Network Clustering in Vehicular Communication Networks

by

Weiwei Li

A thesis submitted in conformity with the requirements for the degree of Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto

Copyright © 2011 by Weiwei Li
Abstract

Network Clustering in Vehicular Communication Networks

Weiwei Li

Master of Applied Science

Graduate Department of Electrical and Computer Engineering

University of Toronto

2011

This thesis proposes a clustering algorithm for vehicular communication networks. A novel clustering metric and an improved clustering framework are introduced. The novel clustering metric, network criticality, is a global metric on undirected graphs which quantifies the robustness of the graph against changes in environmental parameters, and point-to-point network criticality is also defined to measure the resistance between different points of a graph. We localize the notion of network criticality for a node of a vehicular network which can potentially be promoted as the cluster header. We use the localized notion of node criticality in conjunction with a universal link metric, Link Expiration Time (LET), to derive a clustering algorithm for the vehicular network. We employ a distributed multi-hop clustering algorithm based on the notion of network criticality. Simulation results show that the proposed clustering algorithm forms a more robust cluster structure.
Dedication

To my parents
Acknowledgements

I would like to thank all people who have helped me during my master program.

I especially want to record my sincere gratitude to my supervisor, Professor Alberto Leon-Garcia, for his supervision, guidance, and encouragement.

I would also like to thank my parents for their endless love and support since I was born.

I gratefully thank the members of my committee, Professor Elvino Sousa, Professor Shahrokh Valaee, and Professor Raymond H. Kwong for their help and support.

Many thanks to my family and sincere friends who are always accompanying me: Ali Tizghadam, Armin Ghayooru, Khashayar Khavari, Leila Shayanpour, and all the rest. I would like to thank all of them for your help and the happiness we sharing together.
# Contents

1 Introduction  
  1.1 Emergence of Connected Vehicles .......................... 1  
  1.2 Connected Vehicular Network Architecture ..................... 3  
  1.3 Clustering of Vehicular Network .......................... 6  
  1.4 Network Model of This Thesis .......................... 10  
  1.5 Contribution of This Thesis .......................... 11  

2 Overview and Comparison of Vehicular Clustering...  
  2.1 Preliminaries ............................................. 13  
  2.1.1 Distance ............................................. 13  
  2.1.2 The Unnormalizzed Laplacian Matrix ..................... 14  
  2.2 Clustering Algorithms Based on Weighted Graph ............ 15  
  2.2.1 Hierarchical Clustering ................................ 16  
  2.2.2 K-Means Clustering ................................ 18  
  2.2.3 K-Medoids Clustering Algorithm ..................... 21  
  2.2.4 Spectral Clustering ................................ 23  
  2.3 Clustering Algorithms in MANETs .................. 24  
  2.3.1 Lowest-ID and Highest-Degree Clustering .................. 24  
  2.3.2 MOBIC ............................................. 25  
  2.3.3 $(\alpha, t)$-Cluster Framework .................. 26
2.4 Clustering Algorithms in VANETs .......................... 27
  2.4.1 Modified DMAC Algorithm .......................... 27
2.5 Summaries ..................................................... 29

3 Performance Metric for Vehicular Networks .............. 31
  3.1 Clustering Problem .......................................... 31
  3.2 Clustering Performance Metric .......................... 32
    3.2.1 Link Expiration Time (LET) ......................... 33
    3.2.2 Criticality ........................................... 34
    3.2.3 Metric Analysis .................................... 38
  3.3 Centralized Clustering Algorithm ....................... 42
  3.4 Local Clustering Parameter ............................ 48
    3.4.1 Local Network ...................................... 48
    3.4.2 Local Clustering Metric .......................... 50

4 Clustering Algorithm Using Network Criticality... ....... 53
  4.1 Distributed Clustering Algorithm ..................... 53
    4.1.1 States .............................................. 54
    4.1.2 Messages Types .................................... 55
    4.1.3 Algorithm Implementation ......................... 57
  4.2 Performance Evaluation ................................ 63
    4.2.1 Simulation Setup .................................. 64
    4.2.2 Design Parameters ................................ 64
    4.2.3 Low-Velocity Scenario ......................... 71
    4.2.4 High-Velocity Scenario ......................... 74

5 Conclusion ...................................................... 80

Bibliography ...................................................... 81
# List of Tables

1.1 Examples of IEEE 802.11 Standard ........................................ 4
1.2 Two phases for maintaining a cluster structure .......................... 8

2.1 Agglomerative Clustering Algorithms ..................................... 16

4.1 Symbols .................................................................................. 58
4.2 Algorithm: TempCH State for node $u$ ................................. 59
4.3 Algorithm: Idle State for Node $u$ ........................................ 60
4.4 Algorithm: SelectClusterhead .................................................. 60
4.5 Algorithm: Clusterhead State for node $u$ .............................. 61
4.6 Algorithm: Member State for node $u$ .................................... 62
List of Figures

1.1 a vehicular network architecture ............................................. 5
1.2 Cluster structure of a vehicular network ................................. 7
1.3 a vehicular network architecture ............................................. 10

2.1 The Topology of A Small Network .......................................... 17
2.2 Dendrogram for agglomerative clustering ................................. 18
2.3 K-Means Clustering Algorithm ............................................. 20
2.4 K-Medoids Clustering .......................................................... 22

3.1 The impact of $\delta$ on $S[2]$ .................................................. 34
3.2 The position of nodes ............................................................. 42
3.3 $\lambda_m^{-\frac{1}{2}} \phi_m$ and $\phi_m$, $m = 2, 3$ .............................. 43
3.4 Two-dimension for $\lambda_m^{-\frac{1}{2}} \phi_m$ and $\phi_m$, $m = 2, 3$ ......... 44
3.5 Three-dimension for $\lambda_m^{-\frac{1}{2}} \phi_m$ and $\phi_m$, $m = 2, 3, 4$ .... 44
3.6 The k-means clustering algorithm: physical distance, '*' is the centroid of cluster ......................................................... 46
3.7 The simple k-medoids clustering algorithm: physical distance, '*' is the centroid of cluster ..................................................... 47
3.8 The simple k-medoids clustering algorithm: $\tau_{ij}$, '*' is the centroid of cluster ................................................................. 47
3.9 A snapshot of a vehicular network ........................................... 48
3.10 $T$-hop local network of node 7 ............................................. 49
3.11 Example: point-to-point criticality in $T$-hop local network (node 7) ... 51

4.1 State Transition ....................................................... 55
4.2 The clusterhead lifetime for $TTL = 2$, $\tau_{\text{threshold}}^{\text{link}} = 0.1$ ................. 65
4.3 The average lifetime of clusterheads and members .......................... 66
4.4 The state change times with $\delta = 100$ and $TTL = 2$ ..................... 67
4.5 The average lifetime with $\delta = 100$ and $TTL = 2$ ...................... 68
4.6 The average number of clusters, $\delta = 100$ and $\tau_{\text{threshold}}^{\text{link}} = 2$ ........ 69
4.7 The average clusterhead changes per node, $\delta = 100$ and $\tau_{\text{threshold}}^{\text{link}} = 2$ .. 70
4.8 The average cluster state changes per node, $\delta = 100$ and $\tau_{\text{threshold}}^{\text{link}} = 2$ .. 70
4.9 The average number of clusters with $\delta = 100$, $\tau_{\text{threshold}}^{\text{link}} = 2$ ................. 71
4.10 The average cluster size with $\delta = 100$, $\tau_{\text{threshold}}^{\text{link}} = 2$ ...................... 72
4.11 The clusterhead change times per node with $\delta = 100$, $\tau_{\text{threshold}}^{\text{link}} = 2$ ..... 73
4.12 The member change times per node with $\delta = 100$, $\tau_{\text{threshold}}^{\text{link}} = 2$ ........ 74
4.13 The average lifetime of clusters with $\delta = 100$, $\tau_{\text{threshold}}^{\text{link}} = 2$ ............... 75
4.14 The average number of clusters with $\delta = 100$, $\tau_{\text{threshold}}^{\text{link}} = 2$ ............... 76
4.15 The average cluster size with $\delta = 100$, $\tau_{\text{threshold}}^{\text{link}} = 2$ ....................... 76
4.16 The clusterhead change times per node with $\delta = 100$, $\tau_{\text{threshold}}^{\text{link}} = 2$ ..... 77
4.17 The member change times per node with $\delta = 100$, $\tau_{\text{threshold}}^{\text{link}} = 2$ .......... 77
4.18 The average lifetime of clusters with $\delta = 100$, $\tau_{\text{threshold}}^{\text{link}} = 2$ ............... 78
Chapter 1

Introduction

1.1 Emergence of Connected Vehicles

Vehicle manufacturers are interested in implementing vehicular communications to provide both security and comfort applications, such as collision avoidance, road condition awareness and Internet access. The usage scenarios cover a wide range. For example, if one vehicle detects high volume of traffic on a road, it could inform other vehicles in the same area, so they could select another road. Passengers could check emails, download files and even play online games through vehicular communication network.

Vehicular communication network facilitates user mobility which are constrained by many factors. Mobile vehicles have limited transmission range and limited storage capacity; meanwhile wireless links are vulnerable to interference. Except these common problems in wireless networks, one distinguishing feature of a vehicular communication network is the high mobility of vehicles. Equipped with communication device, vehicles can be interconnected while on roads. Here, we summarize the characteristics of vehicular communication networks:

- Mobile - Most vehicles have high mobility and nearby vehicles have low relative velocities. However, vehicles have to running along the trajectory defined by roads.
Thus a vehicular communication network is a dynamic network whose topology is to some extent controlled by the respective traffic road topology.

- **V2V Communication** - V2V (Vehicle-to-Vehicle) communication utilizes WLAN (Wireless Local Area Network). A VANET (Vehicular Ad Hoc Network) is formed as a specific class of MANET (Mobile Ad Hoc Network) based on V2V communication. Through V2V communication, vehicles are able to communicate directly.

- **V2I Communication** - V2I (Vehicle-to-infrastructure) communication prefers WWAN (Wireless Wide Area Network). In a vehicular communication network, assisted infrastructures exist except mobile vehicles. For example, communication among vehicles using 3G (third Generation) wireless network is under the control of BSs (Base Stations). Infrastructure can be either fixed or mobile.

- **Power** - Unlike small mobile devices, vehicles have a powerful battery. Thus, power is not a huge problem here compared to other wireless networks, such as sensor networks.

Considering these characteristics, it is difficult to use traditional network architecture to provide a reliable vehicular communication network. Thus, a novel network architecture and corresponding management system is required.

To explore the suitable architecture of vehicular communication network, let’s first review the traditional communication networks. As we discussed above, a vehicular communication network comprises V2V communication and V2I communication. V2V communication lets nodes communicate directly and requires no infrastructure. However, the self-organized network is limited in a local level. A large-scale employment leads a frequent overhead exchanges or even congestion in VANET. At the same time, V2I communication provides the stable and wide connection with an expensive cost. The two communication methods represent to different network architecture: WLAN and WWAN.
WLAN and WWAN are two different wireless network architectures. Their names indicate the difference in the transmission range. One widespread deployment WLAN technology is IEEE-based 802.11 standards. IEEE 802.11 based WLANs have a small coverage of only hundred meters while they are able to provide high data rates within the local area.

On the other hand, WWAN is a network utilizing powerful infrastructure to achieve a large transmission range. Usually, the infrastructure is fixed. A typical WWAN is cellular network, such as 3G networks. The radius of a WWAN cell can be up to 20 miles with a powerful BS. But the large coverage comes at the price of high cost and relatively low data rates compared to WLANs.

Both WWAN and WLAN architectures are widely employed, however, no unified view has emerged on which is better for vehicular communication networks. Many researches have been focused on improving the performance of the two network architectures separately. Nowadays, there is a tendency to combine two networks together, i.e. build a hybrid network architecture. In this thesis, we will focus on the hybrid network architecture.

1.2 Connected Vehicular Network Architecture

In the hybrid network architecture, we assume that WLAN is based on IEEE 802.11 standard and WWAN is implemented by 3G. Wireless network is constituted by different layers, and each layer has its own responsibilities. WWAN has already had some acknowledged standards. The most popular one is 3G network. In our network architecture, 3G wireless network works as WWAN part. Each vehicle communicates with its BS. As shown in Figure 1.1, BS accesses to Internet through wired network. Usually, 3G network provides relative low throughput (up to 2.4Mbps) with large cell coverage (up to 20 miles).
### Table 1.1: Examples of IEEE 802.11 Standard

<table>
<thead>
<tr>
<th>Standard</th>
<th>Maximum Data Rate</th>
<th>Frequency</th>
<th>Modulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE802.11a</td>
<td>54Mbps</td>
<td>5/3.7GHz</td>
<td>OFDM $^a$</td>
</tr>
<tr>
<td>IEEE802.11b</td>
<td>11Mbps</td>
<td>2.4GHz</td>
<td>DSSS $^b$</td>
</tr>
<tr>
<td>IEEE802.11g</td>
<td>54Mbps</td>
<td>2.4GHz</td>
<td>OFDM, DSSS</td>
</tr>
<tr>
<td>IEEE802.11n</td>
<td>248Mbps</td>
<td>2.4/5GHz</td>
<td>OFDM, MIMO $^c$</td>
</tr>
<tr>
<td>IEEE802.11p</td>
<td>27Mbps</td>
<td>5.9GHz</td>
<td>OFDM</td>
</tr>
</tbody>
</table>

$^a$ OFDM: Orthogonal Frequency Division Modulation  
$^b$ DSSS: Direct-Sequence Spread Spectrum  
$^c$ MIMO: Multiple Input Multiple Output

WLAN, unfortunately, does not have such a unified standard. Many different technologies have been used for the mobile wireless network. At the PHY/MAC layer, among a large number of proposed techniques, what we are interested in is IEEE 802.11 Standards. IEEE 802.11 is a set of standards designed for WLAN. It enables wireless connectivity between electronic devices in a limited range. IEEE 802.11 may not be the most advanced technology to implement such wireless networks; however, it is a mature technique, which is convenient and inexpensive. At the MAC layer, IEEE 802.11 provides a shared access to a wireless channel by Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) as its default mode. Table 1.1 explains more details about IEEE 802.11 Standards. Note that IEEE 802.11p is proposed to support for WLANs in a vehicular network, which is widely employed in current VANETs.

