December 2016

Complex Network Analysis and the Applications in Vehicle Delay-Tolerant Networks

Bo Wu
Clemson University, bwu2@clemson.edu

Measurement and Routing Algorithm Design in Vehicle Delay-Tolerant Networks

---

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Computer Engineering

---

by
Bo Wu
December 2016

Accepted by:
Dr. Haiying Shen, Committee Chair
Dr. Adam Hoover
Dr. Ilya Safro
Dr. Yongqiang Wang

Abstract

Vehicle Delay Tolerant Networks (VDTNs) is one type of Delay Tolerant Networks (DTNs), where vehicles equipped with transmission capabilities are interconnected to form Vehicle NETworks (VNETs). Some applications and services on the top of VDTNs have raised a lot of attention, especially by providing information about weather conditions, road safety, traffic jams, speed limit, and even video streamings without the need of infrastructures. However, due to features such as high vehicle mobility, dynamic scenarios, sparsity of vehicles, short contact durations, disruption and intermittent connectivity and strict requirements for latency, many VDTNs do not present satisfactory performance, because no path exists between a source and its target. To solve this problem, in this dissertation, we propose three routing methods as follows.

Our first VDTN system focuses on the multi-copy routing in VDTNs. Multi-copy routing can balance the network congestion caused by broadcasting and the efficiency limitation in single-copy routing. However, the different copies of each packet search the destination node independently in current multi-copy routing algorithms, which leads to a low utilization of copies since they may search through the same path repeatedly without cooperation. To solve this problem, we propose a fractal Social community based efficient multi-coPy routing in VDTNs, namely SPread. First, we measure social network features in VNETs. Then, by taking advantage of weak ties and fractal structure feature of the community in VNETs, SPread carefully scatters different copies of each packet to different communities that are close to the destination community, thus ensuring that different copies search the destination community through different weak ties. For the routing of each copy, current routing algorithms either fail to exploit reachability information of nodes to different nodes (centrality based methods) or only use single-hop reachability information (community based methods), e.g., similarity and probability. Here, the reachability of node i to a destination j (a community or a node) means the possibility that a packet can reach j through i. In order to overcome

above drawbacks, inspired by the personalized PageRank algorithm, we design new algorithms for calculating multi-hop reachability of vehicles to different communities and vehicles dynamically. Therefore, the routing efficiency of each copy can be enhanced.

However, in SPread, we only consider the VNETs as complex networks and fail to use the unique location information to improve the routing performance. We believe that the complex network knowledge should be combined with special features of various networks themselves in order to better benefit real applications. Therefore, we further explore the possibility to improve the performance of VDTN system by taking advantage of the special features of VNETs. We first analyze vehicle network traces and observe that i) each vehicle has only a few active sub-areas that it frequently visits, and ii) two frequently encountered vehicles usually encounter each other in their active sub-areas. We then propose Active Area based Routing method (AAR) which consists of two steps based on the two observations correspondingly. AAR first distributes a packet copy to each active sub-area of the target vehicle using a traffic-considered shortest path spreading algorithm, and then in each sub-area, each packet carrier tries to forward the packet to a vehicle that has high encounter frequency with the target vehicle. Furthermore, we propose a Distributed AAR (DAAR) to improve the performance of AAR.

Finally, we try to combine different routing algorithms together and propose a DIstributed Adaptive-Learning routing method for VDTNs, namely DIAL, by taking advantages of the human beings communication feature that most interactions are generated by pairs of people who interacted often previously. DIAL consists of two components: the information fusion based routing method and the adaptive-learning framework. The information fusion based routing method enables DIAL to improve the routing performance by sharing and fusing multiple information without centralized infrastructures. Furthermore, based on the information shared by information fusion based routing method, the adaptive-learning framework enables DIAL to design personalized routing strategies for different vehicle pairs without centralized infrastructures. Therefore, DIAL can not only share and fuse multiple information of each vehicle without centralized infrastructures, but also design each vehicle pair with personalized routing strategy.

Extensive trace-driven simulation demonstrates the high efficiency of SPread in comparison with state-of-the-art routing algorithms in DTNs. Furthermore, AAR produces higher success rates and shorter delay in comparison with the state-of-the-art routing algorithms and SPread. Also, DAAR has a higher success rate and a lower average delay compared with AAR since information

of dynamic active sub-areas tends to be updated from time to time, while the information of static active sub-areas may be outdated due to the change of vehicles' behaviors. Finally, DIAL has better routing success rate, shorter average delays and the load balance function in comparison with state-of-the-art routing methods which include SPread and AAR.

