Spatial Partitioning of Large Urban Road Networks

Tarique Anwar* Chengfei Liu* Hai L. Vu* Christopher Leckie†

*Swinburne University of Technology, Melbourne, Australia
{tanwar,cliu,hvu}@swin.edu.au
†University of Melbourne, Melbourne, Australia
caleckie@unimelb.edu.au

ABSTRACT

The rapid global migration of people towards urban areas is multiplying the traffic volume on urban road networks. As a result these networks are rapidly growing in size, in which different sub-networks exhibit distinctive traffic flow patterns. In this paper, we propose a scalable framework for traffic congestion-based spatial partitioning of large urban road networks. It aims to identify different sub-networks or partitions that exhibit homogeneous traffic congestion patterns internally, but heterogenous to others externally. To this end, we develop a two-stage procedure within our framework that first transforms the large road graph into a well-structured and condensed supergraph via clustering and link aggregation based on traffic density and adjacency connectivity, respectively. We then devise a spectral theory based novel graph cut (referred as α-Cut) to partition the supergraph and compare its performance with that of an existing method for partitioning urban networks. Our results show that the proposed method outperforms the normalized cut based existing method in all the performance evaluation metrics for small road networks and provides good results for much larger networks where other methods may face serious problems of time and space complexities.

Keywords

Graph partitioning; Road networks; Spectral clustering

1. INTRODUCTION

Traffic flow patterns in urban road networks have been found to vary significantly in different sub-networks depending on two critical factors– *i*) *spatial* importance, and *ii*) *temporal* importance. Usually roads of each locality, say inside a suburb or part of a suburb in a city, 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 monument or hospital, usually remain more congested than others without any such significance. Additionally, the congestion on roads connecting important places of human gathering like airports, train stations, hospitals, bus stops, etc., remains comparatively higher than others. Similarly in the temporal perspective, roads usually remain busier and more congested in peak hours than off-peak hours. As the sub-networks exhibit distinctive traffic flow patterns, the traffic management decisions for each sub-network need to reflect these differences.

These days the urban road networks are rapidly growing in size and the congestion on them is significantly increasing. Solutions are being explored to make the management of these large networks easier by identifying the distinctively congested areas and considering them as multiple small sub-networks. Their identification further aids in studying and analyzing the congestion and its evolving nature with respect to time. Therefore, the congestion-based spatial partitioning of urban road networks from a large data perspective is becoming a problem of growing importance. Although there exist many other kinds of information networks and the application of graph partitioning on such networks has been studied in the past [15, 8], the geospatial properties of a road network associated with traffic flow patterns make a unique kind of network [5]. The problem was recently raised in the intelligent transportation systems (ITS) community [5]. They contend that transportation networks have unique dynamic features and an arbitrary clustering algorithm may not produce the desired kind of partitions. In this paper we present a framework for congestion-based spatial partitioning of large urban road networks. Same as [5], we consider partitioning the network repeatedly at regular intervals of time using static congestion measures. As we focus on large road networks, scalability of the method for real-world applicability is also a major concern.

The partitioning framework is based on traffic congestion measures on a road network defined by the vehicle density per unit distance on each road segment. Its objective is to identify the different heterogeneous regions of an urban network which internally exhibit homogeneous traffic congestion patterns. The method starts with transforming the actual road network into a *road graph*, which is followed by mining the *road supergraph*. Finally the supergraph is subjected to a spectral theory based novel graph cut called α-Cut to obtain the set of road segments partitioned into several subsets called *road network partitions*. As the partitioning algorithm is applied on a supergraph with much reduced order(number of supernodes), the framework becomes more scalable by managing the computational and space complexity. In summary, we make the following contributions in this paper.

– We mine a well-structured and condensed road super-

(c) 2014, Copyright is with the authors. Published in Proc. 17th International Conference on Extending Database Technology (EDBT), March 24-28, 2014, Athens, Greece: ISBN 978-3-89318065-3, on OpenProceedings.org. Distribution of this paper is permitted under the terms of the Creative Commons license CC-by-nc-nd 4.0

10.5441/002/edbt.2014.32

graph from the road graph. Its main advantage is in making the partitioning method applicable on large road networks where the number of road segments is very large.
- To identify the optimal number of clusters for k-means while forming the condensed supergraph, we devise a measure to help find the optimal clustering.
- We devise a spectral theory based novel graph cut called α-Cut to partition the road supergraph, which outperforms normalized cut in our empirical study.
- Extensive experiments are performed on both small and large road networks to establish its efficacy.

The rest of the paper is organized as follows. Section 2 presents some preliminary theories followed by the problem definition. Section 3 presents the framework briefly. The complete methodology is described in Sections 4 and 5. Experimental results are shown in Section 6, followed by related work in Section 7. The paper is concluded in Section 8.

2. PRELIMINARIES

In this section, we present some preliminary theories on road networks and then formulate the problem.

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 *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 [7, 4]. Unlike the previously attempted problems, the focus of spatial partitioning of road networks is on the road segments and not the intersection points. A trivial representation in the form of a graph by considering roads as links and their intersection points as nodes makes no sense as its partitioning results in 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. A similar idea has been applied in [5].

Definition 1: (**Road Network**) A real urban road network is defined as $\mathcal{N} = (\mathcal{I}, \mathcal{R})$ comprising a set of intersection points $\mathcal{I} = \{\iota_1, \iota_2, \ldots, \iota_{n_\iota}\}$ 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 as the dual of $\mathcal{N}$ by adding each road segment r_i as a node v_i, and establishing an undirected link e_i between each possible node pair (v_j, v_k) if there exist at least one intersection point ι_l which is a common intersection for the roads r_j and r_k. Thus the links stand for the adjacency relationships among the road segments. In this manner, the road network components in a star topology form cliques in the road graph, whereas the linear components remain in the same topology. 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 partitioned into two parts from the middle for traffic of the two opposite directions. Each of the two parts undergo different kinds of traffic flow patterns. For example, on a road that connects outskirts with the city center, the morning office hours would find more traffic heading towards the city center whereas the evening office closing hours would find more traffic in the opposite direction. To accommodate this feature of the urban network, the two traffic directions are considered as separate road segments that share common intersection points and thus are adjacent. Figure 1 shows an example of road network representation in which Figure 1(a) is a sample road map, Figure 1(b) is the corresponding road network of those colored yellow in the map, and Figure 1(c) is the final representation called the road graph. To keep it simple and easy to understand, the traffic direction has not been taken into consideration and all the roads have been considered as one-way roads. The road graph is stored in the form of its $n \times n$ binary adjacency matrix $A_{\mathcal{G}}$ using sparse matrix representation.

2.2 Problem Definition

The problem of *spatial partitioning* of large urban road networks is defined as splitting up a given large urban road network based on traffic congestion measures into several disjoint partitions, keeping intact the associated spatial properties. The different partitions exhibit the property of intra-partition congestion *homogeneity* and inter-partition congestion *heterogeneity*. Let us suppose we have a real urban directed road network $\mathcal{N}$, which is transformed into a *road graph* $\mathcal{G}$ by following the method described in Section 2.1. This graph is then subjected to congestion-based spatial partitioning. Before formally stating the problem on $\mathcal{G}$, we present three 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 a measure of congestion similarity between the pair of nodes. ∎

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 pairs (v_i, v_j) for which v_i and v_j lie in the same partition, where the affinity values are a measure of congestion similarity between the pair of nodes. ∎

Definition 5: (**Partition Connectivity**) A partition $\mathcal{P}_l$ is said to be *connected* if for any given node pairs $(v_i, v_j) \in \mathcal{P}_l$ there exists a path from v_i to v_j or vice versa. ∎

The problem of congestion-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.

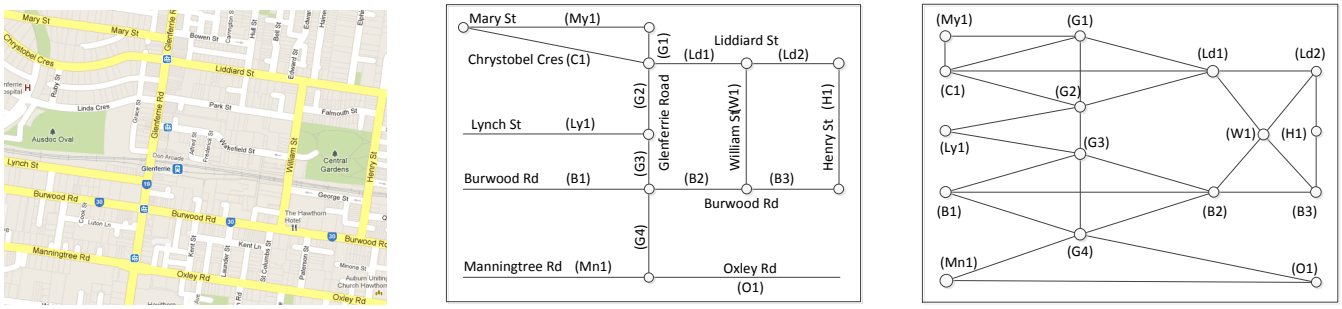
C.1 $\bigcup_{i=1}^{k} \mathcal{P}_i = \mathcal{V}$ and $\mathcal{P}_i \bigcap \mathcal{P}_j = \phi$ for all $i \neq j$;

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

C.3 the *cost of partitioning* of $\mathcal{G}$ is the minimum; and

C.4 the *partition volume* of $\mathcal{G}$ is the maximum.

