mathtt{M_3}$ starting[10] from F3 to F6. A similar performance trend is found also for these larger datasets, which are $F3 > F4 > F5 \simeq F6$ and $F3 > F4 > F5 > F6$ respectively. We observe that the partitioning quality is generally better with fewer rounds of FaDPa+ (or lower values of $\langle$***number***$\rangle$) and doing more of the task by α-

[10]The reason we do not start from F2 is that their DPGs produced after two rounds of FaDPa+ have a large number of DPNs (close to 10,000), and to apply α-Cut on such a large graph is beyond the scope of our computing environment because of the high computational complexity.

Cut. The reason behind this is that a larger value of ⟨*number*⟩ compromise more with the partitioning accuracy, and improve the efficiency[11]. We saw in previous experiments that α-Cut performs better than the density-based FaDPa+, because of its global perspective in partitioning. When FaDPa+ is repeatedly applied for F⟨*number*⟩, it starts with forming small partitions locally which combine with others in the subsequent rounds and form larger partitions in each repetition. In this way, they form hierarchical partitions locally. Once a grouping (in the form of a partition) is formed based on the local density-based information during these steps, it is done permanently for the final result. These small partitions are not broken or re-adjusted later, which leads to increase in lose of accuracy in each round. After ⟨*number*⟩ rounds, α-Cut (that looks into the graph globally) is applied on the DPG constructed from the obtained intermediate partitions to get the final partitions.

Efficiency in FaDSPa: Figure 13 shows the execution time (in seconds) when k is set to 4, 6, 8, and 10, for all the three large datasets. In the X-axis we vary the number of rounds of FaDPa+ from 2 to 10 (F2 to F10). We observe that for any value of k, as the rounds increase the execution time decreases, while at the same time the accuracy degrades, and vice versa. Another thing to note is that, the execution time decreases drastically in the initial rounds (decreases from more than 100 secs to around 40 secs from F2 to F3 for M_1) and this amount of decrease reduces in the subsequent rounds (decreases from around 40 secs to around 30 secs from F3 to F4). Thus the efficiency increases drastically as we progress through the initial rounds but becomes almost stable later, whereas the accuracy decreases as the rounds increase and no such drastic decrease is seen. It suggests that if the efficiency of execution is also a concern in addition to accuracy, then it is worth sacrificing the accuracy in the first few rounds. Thus a good choice of rounds for M_1 could be any one of F3, F4, and F5, as they can do the work efficiently with a satisfactory accuracy. The elbow point can be used as an indicator to locate the best choice. In the higher rounds, the execution time goes a little higher, though it is not very significant. This behavior and the reason behind it are explained later in detail. In the figures we see that the curves for the different values of k almost overlap. It shows that though a larger k requires a longer execution time, the difference is small.

Efficiency analysis in detail: Table 5 shows the running time of FaDPa+ and FaDSPa at varying ϵ_c (by varying the number of *rounds*) on all the datasets. The left most column stands for the method, which is either FaDPa+ or F⟨*number*⟩, where F0 refers to the method of applying α-Cut directly on the road graph. The table has other columns for the number of DPNs obtained after ⟨*number*⟩ *rounds* and the execution time in seconds for each dataset. For the small dataset D_1 that has 420 nodes, the methods F3 and higher are not applicable, as in the third round it FaDPa+ merges all the partitions into a single large partition. It completes running within fractions of a second. Similar to D_1, F4 and higher are not applicable to the real dataset M_s. In the large datasets, the running time decreases as ϵ_c becomes lower (with increasing *rounds*). The reason for this decrease is that ϵ_c decides how much the road graph is to be compressed using FaDPa+ to form the DPG, and thus the size of the matrix that has to be eigen-decomposed for applying α-Cut. As eigen-decomposition is a computationally expensive task, the running time increases substantially as the size of this matrix goes beyond a limit, where the limit depends on the computing environment. We see that there is a large difference between the time taken by F0 and F1 in M_s. It jumps to 10507.20 seconds in F0 from 173.56 seconds in F1. Similarly there is a sudden rise of execution time from F2 to F1 in M_1, and F3 to F2 in M_2 and M_3. On applying α-Cut directly (F0) on M_1, it could not complete execution even in 10 hours, and its application on M_2 and M_3 could not be performed due to higher memory requirement. The reason for such behavior is the expensive eigen-decomposition task. In the initial rounds of FaDPa+, the number of DPNs reduce rapidly whereas in the successive rounds they reduce at a slower rate. In M_1 the number of DPNs at F1, F2, F3, and F4 are 7402, 2990, 1419, and 814, respectively. The difference between the first two cases is more than the difference between the last two. The same trend is also observed with M_2 and M_3.