Figure 1.1 represents a vehicular communication network architecture. The involved entities of the network architecture are listed in the following.

- **Mobile Node (MN):** Mobile Nodes are vehicles running in the network. Vehicular communication network provides service to these mobile nodes. It is the reason why vehicular networks exist. Communication device equipped in a vehicle provides an IEEE 802.11 interface and a 3G interface. These interfaces allow vehicles access
Figure 1.1: a vehicular network architecture

...to both WWAN and WLAN. Usually, mobile nodes move at high speeds along predefined roads in vehicular network. Note that we assume we assume all mobile nodes equipped with both interfaces, and the penetration problem is ignored.

- Base Station (BS): BS is an important infrastructure in 3G cellular network. It supports large storage, powerful computation, and wide area access. BS provides a stable but limited bandwidth to vehicles within their large coverage. Note that BS only has 3G interface to vehicles.

- Server: Servers are functional entities which provide service to their clients. In our case, MNs are the targeted clients. Generally speaking, servers are nodes which are powerful enough to provide specific service to specific clients. In our case, servers are located in WWAN. Typical service provided by WWAN includes Email, web browsing and so on.
1.3 Clustering of Vehicular Network

The purpose of this thesis is to design a clustering algorithm for vehicular networks. The objective of a clustering algorithm is to partition a network into some subnetworks each of which with some similar attributes according to an appropriate metric. In this section, we’d like to introduce the concept of the cluster structure of vehicular networks, the clustering algorithms and why we are interested in clustering vehicular networks.

**The Cluster Structure**

The network architecture is shown in Figure 1.1. There are two methods to implement the communication between vehicular networks and the backbone Internet. The first one is that MNs utilize WWAN directly to communicate with the Internet. And the second one employs both WWAN and WLAN in a cluster structure. In this method, MNs forms different clusters. The cluster is the basic unit to implement V2I communication while V2V communication maintains the intra-cluster communication.

The above two methods indicate two different ways to organize the vehicular network. The first method views the vehicular network as a flat topology. In the flat topology, each node is equal. WWAN and WLAN work in parallel. No matter whether a WLAN exists or not, the communication can be implemented. The second method provides a cluster structure to the network. The cluster structure generates a hierarchical vehicular network. MNs in different tiers take different responsibilities.

The concept of the cluster in a mobile wireless network is a subset of nodes which are physically near to each other (distance) and have similar mobility patterns (velocity). The distance and the velocity are features of nodes.

In each cluster, two basic roles are necessary: clusterhead and member. A simple example of cluster structure is shown in Figure 1.2. The overlay network let MNs utilize WLAN or WWAN alternatively. A cluster has at least one clusterhead to manage other members in the cluster. To simplify the problem, only the single-clusterhead case is considered in the overlay network. The clusterhead collects and forwards all other
members’ Internet access requests. Therefore, only the clusterhead’s WWAN interface is activated in the cluster while every node keeps WLAN interface activated. More roles are acceptable depending on the operation of clustering algorithms.

![Cluster structure of a vehicular network](image)

**Figure 1.2: Cluster structure of a vehicular network**

*The Clustering Algorithm*

In vehicular networks, the cluster structure is implemented by the clustering algorithm. The clustering algorithm supports an appropriate topology for the distributed mobile nodes and decides the effect of routing algorithms. Distributed clustering algorithms provide a self-organized hierarchy to distributed nodes.

The vehicular network is a highly dynamic mobile wireless network; therefore, it requires a robust clustering algorithm. Our view of the concept of robustness is motivated by the concept of network criticality, which is discussed in [31] [30]. Network criticality is a quantification to measure the resistance ability of a network to the unexpected changes. Generally speaking, a robust clustering algorithm indicates that the resulting clustered topology considers not only the instant topology, but also the future changes. Hence, the
clustering algorithm can accurately describe the optimal cluster structure.

To implement a robust clustering algorithm, two problems need to be solved: a clustering metric has to be selected and clustering methods need to be designed.

A clustering metric should provide the quantitative definition of closeness which is fundamental to the clustering problem. Furthermore, clustering metric should reflect the characteristics of its target networks. In VANETs, the network topology is frequently changing. If a metric can resist against the unnecessary changes while maintaining connectivity, it is robust. A suitable metric can reflect features of nodes and improve performance of the clustering algorithm.

Clustering methods are the approaches to assign nodes into clusters. Strictly speaking, in vehicular networks, the problem is not only how to form a cluster structure, but also how to maintain the cluster structure in a dynamic network. Table 1.2 lists two phases needed to form a cluster structure.

<table>
<thead>
<tr>
<th>Table 1.2: Two phases for maintaining a cluster structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster Formation</td>
</tr>
<tr>
<td>Cluster Maintenance</td>
</tr>
</tbody>
</table>

The cluster formation and maintenance work together in the whole operation period. The formation phase provides a cluster based on the current and predictable network topology, while the maintenance phase reduces the impact caused by the unpredictable network changes.

Based on the above analysis, we conclude that the clustering method should be distributed and rapid. Implementing the algorithm distributive is decided by the nature of WLANs. In each cluster, there should be a clusterhead to manage the inter-cluster and intra-cluster communication. And each node should be mapped into one and only one cluster.
Chapter 1. Introduction

The Advantages of Clustering

Finally, we’d like to discuss one question: why do we prefer the cluster structure to the flat network?

At first glance, the cluster structure looks strange and unnecessary since the flat topology is intuitive and simple. However, considering that there might be hundreds or even thousands MNs running in the coverage area of a base station, the usage of the cluster structure is beneficial.

Taking Figure 1.1 as an example, the network contains 9 vehicles in total. It is covered by a WWAN and two WLANs. Assume that 9 MNs send requests to servers in the backbone network and the flat topology is employed. In the worst case, there are 9 sessions operating concurrently. Unfortunately, a base station provides coverage across miles, which means that thousands MNs might be covered. To avoid network congestion, service providers have to increase the quantities of infrastructure or shrink the coverage range. As a result, the high cost of WWAN keeps increasing. Definitely, the flat topology is not a good choice in a large-scale mobile wireless network.

If the cluster structure is employed, two clusterheads are introduced in our example as shown in Figure 1.2. Even the 9 nodes generate 9 sessions at the same time; all members send theirs requests to their clusterheads. And the clusterheads forward these messages from the backbone Internet to its members through the WLAN channels. There are just 2 WWAN channels required in the cluster structure.

It is not difficult to find that clustering is an efficient method to organize large scale wireless networks in a hierarchical way. Especially when we combine WWANs and WLANs together, the two different network mechanisms naturally form a two-layer network. The cellular network provides infrastructures to connect to servers in the fixed networks. The servers can manage networks in a centralized way which is efficient but expensive. WLAN is in charge of maintaining connectivity in a local area. As described above, WWAN maintains more stable connections while WLAN provides higher data
rate. Previous researches prove that opportunistic use of WLAN is employed to improve the throughput of WWAN and vice versa.

1.4 Network Model of This Thesis

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{vehicular_network_architecture.png}
\caption{a vehicular network architecture}
\end{figure}

Considering a vehicular network contains \( n \) nodes. If we take a snapshot at an arbitrary time, the network topology can be described as a graph \( G = (V, E) \), where \( V = (v_1, v_2, ..., v_n) \) is the set of nodes and \( E \) is the set of links between nodes. Figure 1.3 is the graph for our above example in Figure 1.2. We assume the following properties for graph \( G \):

- \( G \) is an undirected graph. For each link, which has a non-negative weight \( w_{uv} \), \( w_{uv} = w_{vu} \) is always true. \( W = [w_{ij}] \) is the weight matrix of \( G \).

- For each edge, \( e(u, v) \), a weight \( w_{uv} \) is defined to represent the weight of the edge between node \( u \) and node \( v \).

- Each node is equipped with GPS, so position and velocity of each node are knowable.
• Each node has the same transmission power in WLAN part, i.e., the transmission power level is fixed for each node.

• Each node, no matter mobile or fixed, is treated equally in the proposed clustering algorithm.

• The WWAN capacity provided by one node is enough to handle a whole cluster’s requests, i.e., in our clustering algorithm, the metrics are based on WLAN.

1.5 Contribution of This Thesis

The thesis sheds light on one problem: how does a vehicular communication network build a robust cluster structure efficiently? Given that clustering is a widely employed method to form an overlay network, the contribution of the thesis includes:

First, we define point-to-point criticality and node criticality, i.e. $\tau_{ij}$ and $\tau_i$ based on the definition of network criticality, which entitle $\tau_{ij}$ to execute network partition. Network criticality is a global measure to evaluate the robustness of networks to environmental changes. The new metric, $\tau_{ij}$, evaluates the robustness of the connectivity between a pair of nodes $(i, j)$. It considers all the paths between a pair of nodes, and it reflects the robustness between nodes and resists the change of network topology. $\tau_i$ is derived from $\tau_{ij}$ and reflects the robustness of a node. As we mentioned above, we want to build a robust hierarchical network, the properties of criticality make them suitable to represent our target. It is the first time that $\tau_{ij}$ and $\tau_i$ are used for clustering networks.

Second, we make the calculation of $\tau_{ij}$ and $\tau_i$ decentralized. The novel metrics are defined as local point-to-point criticality and local node criticality. They are denoted by $\tau_{ij}^{local}$ and $\tau_i^{local}$. They utilize the local network topology to calculate their value. Their values can be rapidly adjusted to the network changes and the required storage and calculation time is significantly reduced.
Finally, we explore the clustering algorithms in both centralized and distributed method. We prove that the centralized clustering algorithm can be explained by the classic k-means clustering algorithm. Being different from the centralized algorithm, we discuss the distributed algorithm in two separate phases: formation and maintenance.

In the following chapters, we will discuss these problems. In Chapter 2, we first review the generalized clustering algorithms based on graph theory. And then we compare the advantages and disadvantages of distributed clustering algorithms operating in MANET and VANET. Chapter 3 introduces the properties of $\tau_{ij}$ and $\tau_i$. Performance of these metrics is analyzed in a centralized k-medoid clustering. A distributed version of $\tau_{ij}$ and $\tau_u$ are also discussed. Chapter 4 propose a distributed clustering algorithm, (CCA) Criticality-based Clustering Algorithm. The framework of clustering algorithm is discussed in this chapter too.
Chapter 2

Overview and Comparison of Vehicular Clustering Algorithms

This chapter describes the clustering algorithms from three aspects. The first aspect explains the generalized community detection algorithms in graph theory. The second one introduces some widely employed clustering algorithms in MANETs. At the end of this chapter, we introduce some current algorithms for clustering in VANETs.

2.1 Preliminaries

2.1.1 Distance

Distance evaluates the similarity between nodes. In a specific metric space, each node has its own coordinate vector. Distance based on these vectors measures how far away one node is from another node in the corresponding space. The closer two nodes are, the more similar two nodes are. In clustering algorithms, the selection of distance impacts the quality of clustering. There are different definitions of distances that are applicable. In Chapter 3, we will discuss the distance measure problem for our algorithm. Therefore, we just introduce the generalized concept of distance, Euclidean distance, in this chapter.
The definition of Euclidean distance is:

\[
\begin{align*}
    d(x_i, x_j) &= \left( \sum_{k=1}^{m} |x_{i,k} - x_{j,k}|^2 \right)^{1/2} = \|x_i - x_j\| \\
    & \quad \text{where } x_i \text{ and } x_j \text{ are two nodes with } x_i = (x_{i,1}, x_{i,2}, \ldots x_{i,m}) \text{ and } x_j = (x_{j,1}, x_{j,2}, \ldots x_{j,m}) \text{ in a Euclidean space } \mathbb{R}^m.
\end{align*}
\]

Note that weight and distance are two different concepts. Weight is a metric employed to evaluate the quality of a link. We can define that weight is a function of distance or not. Distance is a parameter calculated by the definition. No matter correlated to weight or not, distance exists objectively by its definition.

### 2.1.2 The Unnormalizzed Laplacian Matrix

In a graph \( G = (V, E) \), the weight matrix is \( W = \{w_{ij}\}_{i,j=1}^{n} \). Let the diagonal matrix \( M \) have:

\[
    m_{ij} = \begin{cases} 
    \sum_{k=1}^{n} w_{ik} & , i = j \\
    0 & , \text{otherwise}
    \end{cases} \quad (2.2)
\]

\( M \) is the degree matrix for weight \( W \). We have the unnormalized graph laplacian matrix

\[
    L = M - W \quad (2.3)
\]

The unnormalized Laplacian matrix incorporates many properties of graphs. For example, Fiedler has proved that the second smallest eigenvalues of the Laplacian matrix reflects the algebraic connectivity of a graph, and thus the corresponding eigenvector has been named the Fiedler vector. \cite{22}. There are many literatures concluding these properties. \cite{10}\cite{21}\cite{22} Among these properties, the most important property for spectral clustering is shown in the following proposition.
Proposition 1 For every vector $\mathbf{x} = (x_1, x_2, ... x_n)$, we have:[17]

$$x' L x = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} (x_i - x_j)^2$$  \hspace{1cm} (2.4)

Proof: By the definition of $L$,

$$x' L x = x' (M - W) x$$

$$= \sum_{i=1}^{n} x_i^2 m_{ii} - \sum_{j=1}^{n} x_i x_j w_{ij}$$

$$= \frac{1}{2} \left( \sum_{j=1}^{n} x_j^2 w_{ij} - \sum_{j=1}^{n} x_i x_j w_{ij} + \sum_{j=1}^{n} \sum_{i=1}^{n} x_i^2 w_{ij} \right)$$

$$= \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{n} w_{ij} (x_i - x_j)^2.$$ \hspace{1cm} $\square$

In addition to the unnormalized Laplacian matrix, there are two more normalized Laplacian matrices. They are also used for spectral clustering. The differences among these matrices are discussed in [18]. Here, we just discuss the unnormalized matrix, which is closely related to our work. In the following part, we will call the unnormalized Laplacian matrix the Laplacian matrix for simplicity.

### 2.2 Clustering Algorithms Based on Weighted Graph

In a graph, a cluster is named community, and so vehicular clustering can be classified into a community detection problem, which focuses on aggregating similar nodes together with the given features of nodes. For simplification, we do not distinguish between "cluster" and "community" in the subsequent description.