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



(a) Actual road map (b) Road network (c) Road graph

Figure 1: Mathematical representation of road networks

(or linkage) of nodes, **C.3** enforces the condition of inter-partition traffic congestion heterogeneity, and **C.4** enforces intra-partition traffic congestion homogeneity. A partitioning may not satisfy **C.3** and **C.4** together simultaneously, and therefore the best possible trade-off has to be found.

3. FRAMEWORK

The task of road network partitioning can also be viewed as similar to that of clustering road segments based on their traffic density values. However, the main drawback of this approach is that traditional clustering algorithms do not take care of the associated spatial connectivities (connectivity of road segments). Consequently we treat it as a 2-level partitioning problem, in which the first level follows a bottom-up approach considering only data in the form of density values, whereas the second level follows a top-down approach considering both the density data along with the road segment connectivities.

The complete framework for spatial partitioning of road networks, shown in Figure 2, comprises three different modules– *i*) road graph construction, *ii*) road supergraph mining, and *iii*) supergraph partitioning. The first module deals with transforming the real road network $\mathcal{N}$ into a road graph $\mathcal{G}$ to give it a mathematical representation, which is explained as a preliminary step in Section 2.1. Due to the large and rapidly expanding urban area, the size of an urban road network $|\mathcal{R}|$ and the order of the corresponding road graph $|\mathcal{V}|$ may become extremely large, which heavily affects the computational and space complexity for partitioning $\mathcal{G}$. To address this problem, the framework follows a 2-level partitioning.

The first level (which is the second module described in Section 4) mines a road supergraph $\mathcal{G}_s$ from the road graph $\mathcal{G}$ with a much reduced order following a bottom-up approach. It goes through the steps of clustering feature values $v_i.f$ using k-means in Section 4.1 and constructing the road supergraph in Section 4.3.

Furthermore, an extended version of this module introduces the concept of *stable* supernodes. A supernode is considered to be *stable* if its *stability* measure, defined later, is above a predefined threshold. To have a *stable* supergraph, all the supernodes that are found unstable are further split up, which is repeated until they become stable. We can have the supergraph of different structures as per the application environment by varying the stability threshold. The stability threshold scale is also a trade-off between complexity and accuracy. A lower threshold value reduces the complexity by reducing the supergraph *order* while sacrificing some level of accuracy by presuming all nodes inside a supernode to belong to the same final partition. On the other hand a higher value can give more accurate results at the cost of computational and space complexity.

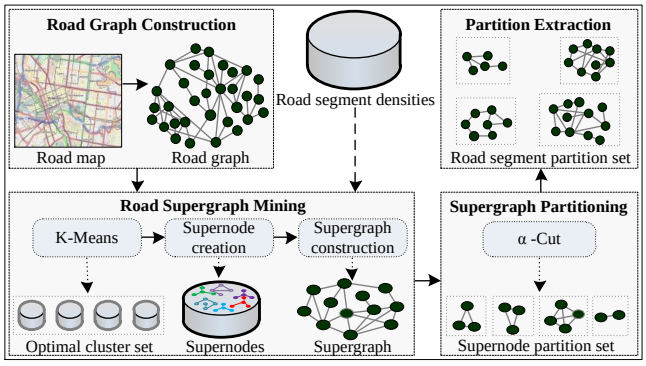


Figure 2: Proposed spatial partitioning framework

The last module of supergraph partitioning, described in Section 5, is the second level partitioning that 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.

4. ROAD SUPERGRAPH MINING

A naive approach to obtain road network partitions is to apply the partitioning algorithm on the road graph directly. However, we reduce the load of partitioning by following a 2-level partitioning. The first level mines a condensed *road supergraph*, before partitioning it in the second level.

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 congestion pattern of a segment more likely to be similar to (or dependent on) other (following or preceding) segments inside the partition. Thus the similar spatial importance of road segments within a partition leads them to exhibit similar congestion patterns. That means if they are grouped based on their traffic density measures, most of the time, they could be expected to be grouped together. To capture this aspect, before applying the partitioning algo-

rithm we group the segments based on their traffic density measures to find their clustering pattern, and use them to construct a condensed road supergraph. The following definitions present the idea of a supergraph used in this paper.

DEFINITION 6: (**Supernode**) Given a road graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a *supernode* ς_i having a feature value $\varsigma_i.f$, is defined as a set of nodes $\{v_j\}$ that exhibit the properties of being similar in terms of their feature values $\{v_j.f\}$ (density measures), and interlinked together. ∎

DEFINITION 7: (**Superlink**) Given a road graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a *superlink* ε_i is defined as a link between a pair of supernodes $(\varsigma_p, \varsigma_q)$, which exists only if there is at least one link $link(v_x, v_y) \in \mathcal{E}$ such that $v_x \in \varsigma_p$ and $v_y \in \varsigma_p$. ∎

DEFINITION 8: (**Road Supergraph**) Given a road graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a *road supergraph* $\mathcal{G}_s$ is defined as an ordered 3-tuple $(\mathcal{V}_s, \mathcal{E}_s, \mathcal{W}_s)$, where $\mathcal{V}_s = \{\varsigma_1, \varsigma_2, \ldots, \varsigma_{n_\varsigma}\}$ is the set of supernodes comprising the set of road segments, $\mathcal{E}_s = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{n_\varepsilon}\}$ is the set of superlinks, and $\mathcal{W}_s = \{\omega_1, \omega_2, \ldots, \omega_{n_\varepsilon}\}$ is the set of weights associated with each of the corresponding superlinks, which is defined as a measure of congestion similarity between the pair of supernodes connected by the superlink. ∎

The task of mining the supergraph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s, \mathcal{W}_s)$ is done in two steps. The first step deals with the feature values $v_i.f$ associated with each node in $\mathcal{G}$ to group them into different clusters, whereas the second step uses these clusters to construct the supergraph. The complete algorithm is shown in Algorithm 1. The popular clustering algorithm k-means is used to cluster the feature values associated with the node set. A major problem with k-means is its requirement for a pre-determined number of clusters. We overcome this problem by designing a novel optimality measure called *moderated clustering gain* (MCG) to determine the optimal number[1] of clusters κ for a dataset (line 6). Instead of considering the optimal value of κ, we consider all those κ for which the MCG value lies above a threshold (lines 3–9), and the one that produces the least number of supernodes is finally selected as optimal (lines 10–16). After creating the supernodes and assigning their feature values as cluster means from the optimal cluster set (lines 17–20), the superlinks in between the supernodes are established and weighted (lines 21–25) to construct the supergraph (line 26).

Algorithm 1: Road supergraph mining (Road graph $\mathcal{G}$, optimality threshold ϵ_θ)

```
1  A_G ← adjacency matrix of G;
2  F ← {v_1.f, v_2.f, ..., v_{n_r}.f};
   // shortlist cluster sets based on MCG threshold ε_θ
3  ϱ ← ϕ;
4  for κ ← 2 to (n_r − 1) do
5      (C^κ, μ(C^κ)) ← k-means(F, κ);
6      Θ(C^κ) ← MCG of C^κ;
7      if Θ(C^κ) ≥ ε_θ then
8          ϱ^κ ← cluster indicator vector of C^κ;
9          ϱ ← ϱ ∪ ϱ^κ;
   // select the optimal cluster set
10 ϱ^θ ← ϕ, C^θ ← ϕ;
11 min ← number of connected components in (ϱ^1, A_G);
12 forall the ϱ^κ ∈ ϱ do
13     comp ← number of connected components in (ϱ^κ, A_G);
14     if min > comp then
15         min ← comp;
16         ϱ^θ ← ϱ^κ, C^θ ← C^κ;
   // create supernodes and assign their feature values
17 V_s ← createSupernodes(ϱ^θ, A_G);
18 forall the C_i^θ ∈ C^θ do
19     forall the ς_j ∈ C_i^θ do
20         ς_j.f ← μ(C_i^θ);
   // establish superlinks and assign their weights
21 E_s ← ϕ, W_s ← ϕ;
22 forall the link(v_x, v_y) ∈ E do
23     if v_x ∈ ς_p and v_y ∈ ς_q and p ≠ q then
24         E_s ← E_s ∪ establishLink(ς_p, ς_q);
25         W_s ← W_s ∪ assignWeight(ς_p, ς_q);
26 G_s ← (V_s, E_s, W_s);
27 return G_s
```

4.1 Feature Value Clustering

This step looks into the feature values $v_i.f$ associated with the nodes in $\mathcal{G}$ without considering its adjacency relationships or connectivities with the intent to get a rough idea of the partitions, which are refined in subsequent steps to find the actual partitions. Let $\mathcal{F} = \{v_1.f, v_2.f, \ldots, v_{n_r}.f\}$ be the set of feature values associated with the set of nodes $\mathcal{V}$ in $\mathcal{G}$. The objective is to extract information about grouping patterns of the feature values, and therefore $\mathcal{F}$ is treated with the k-means clustering algorithm. It results in an organization of feature values in the form of clusters. There are a few limitations of k-means, and one of them is that it may result in a clustering configuration having a local maxima that may not be the global maxima. The outcome depends on the initialization of cluster means. As we have the feature values in a single dimension, we overcome this limitation by firstly sorting the feature values $v_i.f \in \mathcal{F}$ and then initializing the cluster means with feature values at equal intervals. That means, when we have n_r number of sorted feature values, the mean of the jth cluster is initialized by $v_i.f$, where $i = \frac{n_r}{\kappa} \times j$, and the following steps remain the same as the standard k-means algorithm.