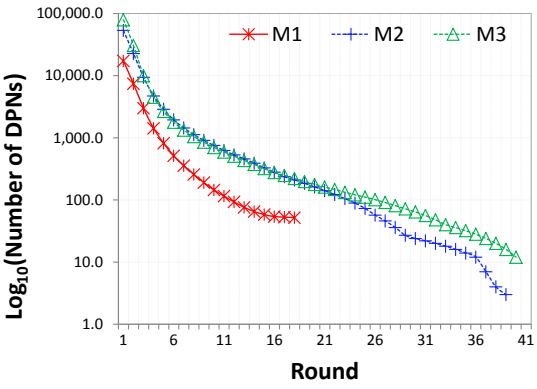
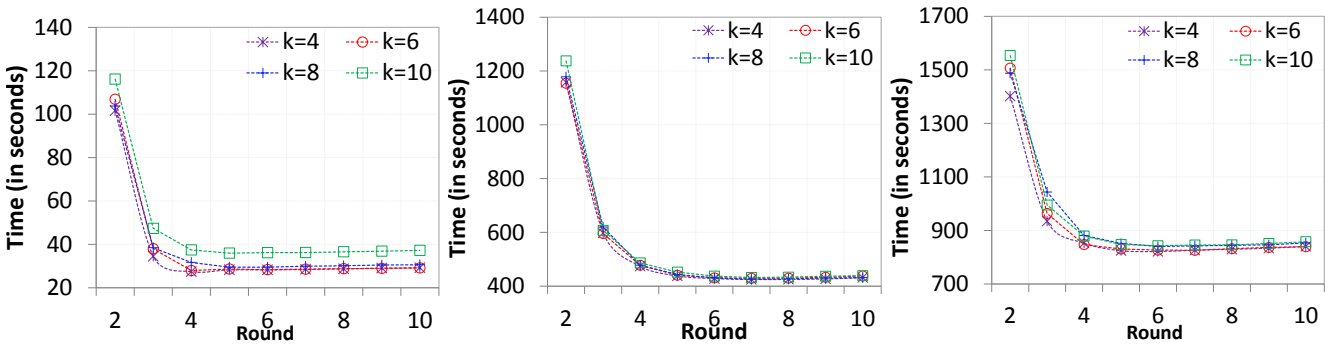


Figure 14: Compression of the DPG at different rounds

While normally it is expected that FaDPa+ would have the lowest running time because no eigen-decomposition task is needed, we actually see that the minimum running time is taken by F5 in case of M_1. It reduces as the method goes close to F5 and then increases gradually as it goes further. Similarly, the minimum execution time for M_2 and M_3 occur at F8, and F7 respectively. It shows that for large datasets, purely density based clustering methods take more time than when it is combined with spectral clustering as in FaDSPa. The reason behind this phenomenon can be explained from Figure 14. For all three datasets, it shows the compression rate of the DPG by using FaDPa+. The vertical axis is the logarithm of the number of DPNs and the horizontal axis is the number of rounds of FaDPa+. We observe that at lower rounds, the number of DPNs reduces rapidly, and at later rounds, the DPNs remain almost constant. Thus achieving further compression requires many more iterations in the higher rounds, which consumes much longer time for FaDPa+. But if α-Cut is applied at that point instead of further compression using FaDPa+, the task can be

[11] Shown in Figure 13 and explained in the next paragraph.