Generally speaking, there are two basic types of clustering methods: hierarchical clustering algorithms and partitional clustering algorithms. Hierarchical algorithms generate cluster structure on the basis of existing cluster structure. These algorithms are implemented by a sequence of partition while the partitional algorithms divide the nodes into...
Table 2.1: Agglomerative Clustering Algorithms

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial: $P_1 = {{v_i}, i = 1, 2, \ldots n}$, $d(C_i, C_j) = d(v_i, v_j)$</td>
<td></td>
</tr>
</tbody>
</table>

For $k = 2$ to $n$

- Find $C_1$ and $C_2$: $d(C_1, C_2) = \min_{C_i, C_j \in P_{k-1}} d(C_i, C_j)$
- $C_3 = C_1 \cup C_2$
- $P_k = (P_{k-1} \setminus C_1, C_2) \cup C_3$
- Update $d(C_i, C_j)$
- $k = k + 1$

end

Clusters through a single partition.

### 2.2.1 Hierarchical Clustering

Hierarchical clustering is a useful method to analyze the overall structure of networks. It depicts the relationship of nodes in a graph by a hierarchical tree named dendrogram. Dendrogram reveals the multilevel structure of a graph. Each level of dendrogram represents a set of clusters, and each cluster defined by top-level is constituted by clusters from lower-level. Each cluster at the lowest level of dendrogram contains only one node. Figure 2.2 is a simple dendrogram generated by hierarchical clustering. [9][13]

Hierarchical clustering can be subdivided into two approaches, agglomerative algorithms and divisive algorithms. Agglomerative algorithms iteratively merge small clusters into bigger ones, which operate "from the bottom up". And divisive algorithms iteratively split big clusters into small ones which go "from the top down". Since agglomerative algorithms are more commonly used in social networks, computer networks and other networks, we will only discuss agglomerative algorithms. A complete introduction and comparison about the two approaches can be found in [23].
Agglomerative algorithms construct dendrogram in a greedy manner. Table 2.1 summarizes a procedure of agglomerative algorithms. At the initial step, it assumes each node $v_i$ is a cluster, and these clusters form the first partitions, $P_1$, of the graph. The distance between two clusters $C_i$ and $C_j$ is represented as $d(C_i, C_j)$. In $P_1$, $d(C_i, C_j)$ is the same as the distance between nodes, $d(v_i, v_j)$. At each step $k$, agglomerative clustering selects the smallest distance, $d(C_1, C_2)$ based on the previous partition $P_{k-1}$, and then merges $C_1$ and $C_2$ into a new cluster $C_3$, which results a new partition $P_k$ at step $k$. Finally, at step $n$, $P_n = \{V\}$.

![Figure 2.1: The topology of a small network: the integer on each link denotes the distance from one end to another.](image)

Table 2.1 indicates that two kinds of distances are required. The first one is $d(v_i, v_j)$, which reflects the dissimilarity between two nodes $v_i$ and $v_j$ as Equation(2.1). Another kind of distance is the distance between clusters, $d(C_i, C_j)$. Agglomerative algorithms propose several techniques to calculate it. In single linkage clustering, $d(C_i, C_j)$ is equal to the minimum $d(v_i, v_j)$, with $v_i$ in $C_i$ and $v_j$ in $C_j$. In complete linkage clustering, the maximum $d(v_i, v_j)$ replace the minimum value to represent distance between $C_i$ and $C_j$. In average linkage clustering, $d(C_i, C_j)$ is the average of $d(v_i, v_j)$. Figure 2.1 is the topology of a simple and small network and Figure 2.2 is the corresponding dendrogram.
One key advantage of hierarchical clustering is that it doesn’t require the number of clusters in a graph before clustering. However, an extra criterion needs to be introduced if we want to obtain an optimal partition among the sequences of partitions. For example, Newman et.al. have proposed modularity to select a better representation of the community structure. [23] Another problem is that hierarchical clustering does not scale well. The greedy manner of clustering leads to the problem. The computational complexity is $O(n^2)$ for single linkage clustering and $O(n^2 \log n)$ for complete and average linkage clustering.

### 2.2.2 K-Means Clustering

K-Means clustering is a classical partitional clustering algorithm. One characteristic of partitional clustering is that the number of clusters $k$ should be specified before clustering process. With the given $k$, k-means clustering is able to assign nodes to corresponding clusters in a single partition. [9]
The goal of k-means clustering algorithm is to optimize a given criterion function. Square error function is the most intuitive and popular criterion function in k-means clustering. Equation (2.5) is the definition of square error function.

\[
J = \sum_{i=1}^{k} \sum_{j=1}^{n} \| x_j^{(i)} - c_i \|^2 \tag{2.5}
\]

where \( x_j^{(i)} \) denotes the \( j \)-th node belonging to the \( i \)-th cluster \( C_i \), and \( c_i \) is the center of cluster \( C_i \). Usually, the center is defined as the centroid of the cluster in k-means clustering algorithm, as shown in Equation (2.6).

\[
c_i = \frac{1}{|C_i|} \sum_{x_j^{(i)} \in C_i} x_j^{(i)} \tag{2.6}
\]

where \(|C_i|\) is the number of node in cluster \( C_i \).

No matter what kind of distance measure k-means clustering algorithms employs, it means that there is a corresponding metric space embedding all nodes. The most intuitive example is physical distance. Based on the definition, the square error function (see Equation (2.5)) is the summation of intra-cluster distance, i.e. the intra-cluster dissimilarity, between each node and its cluster center. Therefore, the objective function of k-means clustering is to minimize Equation (2.5). Note that the distance represents the dissimilarity between nodes. If we use the similarity metric instead of the dissimilarity metric, the objective function should be changed respectively. [13]

**K-Means Clustering Algorithm**

1. Select initial \( k \) cluster centers;

2. Assign each node to the nearest cluster center, i.e., each node joins a cluster whose center has the smallest distance to it;

3. Update the cluster centers based on Equation (2.6);
4. Check all new centers, if they all keep the same as previous centers, k-means clustering meet convergence condition and stop; otherwise, go step 2 and repeat.

K-means clustering is popular due to its easy implementation and fast convergence. The time complexity is only $O(n)$, where $n$ is the total number of nodes. Figure 2.3 is a simple example of k-means clustering based on Figure 2.1. Obviously, k-means clustering achieves the similar result as hierarchical clustering while it requires much less time. In a large scale network, it is a distinguishing advantage of k-means clustering.

At the same time, there are disadvantages limiting the usage of k-means clustering in dynamic, large-scale networks. First of all, k-means clustering is very sensitive to the initial set of centers. Different initial centers lead to different clustering results, and so different numbers of iterations are required to achieve convergence. In worst case, k-means clustering cannot converge to a global minimum of Equation (2.5). Literatures have discussed how to select a proper initial partition for k-means clustering, such as [1][6][12][26]. Random selection is the most common method. It selects the initial cluster centers at random. This way is easy to implement, but the result is variable with variable initial centers. It cannot guarantee the efficiency of algorithm.
Secondly, how to decide the number of $k$ is also a problem. Especially in a dynamic network, the number of clusters is changing due to the mobility of node. These problems will be resolved in the algorithm we propose.

### 2.2.3 K-Medoids Clustering Algorithm

K-medoids clustering is also a partitional clustering algorithm. Both k-means clustering and k-medoids clustering partition $n$ nodes with a preassigned number, $k$.

To implement k-medoids clustering, many algorithms have been proposed, such as PAM (Partitioning Around Medoids), CLARA (Clustering LARge Applications) [14], and CLARANS (Clustering Large Applications based upon RANdomized Search) [24]. However, the time complexities of these algorithms make them inefficient for large data.

Considering that fast convergence is required for mobile wireless network, we’d like to introduce a k-medoids clustering algorithm similar to k-means clustering. To avoid confusion, we call the specific algorithm simple-k-medoids. The medoid of each cluster is defined in Equation (2.7).

$$m_i = \arg \min_{x_j} \sum_{x_k \in C_i} \| x_k - x_j \|^2$$ (2.7)

where $m_i$ denotes the medoid of cluster $C_i$.

In the case for simple-k-medoids, 4 basic steps are required:

1. Select initial $k$ cluster medoids $\{m_i\}, i = 1, 2, \ldots k$;

2. For $j = 1, 2, \ldots n$, assign node $x_j$ to the nearest medoid $m_i$ such that

$$m_i = \arg \min_{l=1,2,\ldots,k} \| x_j - m_l \|^2$$ (2.8)

3. Update the cluster medoids based on Equation (2.7);
4. If achieve convergence condition, stop; otherwise, repeat the last two steps.

Simple-k-medoids is nearly the same as k-means clustering algorithm. It is not hard to conclude that the objective of simple-k-medoids is also to minimize Equation (2.5) just replacing \( c_i \) by \( m_i \). It seems that there may be no reason to study simple-k-medoids clustering while we already have classical k-means clustering.

However, k-medoids clustering is more robust to outliers in comparison with k-means clustering. In k-means, the center is defined as the centroid of the cluster shown in Equation (2.6). It means that the center is not necessarily a real node in the cluster. K-medoids clustering declares that the center of a cluster should be a representative node called medoid. Or we say the cluster center should be an existing node and can be viewed as the median of all nodes in a same cluster. Figure 2.4 shows the k-medoid clustering result on the same topology as Figure 2.3. Note that the location of cluster centers are changed, even the two clustering algorithms provide the same result. Accordingly, the selection of a medoid is based on the most centrally local nodes of the cluster. That’s why the medoid is less sensitive to outliers.

Research related to simple-k-medoids focus on two aspects. The first one is the
performance of the algorithm. Park et. al. have proposed a k-medoid algorithm with the same definition of the medoid. Based on some real and artificial data sets, they point out that simple-k-medoids keeps the robustness to outliers against k-means clustering. And the time complexity is significantly reduced against the complexity of PAM. [25]

The other aspect emphasizes the application of k-medoids clustering. Yen et. al. have proposed an ECT k-means clustering in [33]. The algorithm, which uses a prototype to replace the centroid in k-means clustering, is essentially a k-medoids clustering algorithm. They not only discuss how the different distance measures impact the results of clustering algorithms, but also discuss how the algorithms are used for digital characters clustering.

2.2.4 Spectral Clustering

Spectral clustering is novel and popular in image segmentation. Before introducing spectral clustering, we have already introduced the Laplacian matrix which is an important tool for spectral clustering.

**Spectral Clustering Algorithm**

In spectral clustering, the most common criterion method is the radio cut method. For a set of clusters $C_1, C_2, ..., C_k$ satisfying $C_1 \cap ... \cap C_k = \phi$ and $C_1 \cup ... \cup C_k = V$, the radio cut method is

$$\min_{i=1}^{k} \frac{\sum_{m \in C_i, n \in C_i \setminus C_i} w_{mn}}{|C_i|} \quad (2.9)$$

Luxburg et. al. [18] and Magnus [19] have proved that the optimized solution for Equation (2.9) is the first $k$ eigenvectors corresponding to the $k$ smallest eigenvalues of $L$. However, the $k$ eigenvectors form a real valued $n \times k$ matrix, we have to convert the matrix to a discrete partition. Usually k-means clustering is used for the re-partition.

The spectral clustering algorithms is listed as follows: [18]

- Calculate the Laplacian matrix $L$ based on Equation(2.3);
• Compute the first \( k \) eigenvectors \( v_1, ..., v_k \) of \( L \), \( k \) is a predefined value;

• Let \( v_1, ..., v_k \) be the column of \( V \), which forms \( V \in \mathbb{R}^{n \times k} \);

• View each row of \( V \) as a vector for a node, and cluster the \( n \) nodes into \( k \) clusters \((C_1, C_2, ..., C_k)\) through k-means clustering.

Spectral clustering is derived from graph theory. It converts the original nodes into a metric space whose coordinates are the eigenvectors of \( L \). Combining with k-means clustering in the final step, spectral clustering can be explained as k-means clustering in a specific metric space. The transformation of the metric space emphasizes the features of clusters. Some complex community structure which cannot be detected by k-means on Euclidean space can be clustered by spectral clustering. [9]

However, spectral clustering contains a spectral decomposition of the Laplacian matrix. The calculation complexity is \( O(n^3) \). The main disadvantage of spectral clustering limits the employment in large scale networks.

### 2.3 Clustering Algorithms in MANETs

#### 2.3.1 Lowest-ID and Highest-Degree Clustering

The Lowest-ID Clustering is the simplest clustering algorithm. Proposed by Baker and Ephremides, each node owns a unique ID and the node with minimum ID is selected as a clusterhead.

The Highest-ID Clustering is a connectivity-based clustering algorithm. Proposed by Gerla and Parekh, a node with highest degree, i.e. the maximum number of neighbors, is selected as a clusterhead. [7]

The two clustering algorithms are often compared with other algorithms because they employ two extreme metrics. While the Lowest-ID clustering is more stable at the expense of the small ID bearing more tasks, the Highest-Degree clustering sacrifices
throughput to get more stable clusters, since the throughput decreases while the size of cluster is large.

2.3.2 MOBIC

MOBIC is a distributed clustering algorithm which takes the mobility of nodes into account. [5] It doesn’t require any additional equipment, such as GPS. Defining an aggregate local mobility metric, it tends to select nodes with low relative mobility as clusterheads.

In MOBIC, the signal power received at node $Y$ from $X$ is represented by $RxPr_{X \rightarrow Y}$. This value indicates the distance between the two nodes. The relative mobility metric of a node $Y$ is derived from the ratio of two successive packet transmissions from neighbors:

$$M_{rel}^{rel}(X) = 10 \log_{10} \frac{RxPr_{X \rightarrow Y}^{new}}{RxPr_{X \rightarrow Y}^{old}}$$  \hspace{1cm} (2.10)

If $RxPr_{X \rightarrow Y}^{new} > RxPr_{X \rightarrow Y}^{old}$, which result a positive $M_{rel}^{rel}(X)$, it implies that $X$ is moving closer to $Y$. And if $RxPr_{X \rightarrow Y}^{new} < RxPr_{X \rightarrow Y}^{old}$, $M_{rel}^{rel}(X)$ is negative which implies the two nodes are moving away from each other.