As stated earlier, k-means needs to have a predetermined number of clusters that has to be provided as input to the algorithm. We address this problem by applying k-means repeatedly with different values of κ producing the set of clusters as $\mathcal{C}^\kappa = \{\mathcal{C}_1^\kappa, \mathcal{C}_2^\kappa, \ldots, \mathcal{C}_\kappa^\kappa\}$. It starts with $\kappa = 2$ incrementally and at each value an optimality test is performed. The optimality test compares the MCG measure, described in Section 4.2, with that computed in the preceding iteration at $\kappa-1$ and the following iteration at $\kappa+1$. The point where the value found is higher than both its preceding and following points, represents a *local optimality maxima*. However there is no guarantee that it will serve as the *global optimality maxima*. Applying k-means repeatedly on a large dataset just to learn the optimal number of clusters makes the method computationally very expensive, particularly in situations when the dataset is extremely large. To overcome this problem, repetitive clustering is applied on a randomly generated sample dataset, much smaller than the actual dataset. Let θ be the value of κ that produces the clustering configuration having the global optimality maxima. Instead of considering only the clustering configuration at θ, we consider all values of κ for whom MCG lies above a predefined threshold ϵ_θ, which are passed on to the next step to create supernodes in Section 4.3.

The computational complexity of k-means is $\mathbf{O}(tnd\kappa)$,

[1]We use the Greek lowercase letter kappa (κ) to refer the number of clusters produced by k-means in Section 4.1, and the English lowercase letter k to refer the number of partitions produced by the framework.

where t is the number of iterations needed to converge, n is the number of data items, d is the data dimension, and κ is the number of required clusters. In our case, $d = 1$, which makes it $\mathbf{O}(tn\kappa)$. Also κ is usually very small.

4.2 Optimality Measure

In this section, we design an optimality measure to learn the optimal number of clusters θ suitable for a data set. Let $\mathcal{D} = \{d_1, d_2, \ldots, d_{(n_d)}\}$ be the dataset consisting of n_d data items, where each d_i has m_d feature values forming an m_d-dimensional vector $\langle f_1, f_2, \ldots, f_{(m_d)}\rangle$. The global mean μ^0 is a vector given by $\langle \mu_1^0, \mu_2^0, \ldots, \mu_{(m_d)}^0\rangle$ where $\mu_p^0 = \frac{1}{n_d}\sum_{i=1}^{n_d} d_i.f_p$ corresponding to each feature f_p. Let $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_\kappa\}$ be the set of clusters generated by k-means, then the mean μ^q of each cluster $\mathcal{C}_q$ is a vector of feature means given by $\langle \mu_1^q, \mu_2^q, \ldots, \mu_{(m_d)}^q\rangle$ where $\mu_p^q = \frac{1}{|\mathcal{C}_q|}\sum_{d_i \in \mathcal{C}_q} d_i.f_p$ corresponding to each feature f_p. A measure called *clustering balance* $\mathcal{E}(\mathcal{C})$, defined in [6], has been found to be a good indicator for the optimal number of clusters, outperforming previous measures. Another comparable and computationally efficient measure called *clustering gain* $\Delta(\mathcal{C})$ is defined in the same work. The optimal clustering configuration is achieved at that value of κ where the clustering balance reaches its minimum, whereas clustering gain reaches its maximum. Although these measures were proposed for identifying optimality in hierarchical clustering, they were shown to be suitable for k-means too. In our analysis we found that when these measures are applied with k-means they produce a smaller number of sparse clusters. In this work we extend and improve clustering gain to make the clusters compact and far apart from others, named as *moderated* clustering gain, denoted by $\Theta(\mathcal{C})$. Shown in Equation 1, it is the summation of a value over all clusters $\mathcal{C}_q$, in which the value consists of two parts multiplied by each other. The first part $\Theta_1(\mathcal{C}_q)$ is the clustering gain, whereas the second part $\Theta_2(\mathcal{C}_q)$ is a function of the ratio of intra-cluster and inter-cluster error sums. For each $q \in [1, \kappa]$, the value of $\Theta_2(\mathcal{C}_q)$ lies in the range $[0, 1]$ and it moderates the value of $\Theta_1(\mathcal{C}_q)$ by reducing its effect accordingly.

$$\text{MCG}, \quad \Theta(\mathcal{C}) = \sum_{q=1}^{\kappa} (\Theta_1(\mathcal{C}_q) \times \Theta_2(\mathcal{C}_q)) \tag{1}$$

$$\text{where}, \quad \Theta_1(\mathcal{C}_q) = (|\mathcal{C}_q| - 1)\left\|\mu^q - \mu^0\right\|_2^2$$

$$\Theta_2(\mathcal{C}_q) = \left(1 - \log_2\left(1 + \frac{\sum_{d_i \in \mathcal{C}_q} \|d_i - \mu^q\|_2^2}{|\mathcal{C}_q| \times \|\mu^q - \mu^0\|_2^2}\right)\right)$$

The optimal number of clusters θ is that value of κ where $\Theta(\mathcal{C})$ attains the maxima.

4.3 Supergraph Construction

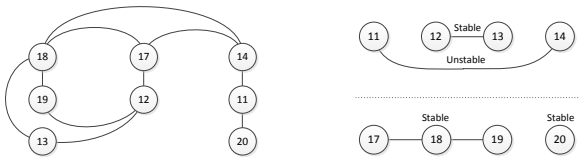
Supergraph construction starts with creating supernodes, then establishing weighted superlinks between them.

4.3.1 Supernode creation

Once the MCG measure for all values of κ are computed for the sample data, all those κ for which the value lies above a predefined threshold value ϵ_θ, are considered for supernode creation. The k-means algorithm is now applied on the complete dataset with the shortlisted values of κ. Let $\varrho = \{\varrho^\kappa : \Theta(\mathcal{C}^\kappa) \geq \epsilon_\theta\}$ be the set of clustering configuration indicator vectors, where each ϱ^κ is of length n_r, and the value of $\varrho^\kappa(i)$ indicates the cluster to which the node v_i belongs. These vectors along with the adjacency matrix $A_\mathcal{G}$ give the connectivity information of nodes. Nodes v_i and v_j are considered as directly connected if they are grouped in the same cluster by k-means and are adjacent as well in the actual road network. Using this information the total number of connected components is computed for each ϱ^κ and the clustering configuration having the minimum number of connected components is finally selected as the optimal ϱ^θ. These components form the supernodes. A lesser number of supernodes makes the framework more scalable for large networks. Therefore in order to get fewer but informative supernodes, the method of supernode creation selects that clustering configuration among the short-listed ones as optimal, which leads to the lowest number of connected components. We apply the standard FIFO based connected components identification algorithm. Its computational complexity is $\mathbf{O}(\max(n_r, n_e))$, where n_r and n_e are the total number of nodes (road segments) and edges (adjacency relationships) in the road graph respectively.

All the connected components corresponding to ϱ_θ are then considered as supernodes to form the set $\mathcal{V}_s = \{\varsigma_1, \varsigma_2, \ldots, \varsigma_{n_\varsigma}\}$. Thus each cluster of nodes which is connected as well in $\mathcal{G}$ is accepted as a supernode. Feature value of each supernode is set as the mean of the cluster (given by k-means) to which they belong, i.e., $\forall \varsigma_i \in \mathcal{C}_j^\theta, \varsigma_i.f = \mu(\mathcal{C}_j^\theta)$.

Setting an appropriate optimality threshold value is crucial to the complexity of the overall algorithm. A lower value would lead to a large number of κ for which the MCG measure would be above the threshold, and would require computing and storing a large number of clustering configuration indicator vectors. It may sometimes lead to have fewer supernodes, but the cost of complexity for so many clustering indicator vectors has to be borne. On the other hand, a higher threshold would lead to fewer clustering indicator vectors but may result in producing more supernodes, which increases the complexity of the partitioning algorithm.