(a) Running time in M_1 (b) Running time in M_2 (c) Running time in M_3

Figure 13: Running time in large networks

Table 5: Running Time (in seconds)

Method	**DPN-D_1**	**Time-D_1**	**DPN-M_s**	**Time-M_s**	**DPN-M_1**	**Time-M_1**	**DPN-M_2**	**Time-M_2**	**DPN-M_3**	**Time-M_3**
FaDPa+	NA	less than 1	NA	16.01	NA	37.58	NA	538.16	NA	887.196
F0 (α-Cut)	420	less than 1	7245	10507.20	17206	-	53494	-	79487	-
F1	109	less than 1	2124	173.56	7402	10723.70	22670	-	30543	-
F2	13	less than 1	224	17.41	2990	116.15	9398	25018.30	9967	28477.60
F3	1	NA	17	15.77	1419	47.39	4715	1237.63	4495	1553.46
F4	NA	NA	NA	NA	814	37.39	2874	607.31	2654	995.10
F5	NA	NA	NA	NA	512	**35.96**	1938	486.61	1785	879.18
F6	NA	NA	NA	NA	355	36.18	1439	452.09	1327	847.59
F7	NA	NA	NA	NA	257	36.20	1125	437.43	1047	**843.25**
F8	NA	NA	NA	NA	190	36.55	906	**432.22**	849	845.61
F9	NA	NA	NA	NA	145	36.83	752	433.77	698	846.46
F10	NA	NA	NA	NA	115	37.19	626	436.90	591	851.39

done in a single iteration, thus requiring much less execution time.

6. Related Work

With the rapid growth in expansion of urban areas and frequent movement of people, the urban road networks has become an important area of research for several transportation related problems. Spatial partitioning of urban road networks is one such problem, having its objective as identifying the differently congested partitions or sub-networks. It has its theoretical foundation in clustering and general graph partitioning, which are well studied problems. This section starts with presenting some basic clustering and graph partitioning techniques, followed by important related works on spatial clustering and spatial network partitioning.

6.1. Clustering and Graph Partitioning

Clustering and graph partitioning apply to a wide variety of research problems, and many solutions have been proposed in the past. The k-means [23] and expectation-maximization [5] clustering algorithms work well for finding ellipsoidal or convex shaped clusters, but fail to find non-convex clusters. Density based clustering algorithms [12] are able to find clusters of arbitrary shapes. Areas of higher density are considered as clusters, and the nodes in the sparse areas separating the dense areas are considered as noise or boundary nodes. The density based spatial clustering of applications with noise (DBSCAN)[7] algorithm is the most popular of this kind. It identifies a cluster by looking into the neighborhood of each object within a predefined radius of ϵ distance. With each *minpts* (predefined) objects in the ϵ-neighborhood, a new cluster is formed. However, this algorithm is highly sensitive to the thresholds ϵ and *minpts*, and choosing an appropriate threshold is very important to get accurate clusters [28]. The authors in [28] recently proposed a fast density based clustering method by finding the density peaks locally. They assume that the cluster centers are surrounded by neighbors with lower local density and they are at a relatively large distance from any point of higher local density. Extending the DBSCAN concepts, SCAN (structural clustering algorithm for networks) [36] partitions a graph based on its structure to detect the clusters, hubs and outliers. A recently proposed algorithm SCAN++ [30] uses a new data structure called directly two-hop-away reachable node set (DTAR) to efficiently partition a graph in order to get the same results as produced by SCAN. Spectral clustering algorithms like minimum cut and normalized cut have remained quite popular [29, 35]. In [6], the authors proposed a spectral cut based on the min-max

clustering principle for graph partitioning in a data clustering point of view. In [35], White and Smith proposed a spectral clustering based solution to find communities in graphs by partitioning. Their objective function is based on network modularity. The modularity of a set of graph partitions is defined as the difference between the observed and the expected fraction of links within a partition. Larger modularity values are correlated with better graph partitioning. The minimization of our α-Cut approximately maximizes the modularity, and thus it gives an indication of good performance of our α-Cut.