Taking the whole set of $M_{rel}^{rel}(X_i)$ as samples, where $X_i$ is a neighbor of $Y$, the aggregate mobility metric is defined as the variance of these samples, with respect to 0.

$$M_Y = \text{var}_0\{M_{rel}^{rel}(X_j)\}_{j=1}^{m}$$  \hspace{1cm} (2.11)

Starting in the Cluster-Undecided state, each node broadcasts its mobility metric to its neighbors periodically. For a set of Cluster-Undecided nodes, the one with the lowest local value of $M$ declares itself Cluster-Head statue among its one-hop neighbors. If a Cluster-Undecided node hears a Cluster-Head statue form a one-hop neighbor, it joins and becomes a Cluster-Member. Thus, the cluster diameter is at most two hops. Re-clustering only occurs when two clusterheads do not hear from each other longer than
Cluster-Contention-interval (CCI).

Although MOBIC takes the mobility factor into account, it is not suitable for vehicular communication. One basic problem is that the mobility estimation is based on the assumption that $RxPr_{X \rightarrow Y}$ is proportional to the transmission distance between nodes. However, the signal strength is affected by many environmental factors, such as interference from other nodes, shadow fading and etc.

2.3.3 $(\alpha, t)$-Cluster Framework

McDonald et.al. have proposed a framework whose objective is to maintain an effective topology to adapt to node mobility. [20] The $(\alpha, t)$-Cluster framework partition the network into clusters, however, no clusterhead exists in these clusters. All members in a same cluster will exchange and store others’ information. Nodes belong to a same cluster are mutually available with a probability of at least $\alpha$ during time interval $t$. Only nodes connected stable enough in time interval $t$ can form a cluster. Cluster maintenance and a routing algorithm for the framework are also discussed in [20].

The path availability is defined as:

$$\Pi^k_{m,n}(t) = Pr(\mathcal{P}^k_{m,n}(t_0 + t) = 1|\mathcal{P}^k_{m,n}(t_0) = 1) \quad (2.12)$$

where $\mathcal{P}^k_{m,n}(t)$ is the status of path $k$ from node $n$ to node $m$ at time $t$, it is equal to 1 if the path is availability, and 0 otherwise.

If node $n$ and $m$ have $\Pi^k_{m,n}(t) \geq \alpha$, they are $(\alpha, t)$-available. An $(\alpha, t)$-cluster is a set of $(\alpha, t)$-available nodes. Thus, network is partitioned based on path availability.

However, in a dynamic network, how to decide the value of $\alpha$ and $t$ is still a problem. $(\alpha, t)$-Cluster Framework focuses on the weight of link. It partitions the network into clusters, and in a cluster, there exists no clusterhead. Nodes are equal members. Different routing algorithms are employed for inter-cluster routing and intra-cluster routing.
2.4 Clustering Algorithms in VANETs

Recently, more and more researchers pay attention on clustering in VANETs. Shea et.al. have propose APROVE (Affinity PROpagation for VEhiclar networks) for clustering a vehicular network. [28] Affinity propagation is an algorithm for image processing, and APROVE has proved that its distributed case can be utilized for VANETs. Wolny et.al. have proposed a MDMAC (Modified Distributed Mobility Adaptive Clustering) algorithm for VANETs. MDMAC adapts parts of DMAC to meet the road traffic mobility patterns in a vehicular network. [32] Since our clustering algorithm, CCA, has the same basic thought as MDMAC, we introduce some details about MDMAC in the following section.

2.4.1 Modified DMAC Algorithm

DMAC (Distributed Mobility Adaptive Clustering) algorithm is proposed to partition nodes of a mobile ad hoc network by Basagni et.al. [3] It is also a clustering algorithm which selects the best node as the clusterhead in MANET.

DMAC emphasizes on the operation framework of clustering. While each node is assigned a weight depending on any possible parameter, DMAC discusses the states of nodes in a clustering structure and the transaction between different states. Even employing same metric as node weight, DMAC proves better performance compared with Lowest-ID and Highest-Degree clustering. And the improvement of performance is caused by the operation mechanism of DMAC. DMAC also allows mobility-related parameter used as node weight. [4]

However, DMAC is still a clustering algorithm in MANETs. It assumes that the network topology stays the same during the operation of the algorithm. Thus, it is suitable for quasi-static ad hoc networks.

As an improvement of DMAC, MDMAC also focuses on discussing the framework of clustering algorithm. It defines node weight as its ID and compares the performance of
MDMAC with that of DMAC. It proves that under a highly mobile scenario, MDMAC needs less updates and less cluster changes. More stable clusters are provided in a dynamic network topology.

There are 3 states for each node in MDMAC: Init, Clusterhead, and Clusternode. We discuss the execution procedure of MDMAC in the following description.

When a node first begins the clustering algorithm, it marks itself in Init state. Through periodically broadcasting Hello message, it collects neighbors’ information and compares neighbors’ node weight with its own node weight. If one of its neighbors is a clusterhead and has a better node weight, the node will join the neighbor’s cluster. Otherwise, it will declare itself become a clusterhead and enters Clusterhead state. This step is the same as DMAC. It selects node with the highest weight to be a clusterhead. Each node is a clusterhead or connects to clusterhead.

The difference between DMAC and MDMAC is on the procedure when a node finds a new node or receives a message from a clusterhead. For example, if a node $u$ receives a Hello message from a new node $v$, it first judges whether $v$ is a clusterhead or not. If $v$ is a clusterhead which is better than $u$’s current clusterhead, different procedures are employed in DMAC and MDMAC. In DMAC, $u$ sends a request to affiliate with the new node $v$, no matter what state $u$ is in. However, in MDMAC, two more conditions are required for $u$ to join the new clusterhead. First, the estimated connection time should be greater than a predefined threshold. Second, the angle between nodes should be less than the threshold. The connection time is estimated by the current position and velocity of nodes.

MDMAC is able to form k-clusters, i.e. clusternodes can be $k$-hop away from clusterhead. The value of $k$ is determined by the TTL (time-to-live) parameter. However, one problem is that the estimated connection time is calculated based on 1-hop neighbors which means that node join a cluster only when it can communicate with its clusterhead. It is a problem caused by the $k$-clusters. In our algorithm, we fix this problem by intro-
ducing a new parameter. More details about DMAC and MDMAC algorithm is found in [3], [4], and [32]

2.5 Summaries

The clustering algorithms for MANETs can be divided into two different types: the node-based clustering and the link-based clustering. Lowest-ID, Highest-Degree, and MOBIC belong to the node-based clustering. [4][5][7] In the heuristic, each node is assigned a weight. The value of the node weight reflects how suitable the corresponding node acts as a clusterhead. ($\alpha$, $t$)-cluster framework is a typical link-based heuristics. If a link availability is above a predefined threshold, the two ends of a link are within a same cluster. [20] The link-based clustering does not target at forming clusters containing clusterheads.

The two heuristics are employed in different scenarios to cluster networks. The node-based heuristic captures the instant node metric. It is suitable for highly dynamic network, while the fast reflection requires a large transmission overhead. The link-based one form clusters based on an empirical threshold. If the threshold is well-selected, the cluster is stable. However, the whole cluster structure depends on the value of the threshold. The unique threshold determines all clusters formation without considering the variable factors in a dynamic network. And a convergence time is required for the cluster formation.

MDMAC is a combination of the node-based heuristic and the link-based heuristic. Like the node-based heuristic, it selects clusterheads based on the node weight, while a link threshold is added. The link threshold is used to judge whether a cluster merge phase should be activated when two or more clusterheads meet. MDMAC employs LET as the link weight. When the value of LET is larger than the link threshold, the cluster merge phase is activated. In MDMAC, the node weight is used to evaluate the suitability of
being a clusterhead, while the link weight excludes unstable factors in a cluster formation phase. MDMAC inherits the advantages of both heuristics.
Chapter 3

Performance Metric for Vehicular Networks

Clustering algorithms partition the network into sub-networks each of which having certain properties. To achieve this goal, we need to address two problems: clustering methods and distance measure. Clustering methods are the approaches to assign nodes into clusters. In Chapter 2, different clustering algorithms have been discussed. Distance measure, or similarity measure, provides a metric to assess the closeness of nodes, which is fundamental to the clustering problem. In this chapter, we provide more details about distance measure.

3.1 Clustering Problem

Mathematical formulation of the clustering problem is as follows:

Let us define a binary variable $y_{im}$:

$$ y_{im} = \begin{cases} 1, & \text{if } v_i \in C_m \\ 0, & \text{otherwise} \end{cases} $$  \hspace{1cm} (3.1)

where $C = \{C_m\}, m = 1...k$ is the set of clusters. $y_{im}$ is a variable to indicate which
cluster node $v_i$ belongs to: If $y_{im}$ belongs to cluster $C_m$, $y_{im} = 1$; otherwise, $y_{im} = 0$.

The mathematical formulation of clustering problem is shown in Equation 3.2.

$$\min \sum_{m=1}^{k} \sum_{i=1}^{n} y_{im} d_{im}$$

$$\sum_{m=1}^{k} y_{im} = 1, i = 1, ..., n$$

$$\sum_{i=1}^{n} y_{im} \geq 1, m = 1, ...k$$

$$y_{im} \in \{0, 1\}$$

(3.2)

where $d_{im}$ is the distance to reflect the closeness between node $v_i$ and cluster $C_m$.

However, Equation 3.2 is a theoretical objective function. The optimal solution for Equation 3.2 can only resolved through centralized methods, which indicates that at least one node can get the global network topology and update its change in time. Obviously, it is really hard to find such a node in vehicular communication networks. In high-dynamic and large-scale vehicular networks, the objective function has to be solved in a centralized way to seek an approximating optimal solution. Hence, many distributed clustering algorithms, such as MOBIC, WCA, and DMAC (see Chapter 2), are proposed. As we analyzing in Chapter 2, seldom algorithms are proposed for highly mobile vehicular networks. That’s why we propose our own clustering algorithm in Chapter 4.

### 3.2 Clustering Performance Metric

A novel metric, criticality, to measure distance between nodes is introduced as the clustering metric. Criticality is not a perceptual metric. It requires a weight matrix, and then the target criticality is derived from the weight matrix based on the random walk theory. In the following, we first present a reasonable weight matrix. The concept of criticality is explained, and we define point-to-point criticality and node criticality to implement the usage of criticality in clustering algorithms. Finally, we reveal more interesting properties about criticality, which explain why criticality is suitable for clustering algorithms.
### 3.2.1 Link Expiration Time (LET)

*LET* is a mobility prediction metric that considers the current distance and relative velocity between two nodes. *LET* estimates the expiration time when the distance between the nodes will exceed a given threshold. It is used in many routing protocols for VANETs. In [16], *LET* is used as the main metric for clustering ad hoc networks. We adopt *LET* for our clustering method; however, we make some improvement to let it more suitable for VANETs.

With the help of GPS, we can get the position and velocity of each vehicle. Let \((x_i, y_i)\) be the coordinates of vehicle \(i\), and \((v_{x_i}, v_{y_i})\) be the velocity of vehicle \(i\) at time \(t\). When the distance between node \(i\) and \(j\) is smaller than the transmission range \(r\), which is the transmission range defined by MAC protocol, then

\[
LET_{ij} = \frac{-(ab + cd) + \sqrt{(a^2 + c^2)r^2 - (ad - bc)^2}}{a^2 + c^2}
\]  

(3.3)

where \(a = v_{x_i} - v_{x_j}\), \(b = x_i - x_j\), \(c = v_{y_i} - v_{y_j}\), \(d = y_i - y_j\). The range of LET is \([0, \infty)\). *LET* considers not only current relative distance between nodes, but also relative velocity. It predicts the stability of links to some extent. A large value of *LET* means a stable link, i.e. it lasts for long time.

Deriving from LET, Barghi et.al. have proposed a new metric to evaluate the stability for each link [2]. The metric, which is named as the stability function, is denoted as:

\[
S_{ij} = 1 - e^{-\frac{LET_{ij}}{\delta}}
\]  

(3.4)

where \(\delta\) is a constant deciding the rising rate of \(S_{ij}\). As proven in [2], the rising rate determines how sensitive \(S_{ij}\) to the change of \(LET_{ij}\). Figure 3.1 shows how \(\delta\) impacts on \(LET_{ij}\): \(LET_{ij}\) is mapped to \(S_{ij}\), which is limited to \((0, 1]\). Therefore, we conclude that larger \(S_{ij}\) indicates a more stable link.

Due to the following reasons we use \(S_{ij}\) instead of \(LET_{ij}\). First, the stability function
Chapter 3. Performance Metric for Vehicular Networks

Figure 3.1: The impact of $\delta$ on $S$[2]

maps $S_{ij}$ to $(0, 1]$, which is more scalable than $LET_{ij}$, whose range is from 0 to $\infty$.

Secondly, this metric is calculated at a time $t$ to predict the link in the future. Only when the velocities of all nodes are fixed, $LET_{ij}$ can reflect the mobility prediction precisely. However, the assumption is impossible in the real world. While the velocities may be fixed in a short period, the ongoing acceleration and deceleration will change the velocities frequently. The stability function enhances the effect of smaller LET and makes the stability metric, $S_{ij}$, more reasonable.

3.2.2 Criticality

$S_{ij}$ only represents the stability of a direct link. It can only reflect the relationship between a node and its one-hop neighbors. Besides $S_{ij}$, many other metrics used for clustering in VANETs have the same feature, such as the highest degree metric in [7], the aggregate mobility metric in [5]. This explains why algorithms are usually limited to one-hop clusters. If we want to build a multi-hop cluster structure, $S_{ij}$ is not enough.
Therefore, criticality is introduced to partition the network.

The concept of network criticality is proposed in [31]. Network criticality is a global measure on a graph which quantifies the robustness of a network graph to the environmental changes, mainly traffic shifts, topology modifications, and changes in the origin and destination for traffic.