(a) Sample graph (b) Supernodes

Figure 3: Supernode stability check

4.3.2 An extension for supernode stability check

Sometimes the clustering and adjacency patterns of the road graph $\mathcal{G}$ may lead to form supernodes that do not guarantee the compactness and tightness of bonding that we want to impose. Figure 3 shows such an example, in which supernodes are formed from a sample graph after applying k-means with $\kappa = 2$ (which is optimal as per our MCG). The $\{11, 14\}$ set is loosely bonded. Considering it as a supernode will bind them together to belong to the same final partition. However, as it is connected to both 17 and 18 of the other cluster, it may suit 14 more to be with them in the final partition. Taking this matter into concern, here we present an extended method that determines the stability of created supernodes and splits them to make them reach a desired stability. We define a measure called *stability* that determines how much the nodes inside a supernode deserve to be together by looking into the tightness of bonding. The

closer the nodes in a supernode are, the higher will be its stability measure.

Algorithm 2: Supernode stability check(Supernode set $\mathcal{V}_s$, stability threshold ϵ_η)

```
1  stack ← initialize a stack;
   // push all the supernodes to check their stability
2  forall the ς_i ∈ V_s do
3      push ς_i into stack;
   // split the unstable supernodes until made stable
4  while stack is not empty do
5      ς_i ← pop from stack;
6      if η(ς_i) < ε_η then
7          ς_pre ← instantiate an empty supernode;
8          ς_post ← instantiate an empty supernode;
9          forall the v_j ∈ ς_i do
10             if v_j.f ≤ μ(ς_i) then
11                 add v_j to ς_pre ;
12             else
13                 add v_j to ς_post ;
14         V_s ← V_s \ ς_i, V_s ← V_s ∪ ς_pre ∪ ς_post;
15         push ς_pre into stack, push ς_post into stack;
16 return V_s;
```

DEFINITION 9: (**Supernode Stability**) The *stability* measure $\eta(\varsigma_i) \in [0,1]$ of a supernode is defined in Equation 2, where $|\varsigma_i|$ denotes the number of nodes v_j in ς_i and $\mu(\varsigma_i)$ denotes the mean of feature values $v_j.f$ of all nodes in ς_i. The supernode ς_i is said to be *stable* if its stability measure is greater than or equal to a pre-defined stability threshold ϵ_η, else it is said to be *unstable*. ∎

$$\eta(\varsigma_i) = \frac{1}{|\varsigma_i|} \times \sum_{v_j \in \varsigma_i} \exp\left(-\text{abs}\left(\frac{v_j.f + 1}{\mu(\varsigma_i) + 1} - 1\right)\right) \quad (2)$$

The above formulation looks into the distance of all nodes from the supernode centroid by the ratio of their feature values to their supernode mean values, and then takes their average value. The 1s are added in the numerator and denominator to avoid the zero values. It would yield its value as 1 when all the nodes inside the supernode have their feature values same as the supernode mean, and lower as much as the node feature values go far from the supernode mean up to 0. The LIFO based stability check algorithm is shown in Algorithm 2. All *stable* supernodes are accepted right away retaining their existing feature value $\varsigma_i.f$ (lines 5–6). However the *unstable* supernodes are further processed to split up into two parts from its centroid (lines 7–13). They are created as two independent supernodes and again checked to confirm their stability (lines 14–15). The interleaved steps of stability checking (lines 5–6) and their splitting into two supernodes from the centroid (lines 7–15) goes on indefinitely until all of them are made stable. The supernodes that were unstable earlier and made stable this way, their means become their new feature values, i.e. $\varsigma_i.f = \mu(\varsigma_i)$. Thus the newly formed set of supernodes is $\mathcal{V}_s = \left\{\varsigma_1, \varsigma_2, \ldots, \varsigma_{n'_\varsigma}\right\}$, where $n'_\varsigma \in [n_\varsigma, n_r]$. The exact gaps between n_ς and n'_ς, and n'_ς and n_r, depend on the imposed stability threshold ϵ_η. It has two extremes. When it is set to 1, the set of all nodes in the road graph $\mathcal{G}$ that have the same feature value $v_i.f$ as well as are linked directly by an edge, is accepted as a complete supernode. In extreme case n'_ς could be equal to n_r if no two nodes have the same feature value. On the other hand when it is set to 0, all connected components obtained using the optimal clustering configuration indicator vector ϱ_θ are considered as supernodes as if without any stability check where $n'_\varsigma = n_\varsigma$. Setting the stability threshold is based on the trade-off between quality and complexity. The worst case complexity of this task is $\mathbf{O}(2n_r - n_\varsigma)$ when all the supernodes are split up repeatedly until only single nodes are left in each supernode, whereas the best case complexity is $\mathbf{O}(n_\varsigma)$ when no supernode needs to be split.

4.3.3 Superlink establishment

For the obtained set of supernodes $\mathcal{V}_s$ and the available road graph $\mathcal{G}$, let $\mathcal{L}_{pq}$ be the set of links $\{e_j = link(v_x, v_y)\}$ existing between all $v_x \in \varsigma_p$ and all $v_y \in \varsigma_q$. A superlink ε_i is established between each pair of supernodes $(\varsigma_p, \varsigma_q)$, for which the condition $\mathcal{L}_{pq} \neq \phi$ is fulfilled (Algorithm 1, lines 21–25). The set of superlinks is denoted by $\mathcal{E}_s = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{n_\varepsilon}\}$. At the same time, each superlink ε_i is weighted by a value $\omega_i \in [0,1]$ in Equation 3, where $|\mathcal{L}_{pq}|$ denotes the number of links in $\mathcal{L}_{pq}$, $\varsigma_p.f$ and $\varsigma_q.f$ are the feature values of ς_p and ς_q respectively, and $\sigma^2(\varsigma) = \frac{1}{n_\varsigma} \times \sum_{i=1}^{n_\varsigma} \left(\varsigma_i.f - \mu^0\right)^2$ is the variance of supernode features with respect to the global mean μ^0.

$$\omega_i = \sqrt{\frac{1}{|\mathcal{L}_{pq}|} \times \sum_{e_j \in \mathcal{L}_{pq}} \left(\exp\left(\frac{-(\varsigma_p.f - \varsigma_q.f)^2}{2 \times \sigma^2(\varsigma)}\right)\right)^2} \quad (3)$$

The above formulation is in the form of a Gaussian function that assigns a similarity measure between the two supernodes of each linked pair, and takes their average value. The effect of individual links on the weight is high if the supernodes connected by the link have closer feature values, and low otherwise. The normalization by $|\mathcal{L}_{pq}|$ normalizes the bias towards the supernode pairs having large number of links but highly dissimilar feature values. Thus the overall weight considers both the number of individual links between the participating supernodes and their feature values, where larger number of links and closer supernode feature values together lead to higher superlink weight. The set of computed weights associated with each superlink is denoted by $\mathcal{W}_s = \{\omega_1, \omega_2, \ldots, \omega_{n_\varepsilon}\}$. Hence the supergraph mining step becomes complete producing $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s, \mathcal{W}_s)$ (Algorithm 1, lines 26–27).

5. ROAD SUPERGRAPH PARTITIONING

Although a preliminary level of grouping of road segments has already happened in the form of supernodes, the number of supernodes still can be very large. Moreover, the linkages that represent the spatial associations have remained underutilized, as they have just directly been employed in supernode creation until this stage. This step aims to group the set of supernodes into different partitions in a top-down manner by using the superlinks by which they are connected. These different supernode partitions are obtained as connected within, and this in turn achieves the ultimate objective of getting partitions as node partitions where spatial adjacencies are maintained.

5.1 Spectral Clustering for Partitioning

Spectral clustering treats clustering as a graph partitioning problem. In our case, we already have a graph that we want to partition. Among the existing graph cuts, normalized cut has been found to be comparatively effective for graph partitioning, due to the reason that it optimizes both the intra-partition homogeneity and inter-partition heterogeneity at the same point [11, 5]. Its optimization 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 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. In this function, both the numerator and denominator take into account just the weighted links, and no consideration is made for the node groupings (or node counts) in the resulting partitions. The links in our road graph are established only if they are adjacent in the road network, and thus the superlinks 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 propose a novel k-way graph cut. Instead of repeated bipartitioning of the whole graph, it produces $k'(> k)$ partitions in just a single iteration, and then applies recursive bipartitioning to produce k partitions, which significantly improves its efficiency.