As graph partitioning is an NP-complete problem, multilevel and heuristic algorithms have also been studied [17]. Zhou et al. [42] aimed to obtain graph partitions in which the nodes inside a partition are structurally close to each other and have similar feature values, and followed a random walk based approach. Sun et al. [31] integrated the problems of ranking and clustering in heterogeneous information networks and proposed the algorithm RankClus that produces clusters with rank information of the objects in the network. There is a wide application of graph partitioning for network community detection. In [20], the authors explored some graph partitioning based community detection methods and evaluated their relative performances. However, most works on graph partitioning face time and space complexity issues with large networks. In the proposed framework, our partitioning algorithm `FaDSPa` overcomes these issues by distributing the partitioning load into two levels, comprising the density based `FaDPa+` and the spectral based α-Cut.

6.2. Spatial Clustering and Spatial Network Partitioning

Spatial clustering is an important component of spatial data mining, which groups similar spatial objects into classes using basic clustering algorithms [27, 10, 11]. In the recent years, it has been studied from different perspectives for different kinds of data including spatial trajectories, traffic data, and spatial streaming data. It is an important component in mining different traffic patterns. Trajectory clustering has remained an important problem for mining people movement patterns [18, 13]. In [18], Lee et al. presented a partition-and-group framework for clustering trajectories. It starts with optimally partitioning each trajectory into a set of line segments using the minimum description length (MDL), and then the grouping phase groups the similar line segments into clusters using a density based clustering method. Recently Hung et al. [13] used clues based on movement behavior to cluster similar trajectories into groups, which leads to find partial trajectory routes. Thereafter they do a clue-aware trajectory aggregation to derive the complete trajectory pattern and route. *FlowScan* proposed by Li et al. in [22] finds the hot routes in a road network by clustering the road segments based on the density of commonly shared traffic. The authors in [34] proposed an efficient incremental algorithm to cluster the spatial data streams collected from sensors. Their method first predicts the clusters roughly using the previous clustering results, and then refines them further in the next stage. In [21], the authors discover the spatial co-clustering patterns in traffic collision data by identifying the sets of non-spatial attribute-value pairs of collision data, e.g., weather conditions and day of the week, that together contribute significantly to the spatial clustering of corresponding collisions.

Though graph partitioning in general has been well studied, not much work have been done on the spatial partitioning of road networks. In [14], the authors proposed a normalized cut based method for spatial partitioning of transportation networks. They tried to achieve three predefined criteria of small variance of within-partition traffic density values, small number of partitions, and spatially near-compact partitions. Their method starts by excessive partitioning of the road network using normalized cut, followed by merging smaller partitions, and then locally adjusting the road segments lying on partition boundaries by replacing them into the neighboring partitions. It works well for small road networks, but suffers from high time and space complexities for large networks. In [8] the authors proposed two heuristic methods to partition the road network into a set of subnetworks that are balanced in terms of their size. The first method follows a recursive approach to find the sparsest cuts that lead to balanced partitions in terms of their size. The second method applies a greedy-based coarsening iteratively along the high-flow links, and terminates when the number of nodes in the coarsened network is equal to the required number of partitions. The authors have reported that their method works fine for small-sized networks, but the running time increases significantly with the increasing the network size because of the expensive computations in determining the optimal maximum concurrent flow (MCF). In our recent work [3], we proposed a method for traffic-based spatial partitioning of large road networks that outperformed existing techniques. The method starts with constructing a road graph from the given road network, followed by mining a road supergraph, and then partitioning the supergraph. The actual road network partitions are then extracted from the supergraph partitions by mapping them to the road network. The supergraph mining is done by clustering the feature values using k-means and connecting the linked nodes in each cluster. The partitioning of the supergraph is done by approximately optimizing a measure called α-Cut, by following a spectral clustering based solution. However, there exist some outstanding issues in the supergraph mining method, explained earlier in Section 1. These are the computationally expensive learning of right number of clusters to create supernodes, the weak relation between k and the number of supernodes, and the non-consideration of node connectivities. We address the issues in this paper by mining a density peak graph (instead of the supergraph) using density based clustering and applying spectral clustering based α-Cut to partition the density peak graph. It makes our framework good in both efficiency and accuracy.