In the future, we will explore the possibility of routing in VDTNs with the help of road side units. Without a centralized infrastructure, the routing performance cannot be guaranteed. Thus, we will also try to explore the possibility to build a hybrid network among vehicles incorporating an infrastructure, which can improve the performance and at the same time decrease the cost by only using a centralized infrastructure.

Acknowledgments

I would like to thank many people who helped me during my Ph.D. study at Clemson University. I would like to first thank my advisor Dr. Haying Shen, who not only shed lights on my study, but also help me solve many difficulties in life. Without her help, I cannot finish my Ph.D. study, not even to mention a productive research. Her passion for scientific research and hard working will always set an example to me, which will benefit me for my whole life.

I would also like to thank my committee members: Dr. Adam Hoover, Dr. Ilya Safro, Dr. Yongqiang Wang and Dr. Richard R. Brooks. Their professional suggestions for my research and valuable comments for my dissertation helped me finish my research. I have learned a lot from their broad knowledge and deep outstanding of research, which helps me pursue my future career.

My life at Clemson University is full of happiness and friendship. I should thank all labmates in the Pervasive Communication Lab for their help and hard working together to accomplish each research projects. Without their accompanies, we research cannot be fruitful and my life at Clemson will be grey.

To the end, the most important is that I am deeply grateful to my parents, whose endless love and support always be the harbors of my heart and soul. My achievement belongs to them.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem Statement

Vehicle Delay Tolerant Networks (VDTNs) is one type of Delay Tolerant Networks (DTNs), where vehicles equipped with transmission capabilities are interconnected to form Vehicle NETworks (VNETs). Some applications and services on the top of VDTNs have raised a lot of attention, especially by providing information about weather conditions, road safety, traffic jams, speed limit, and even video streamings without the need of infrastructures. However, due to features such as high vehicle mobility, dynamic scenarios, sparsity of vehicles, short contact durations, disruption and intermittent connectivity and strict requirements for latency, many VDTNs do not present satisfactory performance, because no path exists between a source and its target. The current routing algorithms include multi-copy algorithms [21, 32, 65, 68] and single-copy routing algorithms [10, 14, 18, 33, 34, 40, 45, 47, 48, 53, 74, 76]. Multi-copy algorithms can balance the network congestion of broadcasting and single-copy routing efficiency. The current basic single-copy routing algorithms can be divided to four categories: probabilistic routing [10, 47], centrality based routing [33, 53], community based routing [18, 34, 45, 76] and location based routing algorithms [14, 40, 48, 74]. The probabilistic routing algorithms [10,47] forward packages gradually through intermediate nodes with higher probability to reach the target node. The community based routing algorithms [18, 34, 45, 76] consider the reachability of nodes to different communities rather than the reachability to nodes. The centrality based routing algorithms [33, 53] applied different centrality criteria such as degree and betweenness for improving routing efficiency in which the multi-hop information can be considered.

Location based routing algorithms [14, 40, 48, 74] predict the future locations of vehicles, find the shortest path from the source vehicle to the target vehicle, and select the vehicles with trajectories on the shortest path as relay vehicles. All the aforementioned routing algorithms have their own problems as described below.

1.1.1 Problems in Multi-copy, Probabilistic, Centrality and Community based Routings in VDTNs

Multi-copy algorithms try to balance the network congestion of broadcasting and single-copy routing efficiency. However, the routing efficiency is influenced by those infrequent contacts between vehicles in VDTNs, called weak ties [27]. Previous study shows that weak ties play an important role in information spreading [27]. In current multi-copy strategies [21, 32, 65, 68], mobilities of different copies of a package are independent from each other. Therefore, different copies may be allocated nearby and search through the same weak ties, which decreases the efficiency of multi-copies.

In the probabilistic routing algorithms, packages are gradually forwarded through intermediate nodes with higher probability to reach the target node. However, VDTNs usually consist of thousands of vehicles (nodes) and sparse topologies, leading to little chance to meet a suitable relay node. However, the community based routing algorithms try to overcome the drawback of probabilistic routing algorithms by considering the reachability of nodes to different communities rather than the reachability to nodes. The community based routing algorithms can only consider the one hop information [45, 76] (such as encounter frequency and similarity) and lacks the capability to count multi-hop information of the reachability of nodes to different communities. For the centrality based routing algorithms, previous studies applied different centrality criteria such as degree and betweenness for improving routing efficiency in which the multi-hop information can be considered. However, the centrality criteria measure the importance of the nodes, but not the reachability of nodes to different nodes.