5.2 The k-way α-Cut

For a given weighted graph, which in our case is the supergraph $\mathcal{G}_s$, let us suppose its supernode 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 $\mathcal{G}_s$ is denoted by A, the *degree matrix* is denoted by D, which is a diagonal matrix having row sums of A at the diagonal, and the *Laplacian matrix* $(D - A)$ is denoted by L. A function $W(\mathcal{P}_i, \mathcal{P}_j)$ is defined in Equation 4 as the sum of weights associated with all the superlinks having its supernode at one end in $\mathcal{P}_i$ and the supernode at the other end in $\mathcal{P}_j$.

$$W(\mathcal{P}_i, \mathcal{P}_j) = \sum_{\varepsilon_r \in \{SLinks(\mathcal{P}_i,\mathcal{P}_j)\}} \omega_r = \sum_{\varsigma_p \in \mathcal{P}_i, \varsigma_q \in \mathcal{P}_j} A(p,q) \quad (4)$$

Definition 10: (**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 superlinks having their supernodes at one end in $\mathcal{P}_i$ and supernodes at other end in any partition other than $\mathcal{P}_i$, i.e., $W(\mathcal{P}_i, \overline{\mathcal{P}_i})$. ■

Definition 11: (**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 superlinks having supernodes 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 superlink connections while partitioning the graph. When this value is divided by the number of supernodes in $\mathcal{P}_i$, it gives the average contribution of each supernode 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 supernodes in $\mathcal{P}_i$, it gives the average contribution of each supernode 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 [11]. However, optimizing any one of these objectives does not guarantee the other. One possible approach is that of normalized cut [11]. It minimizes inter-partition similarity and maximizes intra-partition similarity simultaneously. But that optimization is based on normalized values of cut and association, where the normalization considers the link connectivities between nodes, instead of the nodes directly, and it does not guarantee the optimization of their average cut and association.

To achieve a well balanced optimization of both average cut and average association, in this paper we design a novel k-way graph cut called α-Cut. It aims to achieve a partitioning configuration optimized by the objective function $\min_{\mathcal{P}} \alpha\text{-}Cut(\mathcal{P})$, where $\alpha\text{-}Cut(\mathcal{P})$ is shown in Equation 5.

$$\alpha\text{-}Cut(\mathcal{P}) = \sum_{i=1}^{k} \left(\alpha \times \frac{W(\mathcal{P}_i, \overline{\mathcal{P}_i})}{|\mathcal{P}_i|} - (1-\alpha) \times \frac{W(\mathcal{P}_i, \mathcal{P}_i)}{|\mathcal{P}_i|} \right) \quad (5)$$

It minimizes a combination of two components, which can be separated as minimization of average *cut* representing inter-partition similarity, and maximization of average *association* representing intra-partition similarity. The factor $\alpha \in [0, 1]$ acts as a balance between the two parts, and its value is crucial to obtain the best possible optimized partitions.

5.3 Determining α in α-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 constant value is its dynamic adjustment depending on the nature of the concerned partition. We consider this factor α_i as the portion of connectivity weight contributed by $\mathcal{P}_i$ in the whole supergraph (including intra-connections as well as inter-connections), and define it as the ratio of the summation of its superlink connection weights to the summation of all superlink connection weights in the supergraph, i.e., $\alpha_i = \frac{W(\mathcal{P}_i,\mathcal{V}_s)}{W(\mathcal{V}_s,\mathcal{V}_s)}$. Its value ranges from 0 to 1. On the other hand, $(1 - \alpha_i)$ gives the portion of the connectivity weight contributed by all partitions other than $\mathcal{P}_i$. Putting this value of α_i in Equation 5, α-Cut simplifies as below.

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

Like normalized cut [11], the problem to achieve a partitioning configuration which 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.

5.4 Spectral Clustering Approach to α-Cut

If $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k\}$ is the set of k disjoint partitions of $\mathcal{G}_s$, let $\mathbf{1} \in \mathbb{R}^{n_\varsigma}$ be a vector with each of its values as 1, and $c_i \in \mathbb{R}^{n_\varsigma}$ 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.

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 as follows.

$$\begin{aligned} \alpha\text{-}Cut(\mathcal{P}) = & \sum_{i=1}^{k} \frac{1}{c_i^T c_i} \times \left(\frac{\left(\mathbf{1}^T D c_i\right)^2}{\mathbf{1}^T D \mathbf{1}} - c_i^T A c_i \right) \\ = & \sum_{i=1}^{k} \frac{c_i^T M c_i}{c_i^T c_i}, \text{where } M = \left(\frac{\left(\mathbf{1}^T D\right)^T \left(\mathbf{1}^T D\right)}{\mathbf{1}^T D \mathbf{1}} - A \right) \end{aligned} \quad (6)$$

Algorithm 3: α-Cut Partitioning(Supergraph $\mathcal{G}_s$, number of desired partitions k)

1 $A \leftarrow$ adjacency matrix of $\mathcal{G}_s$;
2 $D \leftarrow$ degree matrix of $\mathcal{G}_s$;
3 $M \leftarrow \left(\frac{\left(\mathbf{1}^T D\right)^T \left(\mathbf{1}^T D\right)}{\mathbf{1}^T D \mathbf{1}} - A \right)$; `// get the α-Cut matrix`
4 $\bigcup_{i=1}^{n_\varsigma} \{(y_i, \lambda_i)\} \leftarrow$ get eigenvector and eigenvalue pairs of M;
5 sort eigenvalues λ_i to have $\lambda_{n_\varsigma} \leq \lambda_{n_\varsigma - 1} \leq \cdots \leq \lambda_1$;
6 select $\{\lambda_{n_\varsigma}, \lambda_{n_\varsigma - 1}, \ldots, \lambda_{n_\varsigma - k + 1}\}$ eigenvalues and corresponding eigenvectors $\{y_{n_\varsigma}, y_{n_\varsigma - 1}, \ldots, y_{n_\varsigma - k + 1}\}$;
7 generate matrix $Y_{n_\varsigma \times k} = \begin{pmatrix} y_1 & y_2 & \ldots & y_k \end{pmatrix}$;
8 $Z \leftarrow$ row normalize Y;
9 $\{z_1, z_2, \ldots, z_{n_\varsigma}\} \leftarrow$ get row vectors of Z;
10 $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\} \leftarrow k\text{-}means\left(\{z_1, z_2, \ldots, z_{n_\varsigma}\}, k\right)$;
11 $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_{k'}\} \leftarrow$ get disjoint partitions from $\mathcal{C}$;
`// global recursive bipartitioning to obtain k partitions`
12 $A'_{k' \times k'} \leftarrow$ compute partition connectivity matrix of $\mathcal{P}$;
13 *queue* $\leftarrow$ initialize a queue;
14 partition set $\mathcal{P} \leftarrow$ initialize with a single partition of A';
15 enqueue A' into *queue*;
16 **repeat**
17 $A' \leftarrow$ dequeue from *queue*;
18 $(\mathcal{P}_1, \mathcal{P}_2) \leftarrow$ bipartition A' using α-Cut;
19 $A'_1 \leftarrow$ create adjacency matrix for $\mathcal{P}_1$;
20 $A'_2 \leftarrow$ create adjacency matrix for $\mathcal{P}_2$;
21 enqueue A'_1 into *queue*, enqueue A'_2 into *queue*;
22 $\mathcal{P} \leftarrow \mathcal{P} \setminus$ partition of A';
23 $\mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{P}_1 \cup \mathcal{P}_2$;
24 **until** *number of partitions in* $\mathcal{P}$ *equals to* k;
25 **return** $\mathcal{P}$;

The derived matrix M is called the α-Cut matrix for $\alpha_i = \frac{W(\mathcal{P}_i, \mathcal{V}_s)}{W(\mathcal{V}_s, \mathcal{V}_s)}$, and the spectral clustering algorithm works on this matrix. Equation 6 is further simplified as $\sum_{i=1}^{k} \frac{c_i^T M c_i}{\|c_i\|^2} = \sum_{i=1}^{k} \left(\frac{c_i}{|c_i|}\right)^T M \left(\frac{c_i}{|c_i|}\right) = \sum_{i=1}^{k} y_i^T M y_i$, 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

$$\min_{\mathcal{P}} \sum_{i=1}^{k} y_i^T M y_i \text{ subject to } y_i^T y_i = 1 \tag{7}$$

This can be 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 [14]. Following the method in [14], the objective is achieved by selecting the 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.

Algorithm 3 presents the complete partitioning method, where the α-Cut matrix is computed in line 3 and eigen-decomposed in line 4. Lines 5–6 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 the binary pattern. Due to the lack of concrete information about clusters, it becomes another problem to separate the k clusters. Here we assume that the clusters are well-separated in the k-dimensional eigenspace, which is a general assumption in spectral clustering [14], and use the eigenvectors (or indicator vectors) to generate a matrix Y of $n_\varsigma \times k$ dimensions (line 7). It is then row-normalized using Equation 8 to have row-vectors z_i of unit length giving the final matrix Z (line 8). Each row-vector z_i represents a supernode ς_i. The set of row-vectors are used to cluster the supernodes by applying k-means to find a set of k clusters $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$ (lines 9–10), where each cluster $\mathcal{C}_i$ comprises one or more row vectors (supernodes) in Z. The supernodes inside each cluster are linked together as they exist in the supergraph. Upon linking them, sometimes even more than one connected components may be found inside a single cluster. These connected components are extracted from each cluster to form disjoint partitions (line 11).

$$Y = \begin{pmatrix} y_{11} & y_{21} & \cdots & y_{k1} \\ y_{12} & y_{22} & \cdots & y_{k2} \\ | & | & & | \\ y_{1n} & y_{2n} & \cdots & y_{mn} \end{pmatrix} = \begin{pmatrix} — & z_1^T & — \\ — & z_2^T & — \\ & \vdots & \\ — & z_n^T & — \end{pmatrix} = Z \tag{8}$$

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

Depending on data, the number of disjoint partitions may sometimes be large, which would yield the partition set from $\mathcal{C}$ 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, there are two approaches as described in [11]– *i*) greedy pruning, and *ii*) 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. Therefore we follow the global recursive bipartitioning approach.