There exist some other works that treat spatial network partitioning as a secondary problem to solve some other problem of primary concern. Some works suggest to partition the network into small subnetworks and use distributed computing in parallel to efficiently solve different transportation related problems in large road networks [32]. The authors in [32] used existing graph partitioning techniques to form a hierarchy of nodes in a spatial network and proposed an index structure called a *partition tree* that can be used for efficient spatial query pro-

cessing. Some works partition the road networks in a way that suits their application, including monitoring proximity relations [37], point to point shortest path query indexing [16], traffic prediction [41], and finding distance-preserving subgraphs [38]. In [24], the authors partition a sensor network such that the data dissimilarity between any two nodes inside a partition is at most δ. They proposed a distributed clustering algorithm called *ELink* that works for both synchronous and asynchronous networks.

Our current and previous [3] papers identify the differently congested partitions (having different levels of traffic) in an urban road network (a snapshot shown in Figure 11), and contribute to the urban traffic congestion management. Crowd management and dealing with traffic congestions are of great importance in transportation networks [19, 33]. Wang et al. [33] developed an interactive system for visual analysis of urban traffic congestion based on GPS trajectories. They clean the trajectories and match them to the road network, based on which they detect the traffic jams. They also studied the traffic-jam propagation using graphs over a period of time. Their work mainly shows congestion on individual road segments having no concrete information about how the congestion is linked in the whole network globally. In contrast, the differently congested partitions produced by our framework gives the high-level insight of traffic congestion in the form of partitions instead of individual road segments. It also helps us analyze the connectivity of congested and non-congested sub-networks via road segments in the urban road network. Some other words related to traffic congestion on road networks include our recent papers [2, 4] where we study the evolution of congestion in different perspectives, and [1] where we identify important segments in a road network having high influence in propagating congestion.

7. Conclusion and Future Work

In this paper, we presented a framework called `FaDSPa` for spatial partitioning of large urban road networks, that employs both density and spectral based clustering. It is based on the data of traffic congestion on a road network defined by the vehicle density per unit distance on each road segment. It starts by transforming the actual road network into a *road graph*, followed by mining a *density peak graph* using our density based clustering algorithm called `FaDPa`. Thereafter we apply our spectral clustering based α-Cut on this graph to obtain the different *road network partitions*. The framework makes use of the locally distributed computations of `FaDPa` and the globally centralized computations of α-Cut together to make it efficient as well as effective. Our experiments on real SCATS data shows that it is very much applicable in real environments. Our method outperforms the existing road network partitioning method based on normalized cut. We found that in small networks our α-Cut performs the best whereas `FaDSPa` produces satisfactory results, but in large networks direct application of α-Cut brings huge time and space complexities that may go beyond the scope of the computing environment. `FaDSPa` handles such large networks with the help of a density peak graph, also providing the flexibility of setting the trade-off between efficiency and accuracy. We found that density based computations are faster, whereas the spectral based computations give better results in terms of quality.

Traffic congestions in peak hours is a big problem in urban road networks. The road network partitions discussed in this paper are the differently congested segments of a road network at a point of time, which are determined by the level of traffic in them. This work focuses mainly on the static partitioning of road networks. However, as the traffic keeps changing dynamically, an important problem for further study is to develop an indexing scheme that can keep on updating the existing partitions efficiently as the new recent traffic data is recorded in real time. From the obtained set of partitions, a congestion can be identified as a partition having its congestion level higher than a threshold. The study of their formation, evolution with time, and identifying the pattern of evolution are crucial problems for future work.