Furthermore, previous routing algorithms more or less take the advantage of the social features of human mobility, such as important nodes and communities. However, for VNETs, it is a question that whether the mobility of vehicles, especially the taxies, have the similar features as human mobility since taxies move randomly which are highly dependent on random customers' demands.

1.1.2 Problems in Location based Routings in VDTNs

The main problem of location based routing algorithms [14,40,48,74] is that these algorithms require highly accurate prediction so that relay vehicles and the packet carrier will be close to each other in a certain short distance (e.g., less than 100 meters). However, it is difficult to achieve accurate prediction because vehicles have high mobility and vehicle trajectories are greatly influenced by many random factors such as the traffic and speed of vehicles. Also, since the shortest path is determined without considering the traffic, there may be few vehicles on the path. Therefore, if the selected relay vehicle is missed due to low prediction accuracy, it is difficult to find other relay candidates, which leads to low routing efficiency.

1.1.3 Combination of Different Routing Algorithms

There are many different kinds of routing algorithms. However, we cannot say one routing algorithm is totally better than another since in different scenario, the same routing algorithm may have different performance. For example, if two vehicles are in the same community, community based routing can be more efficient than centrality and location based routing methods. Therefore, we hope to adopt different routing algorithms under different scenarios. However, when it comes to a specific vehicle pair, such correlations can be influenced by many unknown factors, which makes it difficult to predict the performances of different routing methods.

1.2 Research Approach

We solve the aforementioned problems by proposing three routing algorithms as follows.

1.2.1 SPread: Exploiting Fractal Social Community For Efficient Multi-copy Routing in VDTNs

To deal with the problems we mentioned in Section 1.1.1, first, we analyze real Vehicle NETwork (VNET) *Roma* [49] and *SanF* [58] traces. We find: (i) there are few very important vehicles which have high degrees and PageRank values where PageRank is the most popular algorithm for ranking the importance of pages in World Wide Web; (ii) VNETs consist of communities which are connected by weak ties; (iii) the communities are with fractal structure.

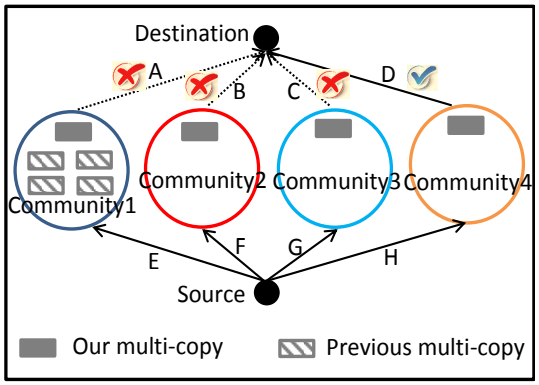


Figure 1.1: Multi-copy routing algorithms comparison

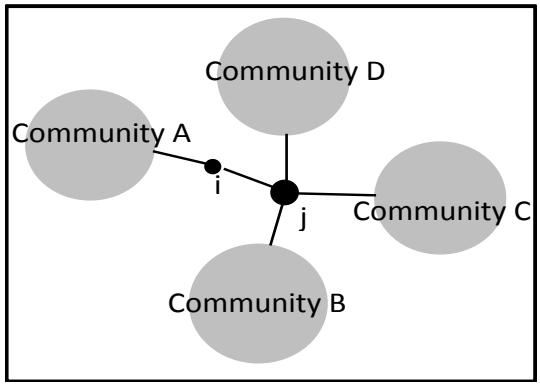


Figure 1.2: An example of personalized important nodes

Then, in order to improve the efficiency of current multi-copy strategies, we design the weak tie based multi-copy strategy by taking the advantage of fractal structure of community. In the weak tie based multi-copy strategy, the multi-copies of each package are scattered to different communities which are close to the target community through weak ties. Then, the multi-copies can search the target node through different weak ties. Therefore, we can improve the multi-copy efficiency since there is a high probability at least one weak tie is connected to the target community during the routing time. As shown in Figure 1.1 for example, in previous multi-copy strategy, different copies of one package are all allocated to community 1 and are all searching path A, while path A is disconnected to the target at the routing time (which is quite normal in DTNs). In our weak tie based multi-copy strategy, we scatter the copies to different communities and therefore, multi-copies can search the target node by different weak ties (A, B, C, D) at the same time and meet a connected path D, which improves the success rate of the routing.