In graph theory, betweenness is a centrality measure for both nodes and links. The random-walk betweenness is an important betweenness centrality in analyzing communication network. It provides a probabilistic approach to analyze the betweenness. Assume a VANET is a connected graph, a random walk occurs when node $i$ travels to node $j$. The random walk forms a finite-state irreducible Markov Chain. The random-walk betweenness is determined by the transition probabilities of the walk. We assume that the transition probabilities are defined by the weight of link as follows:

$$p_{ij} = \begin{cases} \frac{w_{ij}}{\sum w_{ij}}, & \text{if } w_{ij} \neq 0 \\ 0, & \text{otherwise} \end{cases}$$ (3.5)

Let $P_d$ be the probability transition matrix when state $d$ is modified as an absorbing state. $P_d^q$ is the matrix of $q$-step transition probabilities for random walks that terminate at node $d$. $B_d = [b_{sm}(d)]$ denotes the $n \times n$ matrix of betweenness metrics, where $b_{sm}(d)$ denotes the betweenness of node $m$ is based on walks that start at node $s$ and terminate at node $d$. It has been shown [31][30] that:

$$[B]_d = [b_{sm}(d)]_d = \begin{cases} \sum_{q=0}^{\infty} P_d^q, & \text{if } m \neq d \\ 0, & \text{otherwise} \end{cases}$$ (3.6)

Equation 3.6 can be simplified as

$$[B]_d = [b_{sm}(d)]_d = \begin{cases} (I - P_d)^{-1}, & \text{if } m \neq d \\ 0, & \text{otherwise} \end{cases}$$ (3.7)
We use $P(i|j)$ to denote the truncated $(n-1) \times (n-1)$ matrix which deletes $i$-th row and $j$-th column of matrix $P$. The relationship between betweenness and transition probability is written as Equation 3.8.

$$[B]_d(d|d) = (I - P_d(d|d))^{-1} \quad (3.8)$$

The Laplacian of the graph is defined in Equation 2.3. From Equation 3.8, we have:

$$I - P_d(d|d) = I - M^{-1}(d|d) \times W(d|d)$$

$$\Rightarrow I - P(d|d) = M^{-1}(d|d) \times L(d|d) \quad (3.9)$$

From Equation 3.8 and 3.9, we conclude:

$$B_d(d|d) = M(d|d) \times L^{-1}(d|d) \quad (3.10)$$

To write $L^{-1}$ in terms of $L$, [31] uses the Moorese-Penrose generalized inverse matrix of $L$. Equation 3.10 can be rewritten as:

$$\left((B_d(d|d))_s\right)_m = (l^+_sm - l^+_sd - l^+_dm + l^+_dd) \times W_m$$

$$\text{where } W_m = \sum_{m=1}^{n} w_{im}. \quad (3.11)$$

A simple relationship between $b_{sk}(d)$ and $b_{dk}(s)$ is: [30]

$$\frac{b_{sm}(d) + b_{dm}(s)}{W_m} = l^+_ss - 2l^+_sd + l^+_dd \quad (3.12)$$

**Definition 3.1** The network criticality is defined as [31]:

$$\tau = \sum_{(s,d)} \frac{b_{sm}(d) + b_{dm}(s)}{W_m} = \sum_{(s,d)}(l^+_ss - 2l^+_sd + l^+_dd) \quad (3.13)$$
Network criticality is a global metric on graphs. It can be interpreted as the total resistance if we view the network as an electrical circuit [29]. Consider an electrical circuit with the same graph as our original network graph, and with link resistances equal to the reciprocal of link weights. It can be shown that network criticality numerically equals the total resistance distance (effective resistance) seen between different pairs of nodes in the electrical circuit. [31] A high network criticality is an indication of high resistance in the equivalent electrical circuit; therefore, in two networks with the same number of nodes, the one with lower network criticality is better connected. Furthermore, network criticality quantifies the sensitivity of a network to the environmental changes. It has been shown that network criticality equals the average of link betweenness sensitivities, where link betweenness sensitivity is defined as the partial derivative of link betweenness with respect to the corresponding link weight [29]:

$$\tau = \frac{1}{m - 1} \sum_{(i,j) \in E} \frac{\partial b_{ij}}{\partial w_{ij}}$$

Equation 3.14 states that minimization at network criticality results in minimizing the average sensitivity of link betweennesses with respect to the changes in link weights. Environmental changes are captured by the formula.

Having lower criticality brings the robustness enhancement to the network. Suppose that a node is failing or becoming inaccessible so that it is unable to route the traffic passing through it. Minimizing the criticality adaptively results in adjusting the betweenness in such a way that traffic is re-routed to other nodes instead of the impaired one and that brings about higher robustness against unpredictable deleterious situations. We can also define point-to-point or pair-wise network criticality:

**Definition 3.2** We define $$\tau_{ij} = l_{ii}^+ - 2l_{ij}^+ + l_{jj}^+$$ as the point-to-point criticality between the node pair $$(i, j)$$, where $$l_{ij}^+$$ denotes the entry $$[ij]$$ of Laplacian matrix of the graph [31]. $$\tau_{ij} = \tau_{ji}$$ since $$L^+$$ is a symmetric matrix.
Chapter 3. Performance Metric for Vehicular Networks

Note that the point-to-point criticality doesn’t mean that there is a physical link between node \( i \) and \( j \). It could exist between any pair of nodes.

It can be shown that \( \tau_{ij} \) evaluates the commute time between node \( i \) and \( j \). The lower value of \( \tau_{ij} \) means that the node pair \((i, j)\) is less sensitive to the changes in the topology of the network.

Based on \( \tau_{ij} \), we define node criticality as follows.

**Definition 3.3** We define \( \tau_i = \frac{1}{N} \sum_j \tau_{ij} \) as the node criticality of node \( i \), where \( j \) denotes any node, except \( i \), in the graph; and \( N \) is the number of nodes in the graph.

If we take stability metric as the weight matrix, the corresponding random walk then represents the stability between nodes.

### 3.2.3 Metric Analysis

\( \tau_{ij} \) can be viewed as a square Euclidean distance. This is the basis that we declare the possibility of using \( \tau_{ij} \) and \( \tau_i \) as the clustering metrics. If we do the spectral decomposition of \( L^+ \), a corresponding spectral coordinate reveals more details about the distance measure. It proves that the spectral coordinate projects the network topology into a well-formed blocked structure.

**Distance Measure**

The advantage of \( \tau_{ij} \) is that it gives a quantitative metric between any pair of nodes, no matter whether a direct link exists between the two nodes or not. Evaluating the robustness of communication between node pair \((i, j)\), it is natural that we select \( \tau_{ij} \) as the closeness metric \( d_{ij} \). However, Equation 3.2 requires \( d_{im} \), which is the robustness between a node and a cluster. To resolve the problem, \( \tau_{ij} \) should have the properties of Euclidean distance.

Let \( u_i \) be the \( i \)-th unit vector, i.e. \( u_i = [00...1...0]^t \), where 1 is located at \( i \)-th position. Define \( u_{ij} = u_i - u_j \). \( \tau_{ij} \) can be written as:
\[ \tau_{ij} = u_{ij}^T L^+ u_{ij} \]  \quad (3.15)

\[ \Rightarrow \tau_{ij} = u_{ij}^T L^+ \frac{1}{2} L^+ \frac{1}{2} u_{ij} = (L^+ \frac{1}{2} u_{ij})^T (L^+ \frac{1}{2} u_{ij}) \]

Assume \( V = (L^+)^{\frac{1}{2}} u_{ij} \), we have

\[ \tau_{ij} = \| V \|^2 \] \quad (3.16)

Hence, \( \tau_{ij} \) can be viewed as a square Euclidean distance: \( \tau_{ij} = \| V \| \) and \( \tau_i \) is the average square Euclidean distance of node \( i \) to its neighbors. This property is important for clustering algorithms, since clustering algorithms classify different clusters based on the distance among nodes.

The concept of \( \tau_{ij} \) can be analogized to the concept of the commute time in graph theory. The commute time \( c_{ij} \) is the expected time taken for a random walk leave from node \( i \) to reach node \( j \) and return to \( i \) [27]. To explain \( c_{ij} \), the first passage time is introduced first. The first passage time from node \( i \) to node \( j \) in a random walk is defined as the number of transitions which initializes at node \( i \) and visits node \( j \) for the first time. Assuming that \( f_{ij} \) denotes the first passage time from \( i \) to \( j \), the commute time can be represented as follows:

\[ c_{ij} = f_{ij} + f_{ji} \] \quad (3.17)

where \( c_{ij} \) is the commute time. \( c_{ij} \) can be explained as the average steps of a random walk, which is starting at \( i \), will take to reach the destination \( j \) and then return to \( i \). The commute time is also called the commute distance. The commute distance in [18] is described as Equation 3.18.
where \( \text{vol}(V) \) is a constant which is calculated by summing over all \( w_{ij} \) in weight matrix \( W \).

The difference between \( c_{ij} \) and \( \tau_{ij} \) is caused by the method to deal with the intermediate node \( k \). We can derive \( c_{ij} \) from \( \tau_{ij} \) as follows:

Based on Equation 3.12, we have

\[
b_{im}(j) + b_{jm}(i) = W_m(l^+_{ii} - 2l^+_{ij} + l^+_{jj})
\]

\[
\sum_{m=1}^{n} b_{im}(j) + b_{jm}(i) = \sum_{m=1}^{n} W_m(l^+_{ii} - 2l^+_{ij} + l^+_{jj})
\]

\[
\Rightarrow \tau_{ij} \sum_{m=1}^{n} W_m = \text{vol}(V)l^+_{ii} - 2l^+_{ij} + l^+_{jj}
\]

\[
\Rightarrow \text{vol}(V)\tau_{ij} = c_{ij} \tag{3.19}
\]

Note that \( \text{vol}(V) \) is a constant. Thus we conclude that \( \tau_{ij} \) is a linearly dependent function of \( c_{ij} \). The commute time embedding is related to unnormalized spectral clustering.

**Spectral Decomposition**

Suppose the spectral decomposition of \( L \) is \( L = \Phi \Lambda \Phi^t \), where \( \Lambda = \text{diag}(\lambda_1, \lambda_2, ..., \lambda_n) \) is the diagonal matrix with eigenvalues ranging in a ascending order, and \( \Phi = (\phi_1, \phi_2, ..., \phi_n) \) is the matrix of corresponding eigenvectors. Combining the properties of \( L \), we have

\[
0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n.
\]

The Moore-Penrose pseudo-inverse matrix of \( L \) can be denoted as:

\[
L^+ = \sum_{i=2}^{n} \frac{1}{\lambda_i} \phi_i \phi_i^t \tag{3.20}
\]

Using Equation 3.20 to replace \( L^+ \) in Equation 3.15,
\[ \tau_{ij} = \sum_{m=2}^{n} \frac{1}{\lambda_m} u^t_{ij} \phi_m \phi^t_m u_{ij} \]
\[ = \sum_{m=2}^{n} \frac{1}{\lambda_m} (u^t_{ij} \phi_m)(u^t_{ij} \phi_m)^t \]
\[ = \sum_{m=2}^{n} \frac{1}{\lambda_m} (u^t_{ij} \phi_m)^2 \]
\[ \Rightarrow \tau_{ij} = \sum_{m=2}^{n} \frac{1}{\lambda_m} (\phi_{mi} - \phi_{mj})^2 \] (3.21)

Equation 3.21 indicates $\tau_{ij}^{1/2}$ is a Euclidean distance metric and the metric space is formed through spectral decomposition. The coordinate vector for each node is $(\lambda_m^{1/2} \phi_m)$, where $m = 2, \ldots, n$.

Based on these coordinate, we can get the corresponding distance in the metric space. From the view of a clustering algorithm, if distance is large, the possibility that two nodes belong to different communities is high. If distance is small, the nodes are in the same community with high possibility.

As mentioned in Chapter 2, spectral clustering converts the original nodes into another metric space. The metric space utilizes $k$ eigenvectors of the Laplacian matrix corresponding to the first $k$ smallest eigenvalues, where $k$ is a predefined value. We conclude that the coordinate of each node is $(\phi_m)$, where $m = 1, 2, \ldots, k$. Compared to Equation 3.21, the coordinate of each node is $(\lambda_m^{-1/2} \phi_m)$, where $m = 1, 2, \ldots, n$.

Spectral clustering uses the first $k$ eigenvectors while $\tau_{ij}$ contains information from all eigenvectors. $\tau_{ij}$ also takes into account the eigenvalues. Figure 3.3 to Figure 3.5 shows the corresponding result for an example in Figure 3.2. Based on the analysis from Qiu et. al. [27], it is clear that $\lambda_m^{-1/2} \phi_m$ reveals a stronger block-diagonal structure of the weight matrix.

The final step of spectral clustering declares that any traditional clustering algorithm can be used to cluster nodes, and $k$-means clustering algorithm is the common choice. The objective function of $k$-means is error square function, which is similar to our objective function. Viewing $\tau_{ij}$ as a square distance between node $i$ and node $j$, the intuitive choice
is used $k$-means for our clustering based on $\tau_{ij}$. However, our cluster structure wants to tell the clusterhead and the member. Therefore, $k$-medoid clustering algorithm is a suitable choice. As we described in Chapter 2, the simple-$k$-medoid clustering algorithm keeps the objective function of $k$-means clustering while it avoid the impact of outliers. In the following section, we analyze a simple-$k$-medoid clustering algorithm proposed for commute time. Tests show that the clustering algorithm keeps the advantages when the metric changes to $\tau_{ij}$.

### 3.3 Centralized Clustering Algorithm

In this section, we’d like to test the performance of $\tau_{ij}$ in a centralized clustering algorithm. The generalized objective function of clustering is stated in Equation 3.2. As mentioned in Chapter 2, $k$-medoids clustering algorithm is a suitable choices to solve such a minimization function. K-medoids clustering is similar to $k$-means clustering, but the center of each cluster is a existing node named medoid instead of the centroid in $k$-means.
An example utilizing the k-mediod clustering is ECT k-means clustering algorithm. [34] ECT k-means clustering algorithm utilizes the commute time as the clustering metric, and employs the simple-k-medoids clustering to solve the objective function. In this case, $\tau_{ij}$ is selected as the clustering metric.

In k-medoids clustering, the objective function is to minimize an error square function defined as:

$$
\sum_{m=1}^{k} \sum_{i=1}^{n} x_{im} \text{dist}_{iC_m}^2
$$

where $\text{dist}_{iC_m}$ denotes the distance between node $i$ and a cluster $C_m$. K-medoids defines that the distance between a node and its cluster is the distance between the node and the cluster’s medoid, i.e. $\text{dist}_{iC_m} = \text{dist}_{ic_m}$ where $c_m$ is the medoid of cluster $C_m$.