Acknowledgment

This research was supported by Data61 (formerly NICTA), Australian Reasearch Council (ARC) Discovery Project DP140103499, and ARC Future Fellowship grant FT120100723. Data61 is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

References

[1] T. Anwar, C. Liu, H. L. Vu, M. S. Islam, Roadrank: Traffic diffusion and influence estimation in dynamic urban road networks, in: Proc. of the CIKM, 2015.

[2] T. Anwar, C. Liu, H. L. Vu, M. S. Islam, Tracking the evolution of congestion in dynamic urban road networks, in: Proc. of the CIKM, 2016, (to appear).

[3] T. Anwar, C. Liu, H. L. Vu, C. Leckie, Spatial partitioning of large urban road networks, in: Proc. of the EDBT, 2014.

[4] T. Anwar, H. L. Vu, C. Liu, S. P. Hoogendoorn, Temporal tracking of congested partitions in dynamic urban road networks, Transportation Research Record: Journal of the Transportation Research Board 2595 (2016) 88–97.

[5] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum likelihood from incomplete data via the em algorithm, Journal of the Royal Statistical Society. Series B (Methodological) 39 (1) (1977) 1–38.

[6] C. Ding, X. He, H. Zha, M. Gu, H. Simon, A min-max cult algorithm for graph partitioning and data clustering, in: Proc. of the ICDM, 2001.

[7] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proc. of the KDD, 1996.

[8] H. Etemadniaa, K. Abdelghanya, A. Hassan, A network partitioning methodology for distributed traffic management applications, Transportmetrica A: Transport Science 10 (6) (2014) 518–532.

[9] H. Gonzalez, J. Han, X. Li, M. Myslinska, J. P. Sondag, Adaptive fastest path computation on a road network: a traffic mining approach, in: Proc. of the VLDB, 2007.

[10] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann, 2000.

[11] J. W. Han, M. Kamber, A. K. H. Tung, Spatial clustering methods in data mining: A survey, Taylor & Francis, 2001, pp. 188–217.

[12] K. Hans-Peter, K. Peer, S. Jorg, Z. Arthur, Density–based clustering, WIREs Data Mining Knowl Discov 1 (3) (2011) 231–240.