It begins with computing a partition connectivity matrix A' of dimension $k' \times k'$ (line 12). Its values are computed as connectivity strengths between the partitions $A'(i,j) = \sqrt{\frac{1}{numadj(\mathcal{P}_i, \mathcal{P}_j)}} \times \sum_{\varsigma_p \in \mathcal{P}_i, \varsigma_q \in \mathcal{P}_j} \left(A\left(p,q\right)\right)^2$, where $numadj(\mathcal{P}_i, \mathcal{P}_j)$ gives the number of supernode adjacency relationships between the supernodes of $\mathcal{P}_i$ and $\mathcal{P}_j$. This value automatically becomes zero for the pair of partitions that do not share any adjacency relationship. We use a queue to recursively apply the bipartitioning (lines 13–24). The α-Cut algorithm is applied on A' to have two partitions (line 18). Nodes of A' (old partitions) belonging to each partition (new partition) are separated, and two new matrices are created by separating the corresponding rows and columns of A', such that the sum of dimensions of the two new matrices equals to the dimension of A' (lines 19–20). Each matrix now represents one partition. The bipartitioning is again applied on each matrix individually to yield more than two partitions. These interleaved steps of bipartitioning and matrix creation are repeated until the total number of final partitions equals k (lines 16–24).

The computational complexity of eigen-decomposition is $\mathbf{O}(n^3)$ in general and $\mathbf{O}(n^2)$ for sparse matrices. The application of k-means on row-vectors to find the clusters costs $\mathbf{O}(tnk^2)$. In these costs, $n = n_\varsigma$ when the spectral clustering is applied on the supergraph, and $n = n_r$ when it is applied directly on the road graph.

6. EXPERIMENTAL EVALUATION

In this section, we evaluate the proposed framework in terms of different evaluation metrics. Although there exist many works on general graph partitioning, we compare our results to a recent work [5], which is on the same problem, for a close and specific comparison.

6.1 Datasets

We perform experiments on two kinds of datasets, small ($\mathtt{D}_1$) and large ($\mathtt{M}_1$, $\mathtt{M}_2$, and $\mathtt{M}_3$) road networks. Table 1 shows the statistics of all these datasets. The traffic on the small network, shared by the authors of [5], 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 vehicle/metre. In this work we perform experiments at $t = 71$ to compare our results with [5] which used the same dataset.

Table 1: Dataset statistics

	$\mathtt{D}_1$	$\mathtt{M}_1$	$\mathtt{M}_2$	$\mathtt{M}_3$
Place	Downtown San Francisco	CBD Melbourne	CBD(+) Melbourne	Melbourne
Area (sq. ml.)	2.5	6.6	31.5	42.03
Road seg	420	17,206	53,494	79,487
Intersection pt	237	10,096	28,465	42,321

The traffic data for the large networks is generated by a web-based[2] random road traffic generator MNTG [10]. We populate $\mathtt{M}_1$, $\mathtt{M}_2$, and $\mathtt{M}_3$ by 25,246, 62,300, and 84,999 vehicles respectively, and obtain their trajectories for 100 continuous timestamps. 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/metre) at each point of time.

6.2 Evaluation Metrics

The partitioning framework is evaluated using metrics that quantify the quality of 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, **C.1** and **C.2** are automatically fulfilled. **C.3** which enforces inter-partition heterogeneity is evaluated by the *inter* metric[3]. 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. **C.4** which enforces intra-partition homogeneity is evaluated by the *intra* metric[4]. 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.

Additionally, we also evaluate the overall partitioning. The standard metrics of cluster evaluation do not take the associated spatial adjacencies into account. For its proper evaluation, we use two metrics derived from the standard cluster evaluation metrics to make them suitable for the graph partitioning problem. They are the graph Davies-Bouldin index (GDBI)[5] based on Davis-Bouldin index (DBI), and the average NcutSilhouette (ANS) measure defined in [5] especially for partition evaluation. In both these measures, smaller values indicate better partitioning.

6.3 Experimental Results on Small Networks

We perform experiments on the small road network $\mathtt{D}_1$ to compare the partitioning quality of our α-Cut based partitioning framework with other state-of-the-art techniques using performance evaluation metrics listed in Section 6.2, and demonstrate its effectiveness. For an exhaustive analysis from different perspectives we present the results obtained on several different schemes. Here we introduce the notations used for those schemes. `AG` and `NG` are the schemes when α-Cut and normalized cut are applied directly on the road graph respectively, and `ASG` and `NSG` are the schemes when α-Cut and normalized cut are applied on the road supergraph with no stability check respectively.

Results in this section are the median values of evaluation metrics obtained from 100 execution of the algorithm. The reason is that k-means (used to cluster eigenvectors) may sometimes produce slightly different results in different executions because of randomized cluster initialization.

We consider `NG` as the baseline, and comparatively show our results. Figure 4 shows the complete results of `AG` and `ASG` in comparison to `NG` in terms of evaluation metrics for the number of partitions k ranging from 2 to 20. GDBI and ANS measures quantify the overall partitioning quality. In terms of both these measures, both `AG` and `ASG` schemes of our framework outperform `NG` at all values of k. Also in terms of *intra*, that quantifies intra-partition homogeneity, we outperform `NG`. In terms of *inter*, that quantifies inter-partition heterogeneity, `AG` outperforms `NG` at all values except $k = 2$, whereas `ASG` outperforms at all values.

The overall partitioning quality is evaluated by GDBI and ANS, which consider both the inter-partition heterogeneity and intra-partition homogeneity simultaneously. As stated earlier, lower values indicate better partitioning for both these measures. In both the Figures 4(c) and 4(d), `AG` is much lower than `NG` at all values of k. The GDBI measure increases with increasing k, but this is not the case with ANS. In [5], the authors used the ANS measure to learn the number of optimal partitions. They accept the value of k that leads to the ANS minima as the optimal number of partitions, which in this case is 6 for `AG` and 8 for `NG`. The minima of `AG` being much lower than that of `NG`, is found as the better performer. As the schemes `AG` and `NG` applies α-Cut and normalized cut respectively directly on the graph, the obtained results shows the superiority of α-Cut.

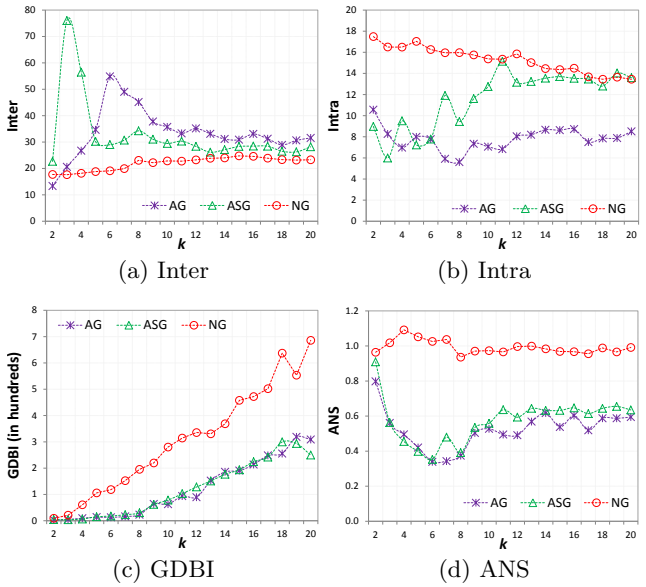
The supergraph technique, as said earlier, is to make the framework applicable for large road networks, in which we may need to compromise the quality to some extent. Here we show the effects of the supergraph on the partitioning quality. Figure 6(a) shows the stability measures $\eta(\varsigma)$ of the 105 supernodes. When the stability threshold $\epsilon_\eta = 0$, the partitioning scheme behaves as `ASG`, whereas $\epsilon_\eta = 1$ makes it behave as `AG`. The figure, which also presents the results

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

[3] $Inter(\mathcal{P}) = \frac{1}{\text{count}(adj)} \times \sum_{\mathcal{P}_i, \mathcal{P}_j \in \mathcal{P}, \mathcal{P}_i \xleftrightarrow{\text{adj}} \mathcal{P}_j} \frac{\sum_{v_p \in \mathcal{P}_i} \sum_{v_q \in \mathcal{P}_j} \text{abs}(v_p.f - v_q.f)}{|\mathcal{P}_i| \cdot |\mathcal{P}_j|}$

[4] $Intra(\mathcal{P}) = \frac{1}{|\mathcal{P}|} \times \sum_{\mathcal{P}_i \in \mathcal{P}} \frac{\sum_{v_p, v_q \in \mathcal{P}_i, p \neq q} \text{abs}(v_p.f - v_q.f)}{|\mathcal{P}_i| \cdot (|\mathcal{P}_i| - 1)}$

[5] $GDBI(\mathcal{P}) = \frac{\sum_{\mathcal{P}_i \in \mathcal{P}} \sum_{\mathcal{P}_j \in \text{neigh}(\mathcal{P}_i)} \left\{ \frac{\text{S}(\mathcal{P}_i) + \text{S}(\mathcal{P}_j)}{\text{S}(\mathcal{P}_i, \mathcal{P}_j)} \right\}}{|\mathcal{P}|}$, where $\text{neigh}(\mathcal{P}_i)$ returns the partitions that are spatially adjacent to $\mathcal{P}_i$, $\text{S}(\mathcal{P}_i)$ returns the average distance of nodes in $\mathcal{P}_i$ from its mean, and $\text{S}(\mathcal{P}_i, \mathcal{P}_j)$ returns the distance between their means.