Employing the distance based on the spectrum space,
Figure 3.4: Two-dimension for $\lambda_m^{\frac{1}{2}} \phi_m$ and $\phi_m$, $m = 2, 3$

Figure 3.5: Three-dimension for $\lambda_m^{\frac{1}{2}} \phi_m$ and $\phi_m$, $m = 2, 3, 4$

\[
dist_{icm} = \tau_{icm}^{1/2} \tag{3.23}
\]

\[
=> \ dist_{icm}^\nu = \tau_{icm} \tag{3.24}
\]

Combining above derivation, we have

\[
d_{im} = \tau_{im} = \tau_{icm}
\]

Finally,
Chapter 3. Performance Metric for Vehicular Networks

\[
\min \sum_{m=1}^{k} \sum_{i=1}^{n} x_{im}d_{im} = \min \sum_{m=1}^{k} \sum_{i=1}^{n} x_{im}\tau_{icm} \tag{3.25}
\]

\[
\sum_{m=1}^{k} x_{im} = 1, i = 1, ..., n
\]

\[
\sum_{i=1}^{n} x_{im} \geq 1, m = 1, ...k
\]

\[
x_{im} \in \{0, 1\}
\]

Equation 3.25 connect the objective function to our metric, \(\tau_{ij}\). Two factors determine the value of Equation 3.25: the assignment of nodes to a cluster, and the centers of clusters.

In the simple-k-medoids clustering algorithm, two basic steps operate iteratively until converging to an optimal solution of Equation 3.25 by the simple-k-medoid clustering algorithm:

1. Assign each node \(u\) to its nearest cluster, \(C_p\):

\[
p = \arg\min_{m} \tau_{uC_{m}} = \arg\min_{m} \tau_{ucm} \tag{3.26}
\]

2. Select the center, \(c_p\), of each cluster among current nodes belongs to the cluster:

\[
c_p = \arg\min_{u} \sum_{x_{um} \neq 0} \tau_{uc_{p}} \tag{3.27}
\]

When the procedure achieve a local minimum, stop.

Taking Figure 3.2 as an example, we observe the clustering results based on different centralized clustering algorithms. Figure 3.6 employs the k-means clustering algorithm, and utilizes the physical distance as the clustering metric. Physical distance cannot tell the relative similarities among different nodes.

If we use the simple k-medoids clustering algorithm and the physical distance, as shown in Figure 3.7, it cannot make the result better. Although the k-medoids clustering
algorithm is good at dealing outliers, the advantages doesn’t help to the specific example.

Figure 3.8 represents the result of the k-medoids clustering algorithm based on \( \tau_{ij} \). Obviously, \( \tau_{ij} \) enlarge the relative similarities among nodes and give an exact result.

Figure 3.8 is just a simple example to show that \( \tau_{ij} \) is able to reveal similarities among nodes. This example is still far from employing \( \tau_{ij} \) in vehicular networks. In fact, there exists some wireless networks employing the centralized clustering algorithm. Yanhua Li et.al. have proposed a graph partitioning algorithm in large-scale wireless network [11]. Their algorithm utilizes the spectral algorithm periodically to partition a large-scale wireless network. A powerful node operates the algorithm and pass the decision to other nodes. We cannot say that it is impossible to find such a powerful node in vehicular networks. Actually, BS is a candidate for a centralized clustering algorithm. However, vehicular networks have more variable topology compared to the large-scale wireless network discussed in [11]. It doesn’t offer enough time to collect the information
Figure 3.7: The simple k-medoids clustering algorithm: physical distance, ‘*’ is the centroid of cluster.

Figure 3.8: The simple k-medoids clustering algorithm: $\tau_{ij}$, ‘*’ is the centroid of cluster.
from other nodes, test the optimal number of $k$, and then pass the results to the whole network. Thus, the distributed method indicates a potential clustering algorithm. The centralized algorithm is promising if we can resolve the effectiveness problem. We will explore its possibility in the future work.

### 3.4 Local Clustering Parameter

$\tau_{ij}$ and $\tau_i$ is defined on the basis of a global network topology. If we want to utilize them for a distributed method, we should give them new definitions based on a local network. In this section, we extend the definition of $\tau_{ij}$ and $\tau_i$. $\tau^{local}_{ij}$ and $\tau^{local}_i$ are defined for the distributed clustering algorithm.

#### 3.4.1 Local Network

Figure 3.9: A snapshot of a vehicular network

As we see, the calculation of $\tau_{ij}$ requires a weight matrix derived from the global network topology. All links of a graph $G$ are required to calculate $\tau_{ij}$. However, one disadvantage of $\tau_{ij}$ is that any trivial change in the network topology leads all $\tau_{ij}$ to be calculated again. In vehicular communication networks, the network topology is highly dynamic, and the centralized server is connected through 3G, whose capacity is limited and expensive. Therefore, the centralized calculation of $\tau_{ij}$ is unacceptable for both the effective and economic consideration. It is impossible to calculate $\tau_{ij}$ in a centralized way.
To employ $\tau_{ij}$ in our distributed clustering algorithm, we propose a local $\tau_{ij}$. The local $\tau_{ij}$ is marked as $\tau_{ij}^{\text{local}}$, and its value derives from a local network topology. The network topology detected by each node is called the local network topology of the node. In wireless network, the local network topology is learned by exchanging Hello messages between nodes.

One important parameter deciding the size of a local network is TTL (Time-To-Live) parameter. TTL tells us how many hops a message can transmit from its source node. In wireless network with the TTL mechanism, each message contains a TTL field. Suppose the number of TTL is set to $T$, a source node transmits a message with $TTL = T$. Nodes receiving the message check the TTL field, if $TTL > 1$, the message is forwarded to all its one-hop neighbor with the value of TTL decremented by 1; otherwise, the forwarding of message is stopped. The procedure repeats until there is no $TTL = 0$. Note that a node that receives duplicated messages will discard the extra messages. The TTL mechanism helps nodes to form a local topology from its $T$-hop neighbors.
node network topology is given in Figure 3.9. The network topology is a snapshot of a vehicular network at time $t$. We know the position coordinates and velocities of each node. Figure 3.10 shows the 1-hop, 2-hop and 3-hop local network topology of node 7 as an example.

### 3.4.2 Local Clustering Metric

Based on the local network topology, we define the clustering metric for the distributed method:

$$\tau_{ij}^{local} = \tilde{l}_{ij}^+ - 2\tilde{l}_{ij}^+ + \tilde{l}_{jj}^+$$  \hspace{1cm} (3.28)

where $\tilde{l}_{ij}^+$ is a entry in $\tilde{L}^+$. $\tilde{L}^+$ is the Moorese-Penrose inverse matrix derived from the local network topology.

The local network topology assumes each node is not aware of the existence nodes without connections to the node. The merit is that the local topology forms a subgraph of $G$ which is always connected. Thus, the assumption that graph is connected is always met in the calculation of $\tau_{ij}^{local}$. Note that each node has its own local topology, which leads $\tau_{ij}^{local}$ may not be the same in node $i$ and node $j$. When we mention $\tau_{ij}^{local}$, we mean $\tau_{ij}^{local}$ in a specific node. Takeing node 7 in Figure 3.10 as an example, Figure 3.11 shows the corresponding $\tau_{ij}^{local}$. To simplify the example, we assume that they are static nodes, i.e. $LET$ is equal to 1 when there exists a link. On the basis of the network topology in Figure 3.10, it is not difficult to tell that node 8, 9, and 10 have the strongest connection to node 7; node 3 has the weakest connection since it only joins the local network when $TTL = 3$. From $LET$, all links are equal to 1 in the static case. However, $\tau_{ij}^{local}$ analyzes the connections between node 7 and other nodes, and provides a more reasonable metric.

Based on $\tau_{ij}^{local}$, we defined local node criticality is:
Figure 3.11: Example: point-to-point criticality in $T$-hop local network (node 7)

\[ \tau_{local}^i = \frac{1}{|n_i|} \sum_{j=1}^{n_i} \tau_{ij}^{local} \]  

(3.29)

where \( \{j\}, j = 1, ..., n_i \) are nodes detected by node \( i \), and \( |n_i| \) is the number of nodes detected by node \( i \).

Comparing \( \tau_{local}^i \) with Equation 3.22, we see that it is an error square function except a factor \( \frac{1}{|n_{ij}|} \) is added in Equation 3.29, if node \( i \) assumes itself is a medoid and there is only one cluster in its local network. Considering that the number of $T$-hop neighbors are different in different nodes, the factor \( \frac{1}{|n_{ij}|} \) is used to let the value of \( \tau_{local}^i \) normalized. \( \tau_{local}^i \) reflects a normalized quality of the network topology node \( i \) is able to achieve if \( i \) is selected as a clusterhead. In each node, the distributed clustering algorithm is considered as a simplified version of k-medoids clustering algorithm. The network is a local network and \( k \) is predefined as 1.

The calculation of \( \tau_{local}^i \) is inspired by ego networks. Ego networks are defined as networks which contain a single node (ego) with other nodes (alters) connected to ego.
Chapter 3. Performance Metric for Vehicular Networks

All nodes and links among these nodes constitute corresponding ego networks. In social networks, research focuses on the relationship between ego metrics and metric based on a completed network. Everett et al. have found that ego betweenness has a high correlation with normal betweenness. The concept of ego networks is similar to the concept of the local networks. In the real wireless networks, the transmission mechanism determines that nodes only detect their $T$-hop neighbors. Definitely, $\tau_i$ takes all paths between node $i$ and other nodes into account. However, if paths are more than $T$-hop, the real applications will never use them. Depending on different local network topology, $\tau_{i}^{local}$ evaluate the robustness of node $i$ in a detectable range. Employing $\tau_{i}^{local}$ to judge the robustness of node $i$ is more reasonable for clustering networks.

Therefore, in our distributed algorithm, each node calculates and broadcasts its $\tau_{i}^{local}$ to join the medoid selection. The node with smallest $\tau_i$ is the medoid we look for the cluster, and naturally, the medoid becomes the clusterhead of the cluster. The size of clusters is limited by TTL.
Chapter 4

Clustering Algorithm Using Network Criticality and Performance Evaluation

In this chapter, we discuss the distributed clustering algorithm. The architecture of vehicular communication network implies that the centralized server cannot implement clustering algorithm effectively while the distributed clustering algorithm cannot guarantee an objective function like k-medoids clustering. Consequently, we propose our own distributed clustering algorithm. The distributed clustering algorithm targets partitioning network into several robust clusters. Robust clusters consider not only the instant topology, but also the future changes of the network topology.

4.1 Distributed Clustering Algorithm

Based on the local clustering metrics, $\tau_{ij}^{local}$ and $\tau_{i}^{local}$, we proposed our own distributed clustering algorithm, the clustering algorithm based on network criticality, i.e. criticality-based clustering algorithm. The clustering algorithm keeps the advantages of the node
weight heuristic and the link weight heuristic. The node weight, $\tau_{local}^i$ values the average robustness of node $i$ as a potential clusterhead, which the link weight, $\tau_{ij}^{local}$ values the robustness between two nodes. The proposed algorithm proves an improvement in previous distributed clustering algorithms in MANETs.

4.1.1 States

The distributed clustering algorithm based on network criticality in a dynamic network involves two phases, the cluster formation and the cluster maintenance. Both phases help each other to generate a robust cluster structure. The local management of the hybrid vehicular network architecture is implemented by WLANs. To form and maintain a well-build cluster structure, 4 different states are available for each node.

- TempCH: When a node initially joins a network, or it disconnects to its cluster, it is in TempCH state. It can be viewed as a specific cluster which just has a clusterhead without members. In TempCH state, a node keeps its WLAN and WWAN interface activated.

- Idle: When a node in TempCH meets other nodes in TempCH state, it transfers to Idle state. Nodes in Idle state should finally transfer to CH state or Member state and form new clusters. It is the main process for the cluster formation. In the state, the cluster does not form yet, so we still keep both interfaces activated in each node.

- CH: The node already attends a cluster and it is the head of this cluster. The work of clusterhead includes: keeping track of all members, forwarding the request from member nodes to WWAN networks, managing the attendance and leave of member nodes. Both WWAN and WLAN interface of the clusterhead are working.

- Member: This state means that a node belongs to an existing cluster and is controlled by a clusterhead. The function of member nodes is much simpler than
clusterheads. It just needs to record the route to the clusterhead and send the WWAN access requests to clusterhead when needed. The WWAN interface turns off until its state changes.

Figure 4.1 summarizes the interaction diagram of states. The distributed algorithm combines the characteristics of the link weight heuristic and the node weight heuristic. If nodes in TempCH, CH, or Member state want to enter other state, they should satisfy both heuristics’ requirements, which enhance the stability of the cluster structure. The only exception is Idle state, which employ the node weight heuristic only. The details of the algorithm are described in the following part.

![Figure 4.1: State Transition](image)

4.1.2 Messages Types

The clustering algorithm employs the same message mechanism as DMAC and MDMAC. [3][4][32] We define the format of these messages to meet our requirement. It is a simple
message mechanism involves three types of messages:

**Hello Message, Hello:** It broadcasts periodically to announce the existing of a node. It contains: $(ID, State, ItsClusterhead, \tau_{local}^i, x, y, vx, vy, TTL, Timestamp)$.

- $ID$ is the unique identification for each node.
- $State$ denotes which states the corresponding node is in.
- The filed $ItsClusterhead$ used to store the current clusterhead of the corresponding node. If the node is in Member state, $ItsClusterhead$ stores the clusterhead in its cluster. Otherwise, it is equal to the node’s ID.
- $\tau_{local}^i$ is the main field which provides the node weight.
- With the help of GPS, Hello messages contain the position of each node and the velocity of each node, i.e., $(x, y)$ and $(vx, vy)$.
- $TTL$ determines the diameter of the local network as we introduced before.
- $Timestamp$ is used to maintain the neighbor list in each node. When a node receives Hello messages from other nodes, it first compares these messages with its neighbor list. If the node which sends Hello message has already recorded in the neighbor list with an older $Timestamp$, it updates $Timestamp$ and keeps the record. If it is the first time receives the Hello message from the node, a new record is added to the neighbor list. Another function of $Timestamp$ is to help each node delete old neighbors in its neighbor list. In the neighbor list, if $Timestamp$ of a record doesn’t change for a predefined period, the record should be deleted.