[13] C.-C. Hung, W.-C. Peng, W.-C. Lee, Clustering and aggregating clues of trajectories for mining trajectory patterns and routes, The VLDB Journal 24 (2) (2015) 169–192.
[14] Y. Ji, N. Geroliminis, On the spatial partitioning of urban transportation networks, Transportation Research Part B: Methodological 46 (10) (2012) 1639–1656.
[15] E. Kanoulas, Y. Du, T. Xia, D. Zhang, Finding fastest paths on a road network with speed patterns, in: Proc. of the ICDE, 2006.
[16] G. Kellaris, K. Mouratidis, Shortest path computation on air indexes, Proc. VLDB Endow. 3 (1-2) (2010) 747–757.
[17] M. Kim, K. S. Candan, Sbv-cut: Vertex-cut based graph partitioning using structural balance vertices, Data Knowl. Eng. 72 (2012) 285–303.
[18] J.-G. Lee, J. Han, K.-Y. Whang, Trajectory clustering: A partition-and-group framework, in: Proc. of the ACM SIGMOD, 2007.
[19] R. Lee, K. Sumiya, Measuring geographical regularities of crowd behaviors for twitter-based geo-social event detection, in: Proc. of the ACM SIGSPATIAL Int'l Workshop on LBSN, 2010.
[20] J. Leskovec, K. J. Lang, M. Mahoney, Empirical comparison of algorithms for network community detection, in: Proc. of the WWW, 2010.
[21] D. Li, J. Sander, M. A. Nascimento, D.-W. Kwon, Discovering spatial co-clustering patterns in traffic collision data, in: Proc. of the ACM SIGSPATIAL Int'l Workshop on Computational Transportation Science, IWCTS '13, 2013.
[22] X. Li, J. Han, J.-G. Lee, H. Gonzalez, Traffic density-based discovery of hot routes in road networks, in: Proc. of the SSTD, 2007.
[23] J. B. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proc. of the 5th Symposium on Math, Statistics, and Probability, 1967.
[24] A. Meka, A. K. Singh, Distributed spatial clustering in sensor networks, in: Proc. of the EDBT, 2006.
[25] M. F. Mokbel, L. Alarabi, J. Bao, A. Eldawy, A. Magdy, M. Sarwat, E. Waytas, S. Yackel, Mntg: an extensible web-based traffic generator, in: Proc. of the SSTD, 2013.
[26] M. F. Mokbel, L. Alarabi, J. Bao, A. Eldawy, A. Magdy, M. Sarwat, E. Waytas, S. Yackel, A demonstration of mntg– a web-based road network traffic generator, in: Proc. of the ICDE, 2014.
[27] R. T. Ng, J. Han, Efficient and effective clustering methods for spatial data mining, in: Proc. of the 20th Int'l Conf. on Very Large Data Bases, VLDB '94, 1994.
[28] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.
[29] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 888–905.
[30] H. Shiokawa, Y. Fujiwara, M. Onizuka, Scan++: Efficient algorithm for finding clusters, hubs and outliers on large-scale graphs, Proc. VLDB Endow. 8 (11) (2015) 1178–1189.
[31] Y. Sun, J. Han, P. Zhao, Z. Yin, H. Cheng, T. Wu, Rankclus: Integrating clustering with ranking for heterogeneous information network analysis, in: Proc. of the EDBT, 2009.
[32] J. Wang, K. Zheng, H. Jeung, H. Wang, B. Zheng, X. Zhou, Cost-efficient spatial network partitioning for distance-based query processing, in: Proc. of the MDM, 2014.
[33] Z. Wang, M. Lu, X. Yuan, J. Zhang, H. v. d. Wetering, Visual traffic jam analysis based on trajectory data, IEEE Trans. on Visualization and Computer Graphics 19 (12) (2013) 2159–2168.
[34] L.-Y. Wei, W.-C. Peng, An incremental algorithm for clustering spatial data streams: exploring temporal locality, Knowledge and Information Systems 37 (2) (2013) 453–483.
[35] S. White, P. Smyth, A spectral clustering approach to finding communities in graph, in: Proc. of the SDM, 2005.
[36] X. Xu, N. Yuruk, Z. Feng, T. A. J. Schweiger, Scan: A structural clustering algorithm for networks, in: Proc. of the ACM SIGKDD, 2007.
[37] Z. Xu, H.-A. Jacobsen, Processing proximity relations in road networks, in: Proc. of the ACM SIGMOD, SIGMOD '10, 2010.
[38] D. Yan, J. Cheng, W. Ng, S. Liu, Finding distance-preserving subgraphs in large road networks, in: Proc. of IEEE ICDE, 2013.
[39] D. Yan, L. Huang, M. I. Jordan, Fast approximate spectral clustering, in: Proc. of ACM SIGKDD, 2009.
[40] M. J. Zaki, W. Meira Jr., Data Mining and Analysis: Fundamental Concepts and Algorithms, Cambridge University Press, 2014.
[41] B. Zhang, K. Xing, X. Cheng, L. Huang, R. Bie, Traffic clustering and online traffic prediction in vehicle networks: A social influence perspective, in: Proc. of the IEEE INFOCOM, 2012.
[42] Y. Zhou, H. Cheng, J. X. Yu, Graph clustering based on structural/attribute similarities, Proc. VLDB Endow. 2 (1) (2009) 718–729.