(a) Inter (b) Intra

(c) GDBI (d) ANS

Figure 4: Road graph and supergraph partitioning results in small networks

obtained by ASG, shows how the inclusion of the supergraph technique affects the partitioning. The GDBI and ANS plots show that partitioning a network by ASG is qualitatively almost the same as that by AG. However, AG generally lies below ASG in the ANS plot indicating its superiority. At few points ($k = 4, 5$) in Figure 4(d), ASG is better than AG. The reason is that those values do not suit the dataset for partitioning and therefore sometimes it takes place arbitrarily. Applying the stability check on the supergraph with any value between 0 to 1 as its threshold results in a partitioning that is qualitatively between AG and ASG.

To summarize the overall results, like Ji and Geroliminis [5], we consider the ANS measure as the deciding factor for the optimal number of partitions. We now compare the best (lowest) ANS measures (which gives the optimal partitioning) of all the schemes along with [5]. Table 2 shows that both of our schemes AG and ASG are much lower (better) than NG and [5]. As Ji and Geroliminis perform additional adjustments after partitioning by normalized cut, their partitions are somewhat improved in quality than NG, but even then our method outperforms theirs.

Table 2: Overall quality of partitioning

Scheme	ANS	k	Scheme	ANS	k
AG	**0.3392**	6	NG	0.9362	8
ASG	0.3526	6	Ji and Geroliminis [5]	0.6210	3

We can also look into the partitioning quality more closely in Figures 4(a) and 4(b), which show the inter-partition and intra-partition distances separately. As we want to obtain a partition set having the highest possible inter-partition distances, higher values of *inter* indicate better partitioning. Except $k = 2$, at all values of k in the range, AG has higher values than NG. Thus if the optimal number of partitions for this data comes out to be 2, which is not true (found as 6 in previous paragraphs), NG outperforms AG in terms of this measure. The value of AG increases rapidly until $k = 6$, which is the maxima. After that point it decreases rapidly again, and gradually comes to relatively stable values. The maxima of AG for *inter* lies at $k = 6$ which coincides with the minima of ANS. Another perspective to evaluate the partitioning is to look into the intra-partition distances using *intra*. As our objective is to minimize intra-partition distances, lower values are an indicator of better partitioning. In the figure we can see that at all values of k in the range, AG has lower values than NG.

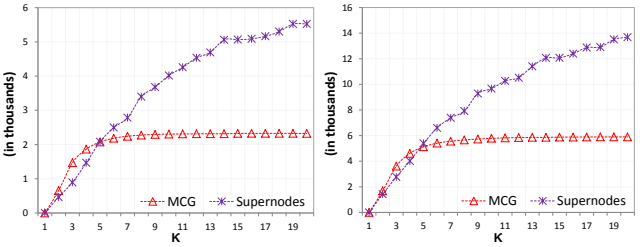
In the curve of ASG of *inter*, there is a sudden rise at $k = 2$, but then it comes down in between AG and NG. Similarly *intra* fluctuates over the initial values of k, after which it comes in between the other two, whereas at higher values its trend becomes similar to NG. The reason for the abnormal behavior at the initial values is that they do not suit the dataset for partitioning by the α-Cut. When the partitioning is applied at those values, it takes place arbitrarily for some instances, which makes it behave abnormally. As is evident from the ANS plot, the initial values of k are not so good for partitioning.

6.4 Experimental Results on Large Networks

We perform experiments on large road networks M_1, M_2 and M_3 to validate the scalability of our framework. Additionally, we also show that the quality of partitioning large networks is comparable to that of partitioning small networks.

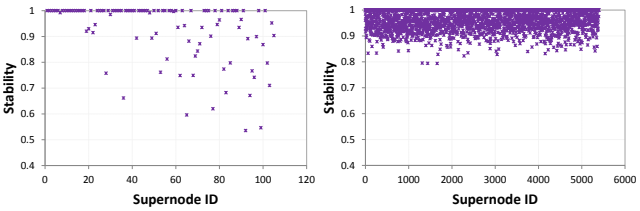
Figure 5 shows the MCG measures and the number of supernodes obtained from the cluster sets produced by k-means at different values of κ on M_1 and M_2. At the initial values, the MCG measure rises steeply up to some point, beyond which there is little change. In case of M_1, the maxima 2326.88 is attained at $\kappa = 18$, after which it starts declining gradually, but the major rise is only up to $\kappa = 5$. As higher MCG measures indicate better clustering, the best quality cluster set of M_1 is obtained at $\kappa = 18$, but those obtained at lower values, up to $\kappa = 5$ with an MCG measure of 2075.16, do not differ much in quality. If we look into the number of obtained supernodes, it increases monotonically with the increasing value of κ. As having larger number of supernodes adds on complexity to the remaining partitioning task, it is worth choosing the value of κ after which there is little increase in MCG. We get this value by fixing the optimality threshold ϵ_θ to 2000 for M_1 and 5000 for M_2. It leads to an optimal κ of 5 for both datasets, and the obtained number of supernodes are 2,081 and 5,391 respectively. Thus our supergraph technique reduces the adjacency matrix dimension from 17,206 and 53,494 to 2,081 and 5,391 for M_1 and M_2 respectively. Similarly, the optimal κ for M_3 is found as 5, which produces 9179 supernodes. This significantly reduces both the space and time complexity, and if required the complexity can further be reduced by selecting a lower κ, in which the partitioning quality may degrade to some extent. Figure 6(b) shows the supernode stability measures of 5391 supernodes of the M_2 dataset. We can see that most supernodes are highly stable. Therefore we proceed with these supernodes for supergraph construction and its partitioning by α-Cut.

Figure 7 shows the final partitioning results obtained for all the three large datasets. The plots show the measures of respective metrics in Y-axis at different values of k. As shown in Figures 7(b), 7(d), and 7(f), we get the best (lowest) ANS measures of 0.423 at $k = 4$, 0.511 at $k = 5$, and 0.512 at $k = 5$ on M_1, M_2, and M_3, respectively. These values are not as good as we found on small networks (AG- 0.3392



(a) M_1 (b) M_2

Figure 5: MCG measure and number of supernodes in large networks



(a) D_1 (b) M_2

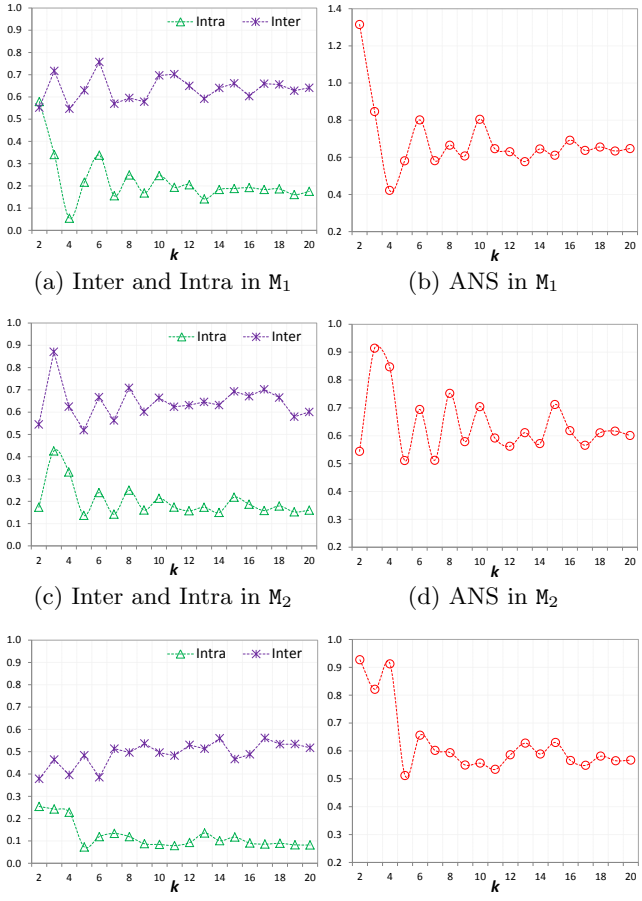
Figure 6: Stability measure of supernodes

and ASG- 0.3526) in Section 6.3, but still they are much better than the small network baseline results (NG- 0.9362, [5]- 0.6210). Moreover, results indicate that the partitioning of M_1 is qualitatively better than that of M_2 and M_3, but worse than that of D_1. It shows that as the size of the road network increases, the partitioning quality decreases.