Through Hello messages, each node can generate and refresh its local network topology. Periodic Hello messages provide each node a neighbor list. The addition and deletion of neighbors in the list reflect the topology changes. The clustering algorithm operates on the basis of the up-to-date neighbor list.
Hello messages provide the neighbor list with the changing of the network topology. And Hello messages also indicate the cluster structure. For example, if a node is in TempCH state, the filed State in its broadcasting Hello message is TempCH and ItsClusterhead is equal to ID field.

Except Hello message, two more types of messages are used in our clustering algorithm. These messages drive specific procedures to implement the clustering algorithm.

**Clusterhead message, Clusterhead** (*u*): it broadcasts by clusterhead *u* to indicate the information of clusterhead and the corresponding cluster. It announces the existing of a cluster.

**Join message, Join** (*u*, *v*): When a node *u* wants to join a cluster with clusterhead *v*, it sends the request message.

Given the function of Clusterhead(*u*) and Join(*u*, *v*), their message format can be achieved by adding a field in hello message to announce the message type. Note that if there are more limitations for the clustering algorithms, such as the maximum cluster size, the capacity of the clusterhead, and so on, more fields should be added to provide corresponding information. In the thesis, we discuss some limitations for the clustering algorithm.

### 4.1.3 Algorithm Implementation

In this section, we explain the implementation of the *τ*-based clustering algorithm. As we described, the main roles in the cluster structure are the member and the clusterhead. Nodes in Member state know their clusterhead, and request their clusterheads for WWAN access. Nodes in Clusterhead state record all cluster members and management the connection requests for all members. The management makes wireless networks scalable. Two more states, TempCH and Idle, are introduced to improve the cluster structure.

Table 4.1 lists symbols used in our algorithm.

As mentioned before, TempCH can be viewed as a specific cluster with no member.
Table 4.1: Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>weight((u))</td>
<td>weight of node (u), i.e. (\tau_{\text{local}}^u)</td>
</tr>
<tr>
<td>CHweight((u))</td>
<td>weight of node (u)’s clusterhead</td>
</tr>
<tr>
<td>weight((u,v))</td>
<td>weight of link ((u,v)), i.e. (\tau_{\text{local}}^{uv})</td>
</tr>
<tr>
<td>link(\text{threshold})</td>
<td>threshold of link, i.e. (\tau_{\text{link}}^{\text{threshold}})</td>
</tr>
<tr>
<td>(N(u))</td>
<td>the neighbor list of (u)</td>
</tr>
<tr>
<td>(\text{CH}(u))</td>
<td>current clusterhead of node (u)</td>
</tr>
<tr>
<td>(\text{CHList}(u))</td>
<td>current clusterheads connect to node (u)</td>
</tr>
<tr>
<td>(\text{MemberList}(u))</td>
<td>members affiliate to (u) when (u) is a clusterhead</td>
</tr>
</tbody>
</table>

When a node joins the network, it begins with TempCH state. Table 4.2 lists the procedure executed by nodes in TempCH state. When node \(u\) in state TempCH finds a clusterhead in the neighbor list, \(u\) sends Join message to the clusterhead if two conditions are meet. First, the node weight of node \(v\) is smaller than that of node \(u\), i.e. \(\tau^v\) is smaller than \(\tau^u\). Second, \(\tau_{\text{local}}^{uv}\) is smaller than a predefined link threshold, \(\tau_{\text{link}}^{\text{threshold}}\). Note that \(\tau_{\text{local}}^{uv}\) is calculated by node \(u\) based on its local network topology. In our algorithm, \(\text{link}_{\text{threshold}}\) denotes the selected \(\tau_{\text{link}}^{\text{threshold}}\). Experiments prove that the value of threshold is important to the clustering result. In our simulation part, we also test different values of the threshold.

Now we explain why we set the two conditions for attending a cluster. The first condition is easy to understand, because \(\tau_{\text{local}}^i\) evaluates the robustness of a node providing if the node is the clusterhead of the local network. When a node wants to join a cluster, it compares its weight with the clusterhead’s weight to see whether the clusterhead can provide a more robust cluster than itself. Only when the clusterhead promises a better cluster, the node applies for joining the cluster.

The second condition employs the link-based clustering heuristic. Given that \(\tau_{\text{local}}^i\) reflects the average connectivity situation of the local network topology, a clusterhead may have a good average connectivity, while the link between the clusterhead and the targeted TempCH node is weak. The link threshold avoids a short period between joining
Table 4.2: Algorithm: TempCH State for node $u$

$CH_{weight}(u) \leftarrow weight(u)$
$CH(u) \leftarrow u$

for each $x \in N(u)$
    if $x \in CH_{List}(u)$ and $weight(x) < weight(u)$ and $weight(u, x) < link_{threshold}$
        $CH_{weight}(u) \leftarrow weight(x), CH(u) \leftarrow x$
    end
end
if $CH(u) \neq u$
    send Join($u, CH(u)$)
    $u$ enters Member state.
end
for each $x \in N(u)$
    if $x$ is in TempCH state
        $u$ enters Idle state
    end
end

and leaving a cluster, which guarantees a stable structure to some extent.

In Table 4.2, if node $u$ can not find a suitable cluster, it keeps TempCH state until meeting other nodes in TempCH state. These nodes transfer their state to Idle state.

TempCH is one characteristic of our clustering algorithm. It not only gives nodes an initial state, but also separates a specific type of the cluster from the cluster structure, clusters without members. TempCH, although, keeps WWAN interface open, it is not a real clusterhead. It don’t broadcast Clusterhead($u$) messages.

**TempCH State**

**Idle State**

Nodes in Idle state lead to new clusters. If nodes in TempCH cannot join suitable clusters, Idle state is triggered to form clusters. This rule guarantees cluster formation phase only occurs among Idle nodes. The node with lowest weight announces itself as a clusterhead through Clusterhead messages; other nodes wait for the appearance of a clusterhead and join the cluster. The mechanism guarantees nodes in Idle state finally
Table 4.3: Algorithm: Idle State for Node $u$

```
sel\text{\textit{selectCH}} \leftarrow \text{SelectClusterhead}
if \text{selectCH} = u
  send Clusterhead(u)
  $CH(u) \leftarrow u, CHweight(u) \leftarrow weight(u)$
  $u$ enters CH state.
else
  while true
    receive Clusterhead(v)
    send Join(u,v)
    $CH(u) \leftarrow v, CHweight(u) \leftarrow weight(v)$
    $u$ enters Member state.
  end
end
```

Table 4.4: Algorithm: SelectClusterhead

```
(selectCH, selectCHweight) \leftarrow (u, weight(u))
for each $x \in N(u)$
  if $x$ is in Idle state and weight($x$) < selectCHweight
    (selectCH, selectCHweight) \leftarrow (x, weight(x))
  end
end
return (selectCH)
```

enter to Clusterhead or Member state. Table 4.3 explains the procedure employed in Idle node.

Note that only the node weight heuristic is employed in Idle state. As introduced in Chapter 2, the node heuristic and the link heuristic are two separated heuristics originally. The pure link weight heuristic, such as $(\alpha, t)$-cluster framework, don’t elect clusterheads for clusters, which leads a loose cluster structure. The addition and leave of nodes cannot be announced in time. The general link heuristic is only suitable for quasi-static networks. Thus, we only utilize the link weight heuristic between clusterheads or between clusterheads and other nodes, which protects the effectiveness of our clustering
Table 4.5: Algorithm: Clusterhead State for node $u$

<table>
<thead>
<tr>
<th>Condition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{CHList}(u) \neq \emptyset \text{ and } \text{Size}(u) &lt; \text{MinimumSize}$</td>
<td>$\text{ChangeState} \leftarrow \text{false}$</td>
</tr>
<tr>
<td>for each $x \in \text{CHList}(u)$</td>
<td></td>
</tr>
<tr>
<td>if $\text{weight}(x) &lt; \text{weight}(u)$ and $\text{weight}(u, x) &lt; \text{link}_{\text{threshold}}$</td>
<td>$\text{ChangeState} \leftarrow \text{true}$</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>if $\text{ChangeState} = \text{true}$</td>
<td>$u$ enters TempCH state</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>if $\text{MemberList}(u) = \emptyset$</td>
<td>$u$ enters TempCH state</td>
</tr>
</tbody>
</table>

Algorithm:

**Clusterhead State**

A clusterhead maintains the existing cluster. Only when node $u$ is in CH state, it maintains MemberList($u$) to keep track of its current members. It is the agent between WWAN and WLAN network. A clusterhead is determined when a cluster forms. When the clusterhead changes, to some extent, the cluster also changes. Table 4.5 is operated when node $u$ becomes a clusterhead.

Only in two scenarios, the clusterhead may leave this state. First, if the clusterhead looses all its members, it enters to TempCH state. Second, if two or more clusterheads moving at similar patterns meet together, they seek to merge a new cluster. The clusterheads with higher weights should change their state to become a member of the new cluster. However, to avoid frequent state switch, two restrictions are set. One is that the weight of link between clusterheads should be smaller than the predefined threshold, $\text{link}_{\text{threshold}}$. Another one is that the size of the cluster should be smaller than a predefined constant: $\text{MinimumSize}$. It is nature that nodes in Clusterhead state utilizes...
Table 4.6: Algorithm: Member State for node $u$

<table>
<thead>
<tr>
<th>Algorithm: Member State for node $u$</th>
</tr>
</thead>
<tbody>
<tr>
<td>while true</td>
</tr>
<tr>
<td>if $CH(u) \notin N(u)$</td>
</tr>
<tr>
<td>u enters TempCH state</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>if $H(u) &lt; H_{\text{threshold}}$</td>
</tr>
<tr>
<td>for each $x \in CHList(u)$</td>
</tr>
<tr>
<td>if $weight(x) &lt; CHweight(u)$ and $weight(u, x) &lt; link_{\text{threshold}}$</td>
</tr>
<tr>
<td>u enters TempCH state</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>end</td>
</tr>
</tbody>
</table>

$	au_{ij}^{local}$ and $\tau_{i}^{local}$ to weigh whether clusters are similar enough. However, the minimum size indicates the different roles between TempCH and Clusterhead. When a node becomes a clusterhead, it should take the cluster stability into account. The limitation of size avoids the frequent state changes causing by one clusterhead giving up its responsibility. The limitation on splitting clusters makes our algorithm different from DMAC and MDMAC.

**Member State**

When a node enters Member state, it needs to keep connections with its clusterhead. If the clusterhead quits Clusterhead state, the cluster is disappeared. Therefore, members evaluate the connectivity with clusterhead and detect nodes in their vicinity. When a more robust clusterhead appears, it might change its clusterhead to avoid a potential disappearance of the current clusterhead. A parameter named the satisfaction factor is introduced to quantify the connectivity between members and clusterhead.

\[
H(u, t) = \alpha H(u, t-1) + (1 - \alpha) I(u, t)
\]  

(4.1)

Denoting as $H$, Equation 4.1 defines the calculation of $H(u, t)$, where $u$ represents a node in Member state, and $t$ represents the time. $I(u, t)$ is an indicator function that $\tau_{u,CH(u)}$ is below a threshold at time $t$, and we have:
\[ I(u, t) = \begin{cases} 
1, & \tau_{u,CH(u)} < \text{link}_\text{threshold}; \\
0, & \tau_{u,CH(u)} \geq \text{link}_\text{threshold}; 
\end{cases} \]

where \text{link}_\text{threshold} is a designed parameter for required link robustness.

Only when \( H(u, t) \) is smaller than a predefined threshold, the procedure shown in Table 4.6 is activated. And the following conditions are similar to conditions shown in Table 4.2. If node \( u \) can find a node in CH state whose node weight and link weight are both better than its current clusterhead, it means that some cluster may be more suitable for the node. The satisfaction factor separates nodes in TempCH state and in Member state. When a node joins a cluster, it cannot change to a stronger clusterhead immediately, even the clusterhead is better. It is a tradeoff between effectiveness and stability. To keep the effectiveness of the clustering algorithm, the cluster switch should be discouraged, while the cluster switch guarantees the stability of the cluster structure. The satisfaction factor provides a method to balancing both sides.

It is interesting to find that nodes in Clusterhead state and Member state both have common places as nodes in TempCH state except that some restrictions are added on the basis of their different states. The common place is that three states utilize the link weight and the node weight heuristic, while their different responsibilities deciding by there different characters lead extra conditions in the cluster maintenance phase. Unfortunately, Idle state ignores the link weight heuristic, because it is relatively difficult to employ both heuristics in the cluster formation phase.

### 4.2 Performance Evaluation

All simulation results for the clustering algorithm are presented in this section. To evaluate the performance of our algorithm, we validate it against MDMAC. For MDMAC, node degree is selected as the clustering metric for nodes, and \( LET \) is selected as the
metric for links, which are employed by Grzegorz et.al. [32]. Metrics measuring clustering performance include: average number of clusters, average cluster size, average number of cluster state changes, average clusterhead lifetime, and average member lifetime.

4.2.1 Simulation Setup

Each simulation lasted 1000 s, while only the data collected from the last 500 s were used in clustering algorithms. Employing WLAN technology, the transmission range was set to 250 meters. Hello messages were sent every 1 s. The vehicular traffic is generated by a traffic simulator, SUMO. [15]

We design a rectangular scenario to evaluate the clustering performance. The scenario employs a rectangular 3-lane road whose size is 2500 × 500 m. The scenario is unidirectional. Considering that relative velocity is utilized to calculate $\tau_{local}^i$, the direction does not affect the clustering result much.

The rectangular scenario is tested in a low-velocity case and a high-velocity case to collect reasonable data. In the low-velocity case, the average velocity is 16 m/s; and the range of velocity is from 12 m/s to 20 m/s. In the high-velocity case, the minimum velocity is 15 m/s, and the maximum velocity is 35 m/s. The average velocity is 25 m/s.

4.2.2 Design Parameters

In this part, we study how parameters affect each other and the clustering performance. The link threshold, $\tau_{link}$, decides how robust the links are between clusterheads and their members. $S_{ij}$ is the stability factor scaling the value of weight matrix; and $\delta$ decides the rising rate of $S_{ij}$. $TTL$ decides the size of the local network topology. To find the optimal parameter for the rectangular vehicular scenario, different values of these parameters are tested in the simulation, and the changes of performance metrics are compared. Some examples are listed in the following.

As we mentioned, $\delta$ decides the rising rate of $S_{ij}$; and the calculation of $LET$ is
based on an assumption that the velocities would not change. Figure 3.1 shows us how $\delta$ adjusts the scale of $S_{ij}$. To further explore the relationship between $\delta$ and the clustering performance, we assume $\tau_{\text{threshold}}^{\text{link}} = 0.1$ and swept $\delta$ over 30, 50, 100, 200, and 500.

Figure 4.2 describes a sample distribution of the clusterhead lifetime when $TTL = 2$, $\tau_{\text{threshold}}^{\text{link}} = 0.1$. When $\delta$ is less than 100, the differences of the lifetime distribution is trivial, which is also shown in Figure 4.3. When $\delta = 500$, the nodes represent corresponding values of lifetime are concentrated in the range of $[20, 200]$. Generally speaking, the smaller $\delta$ is, the more even the distribution of the clusterhead lifetime scatters. This phenomenon proves that $\delta$ do impact the clustering performance.

Figure 4.3 lists the average lifetime of clusterhead and member. It reveals more details about the impact of $\delta$ based on the average lifetime. Different values of $TTL$ are employed in this case.

The average lifetime is influenced by $\delta$, or we can say, $S_{ij}$. For the clusterhead lifetime, we notice that no matter how $TTL$ varies, the clusterhead lifetime is changing with the
increase of $\delta$. The overall tendency of the clusterhead lifetime is identical: The value of average clusterhead lifetime fluctuates in a relative small range when $\delta \leq 100$; and when $\delta \geq 200$, the performance of average lifetime doesn’t show any improvement with the increase of $\delta$, but also becomes worse.

![Graphs showing the average lifetime of clusterheads and members for different TTL values](image)

**Figure 4.3:** The average lifetime of clusterheads and members

We’d like to explain why the clusterhead lifetime changes in this way. The value of $\delta$ impacts the value of $S_{ij}$, which is the weight matrix in our clustering algorithm. It determines the extent of the network topology acting on $\tau_{ij}$. The value of $\delta$ decides the effect of the link-weight heuristic. When $\delta = 30, 50, 100$, the clusterhead lifetime has some slight mutation. The characteristic of $S_{ij}$ determines that the clustering algorithm is not too sensitive to the change of $S_{ij}$ when $\delta$ is small. Figure 3.1 shows us that $S_{ij}$ doesn’t converge to 1 when $LET \geq 1000$ with $\delta = 500$. It is unreasonable to assume that a vehicle keeps the same velocity within 1000s in the real scenario, i.e. the scaling
effect of $S_{ij}$ is weaken with a large $\delta$. And $\tau_{theshold}^{link}$ cannot detect the topology changes effectively. The sharp decrease of the clusterhead lifetime is caused by this reason.

The average member lifetime shows the similar tendency. The value of average member lifetime is much smaller than that of the clusterhead lifetime. It is natural since we allow that members switch to a better clusterhead to maintain the robustness of the cluster structure.

$\tau_{theshold}^{link}$ is another important parameter for the link weight heuristic. Figure 4.4 and Figure 4.5 show the state change times, i.e. the accumulative number of state changes during the simulation time, and the average lifetime with different $\tau_{theshold}^{link}$. It is not difficult to identify that the range of optimal $\tau_{theshold}^{link}$ is from 2 to 4. In the following case, we select $\tau_{theshold}^{link} = 2$.

We can see that the average lifetime is impacted by $TTL$ too. In Figure 4.3, the average lifetime doesn’t show a significant improvement when $TTL \geq 3$. Figure 4.6, 4.7, and 4.8 compare the average number of clusters, the average clusterhead change times,
Figure 4.5: The average lifetime with $\delta = 100$ and $TTL = 2$.

and the average cluster state change times from $TTL = 1$ to 4. To make the curve more clearly, we smooth the number of clusters and record the average number of clusters in Figure 4.6. The average number of clusters at a time point, $T_1$, is the average value of the cluster number from time period $T_1 - T$ to $T_1$, where $T$ is the smoothing window. We set $T = 10$ in our simulation.

Figure 4.7 and 4.8 show the average change times per node. It is nature that the state change frequency is decreased with the augment of the local network. We notice that the average member changes per node (Figure 4.8) doesn’t show a significant improvement from $TTL = 1$ to $TTL = 4$ compared with the average clusterhead changes in Figure 4.7. The reason is that the switch of clusterhead takes $\tau_{ij}^{local}$ into account. The raise of $TTL$ influences the selection of clusterhead directly, when more nodes are included in the local network. However, the member state switches usually consider $\tau_{ij}^{local}$ between the member and the clusterhead. $\tau_{ij}^{local}$ represents the end-to-end network criticality and evaluates the robustness of a path. It is not so sensitive to the change of $TTL$. Hence,
we define that the range of $TTL$ is from 1 to 3 for the simulation.

These figures prove that the clustering performance is similar when $TTL \geq 3$. In Chapter 3, our clustering metric is interpreted as a resistor in a resistance network. If we put this theory in the local network topology, it can be explained as follows: the increment of $TTL$ means that more nodes are considered in a specific node’s local network topology. Definitely, the number of possible paths between a pair of nodes is increased; and the addition of one path can be viewed as introducing a resistor as a parallel connection, which leads to the decrease of the total resistance. However, the decreasing rate of the total resistance should be reduced when more and more paths join the local calculation. The principle is proved in Figure 4.6, 4.7 and 4.8 by growing the value of $TTL$ from 1 to 4.
Figure 4.7: The average clusterhead changes per node, $\delta = 100$ and $\tau_{\text{link}} = 2$

Figure 4.8: The average cluster state changes per node, $\delta = 100$ and $\tau_{\text{threshold}} = 2$
4.2.3 Low-Velocity Scenario

In this section, our clustering algorithm is compared with MDMAC in a low-velocity scenario. As we described, the average velocity is 16\text{m/s} in the scenario. We use CCA (Criticality Clustering Algorithm) to mark our algorithm, since it is based on network criticality. CCA selects $\delta = 100$ and $\tau_{\text{link \_threshold}} = 2$ as clustering parameters. MDMAC uses the highest-degree as the clustering metric.

![Graph showing the average number of clusters with $\delta = 100$, $\tau_{\text{link \_threshold}} = 2$](image)

Figure 4.9: The average number of clusters with $\delta = 100$, $\tau_{\text{link \_threshold}} = 2$

Figure 4.9 compares the average number of clusters between CCA and MDMAC. The changes of average cluster number are close when $TTL = 1$. When $TTL = 2$, MDMAC represents a larger variation compared with MDMAC with $TTL = 1$. The larger variation is caused by the clustering metric of MDMAC. The highest-degree is based on the 1-hop local network. $LET$ is the metric for links in MDMAC. The highest-degree and LET cannot control nodes which indirectly connect to the clusterhead. They are not accurate metrics for the multiple hops cluster structure. At the other hand, CCA indicates a stable cluster structure by the smooth curve when $TTL = 2$. The slope of its
corresponding curve is relative stable in a long period. This is because $\tau_{\text{link}}^{local}$ estimated that the global network based on its local network. The multi-hop local network provides more information and $\tau_{\text{link}}^{local}$ evaluates the network robustness more accurately. This is an interesting property for multi-hop wireless networks.

Figure 4.10 shows the average cluster size of CCA and MDMAC. Both algorithms represent a fluctuating average cluster size. Generally speaking, it is hard to tell which algorithm is better in the low-velocity scenario only depending on the average cluster number and size. They have close performance when $TTL = 1$. CCA and MDMAC have a similar variation extend during the simulation period. However, CCA shows a more smooth curve when $TTL = 2$. The average number of clusters in CCA keeps a relative stable slope during a longer time period compared to that performance in MDMAC. We compare the performance again in the high-velocity scenario, and more information is provided in the following section. Figure 4.9 and Figure 4.10 don’t display a distinct advantage of CCA in the low-velocity scenario.
scenario, but provide promising characteristics.

Figure 4.11: The clusterhead change times per node with $\delta = 100$, $\tau_{\text{link}}^{\text{threshold}} = 2$

Figure 4.11 compares the average clusterhead change when $TTL = 1$ and $2$. It is clear that this performance of CCA outperforms that of MDMAC. Note that the average clusterhead change of MDMAC varies a lot under different values of $TTL$. However, our algorithm doesn’t show such a big difference. When $TTL = 2$, the performance only improves a little. One reason of the stable clusterhead switch is that CCA separates TempCH state from CH state to avoid the formation of smaller clusters, and another reason is network criticality evaluating the point-to-point robustness of links and taking the mean value as the robustness of nodes. Definitely, CCA is not as sensitive to the change of $TTL$ as MDMAC.

Figure 4.12 reveals the average state of member switches. It has the same tendency as the clusterhead change times. The total clusterhead change times are above 4 and 7 with $TTL = 1$ and $2$ in MDMAC, and the member changes are 10 and 12. At the same time, CCA’s clusterhead change times are both under 1 and the member changes
Figure 4.12: The member change times per node with $\delta = 100$, $\tau_{\text{link}}^{\text{threshold}} = 2$

are both around 2. CCA reduces the state change frequency significantly. The state change performance proves that CCA provides more robust clusters than MDMAC. In comparison to the fluctuated slopes of MDMAC, we believe CCA is a better clustering algorithm for mobile wireless networks, especially for highly mobile networks.

Figure 4.13 shows the average lifetime of clusterheads. $TTL$ is varied from 1 to 3. Considering nodes in $TempCH$ are clusterheads without members, we exclude such clusterheads in MDMAC. There is no doubt that the average lifetime of CCA is much longer than that of MDMAC. The advantage in the average lifetime supports our declaration that criticality is a more suitable metric to quantify the robustness of network connectivity.

### 4.2.4 High-Velocity Scenario

The high velocity case explains more advantages about our algorithm, CCA. In the high-velocity scenario, the average velocity of vehicles is 25m/s, which is much higher than the velocity in the previous scenario. In this section, we not only discuss the clustering
performance in the high velocity case, but also compare the clustering performance with the performance in the low-velocity scenario.

From Figure 4.14, we observe the average number of clusters of CCA and MDMAC. Unlike the performance in the low-velocity scenario, it is easy to see that the slopes of these curves in CCA are much more stable than these in MDMAC. Under different velocities, the total number of clusters doesn’t vary a lot. However, the variation in MDMAC is severer compared with the variation in CCA. As we mentioned in Section 4.2.3, the high-velocity case reveals an important property of CCA: it take a robust metric for the dynamic wireless network, and adjust the cluster structure based on the robustness of the network.

Figure 4.15 represents the average cluster size in the high-velocity scenario. The average cluster size is relatively stable in CCA. Again, we notice that CCA tends to form clusters with small sizes, but the average size is still above 4 in this case. 

The average clusterhead change times, and average member change times are dis-
Figure 4.14: The average number of clusters with $\delta = 100$, $\tau_{\text{threshold}}^{\text{link}} = 2$

Figure 4.15: The average cluster size with $\delta = 100$, $\tau_{\text{threshold}}^{\text{link}} = 2$
Figure 4.16: The clusterhead change times per node with $\delta = 100$, $\tau_{\text{link}}^{\text{threshold}} = 2$

Figure 4.17: The member change times per node with $\delta = 100$, $\tau_{\text{threshold}}^{\text{link}} = 2$
played in Figure 4.16, and Figure 4.17 respectively. Figures show that the cluster switch times of CCA are still much less than those of MDMAC. CCA and MDMAC utilize the same clustering algorithm framework, which is a combination of the link weight heuristic and the node weight heuristic. Hence, the clustering metrics lead to the main difference. If we compare these figures with Figure 4.11 and Figure 4.12, an interesting phenomenon is that the state switch times of MDMAC are reduced in the high-velocity scenario when $TTL = 1$, and increased when $TTL = 2$. Except the setting of the link threshold, the phenomenon should be caused by MDMAC’s clustering metrics: the highest degree and LET. At the same time, the switch times of CCA increased when $TTL = 1$ and 2, this tendency denotes that CCA is a robust clustering algorithm to the fluctuating velocity, i.e. the fluctuating network topology.

The average clusterhead and member lifetime are impacted by the velocity change. In Figure 4.18, MDMAC does not show obvious varies under different $TTL$s. Considering the average lifetime of MDMAC in Figure 4.13, the augment of the hops cannot
introduce any improvement for MDMAC in the high-velocity scenario. Although the average lifetime is impacted by the high speed, CCA still shows the improvement in Figure 4.18, especially when the number of hop rises from $TTL = 1$ to 2. We believe that the properties of the network criticality lead to the improvement of the lifetime performance.
Chapter 5

Conclusion

In this thesis, we proposed a clustering algorithm for VANETs based on the notion of localized network criticality. To the best of our knowledge this is the first work in the area of vehicular network which employs a localized robust graph metric to guide the process of building clusters.

Considering the highly dynamic nature of the vehicular network, we emphasize the robustness of the resulting clustering structure. Network criticality is a global metric that quantifies the robustness of a network topology against the variable environment including the network topology. To implement its usage in vehicular networks, we derived two new concepts: point-to-point criticality and node criticality. Point-to-point network criticality measures the resistance between a pair of nodes in the network. And node criticality evaluates the average resistance for one node. At the same time, this thesis defines distributed versions of criticality, $\tau_{ij}^{local}$ and $\tau_{i}^{local}$. The distributed version allows us to utilize criticality for distributed clustering algorithms. It is the first time that $\tau_{ij}^{local}$ and $\tau_{i}^{local}$ are used in wireless networks.

In the clustering method, the simple-k-medoids clustering algorithm is utilized to test the performance of point-to-point network criticality. Considering the difficulty of employing a centralized clustering algorithm in dynamic networks, we don’t go further
on this part. We plan to explore the effectiveness of the centralized clustering algorithm in the future work.

For the distributed case, we propose a clustering algorithm, CCA. We localize the notion of point-to-point network criticality and discuss its usage as a metric for clustering algorithms. Few other works employ such a localized graph metric for clustering a network. The clustering framework of CCA is proposed on the basis of previous clustering algorithms for MANETs. We compare the performance of CCA to MDMAC, a modified version of DMAC. The simulation evaluates the two algorithms in different scenarios. CCA significantly improves the lifetime of clusters and the state change times. It is more suitable for multi-hop mobile wireless networks. The simulation result proves that CCA shows a more robust cluster structure.
Bibliography