As we get the lowest ANS measure for M_1 at $k = 4$, the best possible way to partition this network is to divide it into 4 segments as produced by our framework, each of which exhibit distinctive traffic congestion inside. However, if the congestion pattern has to be analyzed more closely, we can also have more partitions, and $k = 7, 9, 13, \ldots$ being the local minima serve as good candidates for the number of partitions. Similarly, some other suitable candidates for having a good congestion-based partitioning are $k = 7, 9, 12, 14, \ldots$ for M_2, and $k = 9, 11, 14, 17, \ldots$ for M_3.

At lower values of k, a small change makes a big effect in the partitioning quality, as can be seen from the fluctuations in Figures 7(b), 7(d) and 7(f). However, as k becomes large, the fluctuations diminish. The reason behind this is that at smaller values of k, say 2, when it is increased to 3, a large re-arrangement takes place inside the partitions. It would be much larger than the re-arrangement that takes place at higher values of k, say when it increases from 22 to 23. Unlike the results of the small network, the *intra* and *inter* measures here are very small. The reason is that the road segment densities in M_1 and M_2 are much lower than those in D_1, and those in M_3 is even lower than all.

For large road networks, the most time-taking task in the framework is the eigen-decomposition (Algorithm 3, line 4). This becomes a major overhead when dimension of the matrix M becomes large. We overcome this issue (up to some extent) in our study by applying a high performance algorithm developed and used in Matlab [3]. It reduces of the original matrix to a condensed form by orthogonal transformations, decomposes the matrix, and then transforms it back. Table 3 shows the running time of our framework in number of seconds consumed in its complete execution.



(a) Inter and Intra in M_1 (b) ANS in M_1

(c) Inter and Intra in M_2 (d) ANS in M_2

(e) Inter and Intra in M_3 (f) ANS in M_3

Figure 7: Road supergraph partitioning results in large networks

The total time has been broken down to show the individual times consumed int the different modules, where the different modules are those described in Section 3. For the small dataset D_1, it takes just fractions of a second to complete, whereas for large datasets M_1, M_2, and M_3 it takes 2.15 minutes, 31.75 minutes, and 1.64 hours, respectively. Module 1 takes the lowest amount of time, whereas module 3, which includes the eigen-decomposition task, takes the highest amount of time. The total time taken for M_3 is certainly very high, and therefore while applying repeated partitioning on an urban road network, at the beginning it can be started by partitioning the whole network. But after having its relatively small partitions, they can be repeatedly subjected to partitioning distributively with the changing congestion measures with respect to time. In this way it helps in reducing the running time, and can even be applied in real-time if the network becomes as small as M_1.

Table 3: Running Time (in seconds)

Module	D_1	M_1	M_2	M_3
1	less than 1	9	24	137
2	less than 1	54	848	2044
3	less than 1	66	1033	3726
Total	less than 1	129	1905	5907

7. RELATED WORK

In recent years, the growth in multidimensional geospatial datasets has attracted the attention of spatial database researchers to address the problems of transportation systems [4, 1]. However, little work has been done on spatial partitioning of urban road networks. In a recent work [5], Ji and Geroliminis 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 works in three steps, starting with excessive partitioning of the road network using normalized cut, followed by merging smaller partitions up to a certain level, and then locally adjusting the road segments lying on partition boundaries by replacing them into the neighboring partitions, if doing so increases the segment uniformity. Their method can suffer from time and space complexity for large road networks.

This problem is very much related to general graph partitioning, which is a well studied problem. It has applications to a wide variety of areas, and many solutions have been proposed in the past. Spectral clustering algorithms like minimum cut and normalized cut have remained quite popular [11, 13]. In [2], the authors proposed a spectral cut based on the min-max clustering principle for graph partitioning in a data clustering point of view.

The modularity of a set of graph partitions is defined as the difference between the observed and expected fraction of links within a partition. Larger modularity values are correlated with better graph partitioning. To maximize modularity while partitioning a graph, in [13] 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 the modularity matrix. This matrix actually equals to the negative of our α-Cut matrix derived in Equation 6. As we obtain the partitioning by selecting the k smallest eigenvalues and corresponding eigenvectors, both the techniques result in the same set of eigenvalues and eigenvectors, and thus the same partitioning. It means that the minimization of α-Cut approximately maximizes the modularity.

As it is an NP-complete problem, multilevel and heuristic algorithms have also been studied [8]. Zhou et al. [15] 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. [12] 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 also a wide application of graph partitioning for network community detection. In [9], the authors explored some community detection methods and evaluated their relative performances. However, most the works on graph partitioning face time and space complexity issues with large networks.

8. CONCLUSION

In this paper, we presented a spectral clustering based framework for traffic congestion-based spatial partitioning of large urban road networks. We first formally gave a mathematical representation of actual road networks and transformed the road network into a road graph, and then to a road supergraph by clustering the node feature values. The novel k-way α-Cut partitioning algorithm is applied on the supergraph to obtain k partitions. The mining of the supergraph leads to a preliminary grouping of road segments in the form of supernodes, which significantly reduces the partitioning load. This technique makes the framework scalable and suitable to handle the rapidly growing urban road networks. The α-Cut algorithm, proposed in this paper, aims to achieve a good balance of average cut and average association through spectral clustering. This algorithm also approximately maximizes the network modularity. In our experiments, we found that it produces partitions qualitatively better than normalized cut. We performed experiments on a small road network of Downtown San Francisco to demonstrate the framework effectiveness, and on three large road networks of Melbourne of different sizes to demonstrate its scalability along with effectiveness. In all the four networks, we outperform the existing techniques in terms of different performance evaluation metrics.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. This research was supported by the Australian Research Council (ARC) Discovery Project DP120102627 and the ARC Future Fellowship grant FT120100723.

9. REFERENCES

[1] Z. Chen, H. T. Shen, and X. Zhou. Discovering popular routes from trajectories. In *Proc. of the ICDE*, pages 900–911, 2011.

[2] C. Ding, X. He, H. Zha, M. Gu, and H. Simon. A min-max cult algorithm for graph partitioning and data clustering. In *Proc. of the ICDM*, pages 107–114, 2001.

[3] J. J. Dongarra, D. C. Sorensen, and S. J. Hammarling. Block reduction of matrices to condensed forms for eigenvalue computations. *Journal of Computational and Applied Mathematics*, 27(1–2):215–227, Sept. 1989.

[4] H. Gonzalez, J. Han, X. Li, M. Myslinska, and J. P. Sondag. Adaptive fastest path computation on a road network: a traffic mining approach. In *Proc. of the VLDB*, pages 794–805, 2007.

[5] Y. Ji and N. Geroliminis. On the spatial partitioning of urban transportation networks. *Transportation Research Part B: Methodological*, 46(10):1639–1656, 2012.

[6] Y. Jung, H. Park, D.-Z. Du, and B. L. Drake. A decision criterion for the optimal number of clusters in hierarchical clustering. *J. of Global Optimization*, 25(1):91–111, Jan. 2003.

[7] E. Kanoulas, Y. Du, T. Xia, and D. Zhang. Finding fastest paths on a road network with speed patterns. In *Proc. of the ICDE*, 2006.

[8] M. Kim and K. S. Candan. Sbv-cut: Vertex-cut based graph partitioning using structural balance vertices. *Data Knowl. Eng.*, 72:285–303, Feb. 2012.

[9] J. Leskovec, K. J. Lang, and M. Mahoney. Empirical comparison of algorithms for network community detection. In *Proc. of the WWW*, pages 631–640, 2010.

[10] M. F. Mokbel, L. Alarabi, J. Bao, A. Eldawy, A. Magdy, M. Sarwat, E. Waytas, and S. Yackel. Mntg: an extensible web-based traffic generator. In *Proc. of the SSTD*, pages 38–55, 2013.

[11] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, Aug. 2000.

[12] Y. Sun, J. Han, P. Zhao, Z. Yin, H. Cheng, and T. Wu. Rankclus: Integrating clustering with ranking for heterogeneous information network analysis. In *Proc. of the EDBT*, pages 565–576, 2009.

[13] S. White and P. Smyth. A spectral clustering approach to finding communities in graph. In *Proc. of the SDM*, 2005.

[14] M. J. Zaki and W. Meira Jr. *Data Mining and Analysis: Fundamental Concepts and Algorithms*, pages 421–435. Cambridge University Press, 2014 (to appear).

[15] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. *Proc. VLDB Endow.*, 2(1):718–729, Aug. 2009.