For the routing of each single-copy, in order to enhance the routing efficiency and overcome the drawbacks in current sing-copy routing algorithm, we design personalized CommunityRank and VehicleRank algorithms for calculating multi-hop reachability of vehicles to communities and vehicles, which are inspired by personalized PageRank algorithm [31]. By taking the advantage of the community structure of VNETs, we divide the routing to two phases: inter-community and intra-community. In the inter-community phase, packages are trying to reach the target community by a table recording the multi-hop reachability of nodes to different communities. In the intra-community phase, packages are trying to reach the target node by a table recording the multi-hop reachability of nodes to nodes in the same community. The multi-hop reachability of vehicles can enhance the single-copy routing efficiency. For example as shown in Figure 1.2, node a has a higher

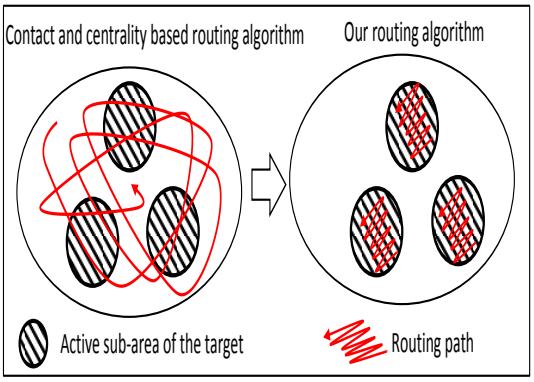


Figure 1.3: Current routing algorithm vs. AAR.

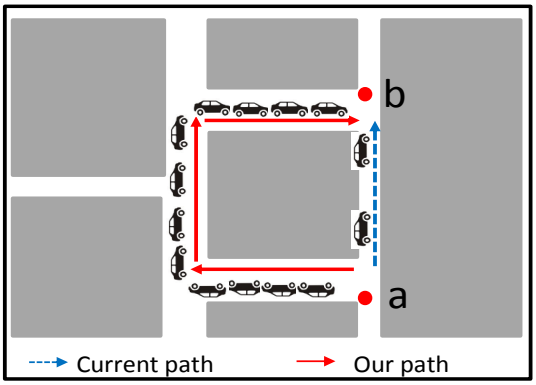


Figure 1.4: An example of the traffic-considered shortest path spreading algorithm.

rank for routing to community A, while node b has a higher rank for routing to communities B and C. In previous algorithm based on centrality, all the packages will choose node b. While in our algorithm, the packages with a target node in community A will choose node a instead of node b to forward packages.

1.2.2 Exploiting Active Sub-areas for Multi-copy Routing in VDTNs

To deal with the problems we mentioned in Section 1.1.2, we first analyze real Vehicle NETwork (VNET) *Roma* [49] and *SanF* [58] traces and gain the following two observations: i) each vehicle has only a few active sub-areas in the entire VDTN area that it frequently visits, and ii) two frequently encountered vehicles usually have high probability to encounter each other in their active sub-areas, while have very low probability to encounter each other in the rest area on the entire VDTN area. We then propose Active Area based Routing method (AAR) which consists of two phases based on the two observations correspondingly.

As shown in Figure 1.3, unlike the contact and centrality based routing algorithms that search the target vehicle in the entire VDTN area, AAR constrains the searching areas to the active sub-areas of the target vehicle, which greatly improves the routing efficiency. AAR first distributes a packet copy to each active sub-area of the target vehicle, and then in each sub-area, each packet carrier tries to forward the packet to a vehicle that has high encounter frequency with the target vehicle. Specifically, AAR consists of the following two algorithms for these two steps.

Traffic-considered shortest path spreading algorithm. It jointly considers traffic and path length in order to ensure there are many relay candidates in the identified short paths

to efficiently distribute multiple packet copies. Figure 1.4 shows an example of the basic idea of our traffic-considered shortest path spreading algorithm. Current location based routing algorithms relay the packet from road intersection a to b through the shortest path (i.e., the dotted line) but fail to consider whether there are enough relay vehicles in the path. If the shortest path only consists of small roads with less traffic, it leads to a long time for a packet to reach the target sub-area. In our spreading algorithm, the packet is routed along the circuitous path (i.e., the solid line) which consists of main roads that are full of traffic. Then, the packet can easily find next hop relay vehicle and reach b faster in spite of the longer length of the path.

Contact-based scanning algorithm. It restricts each packet copy in its responsible active sub-area to find relay vehicles with high encounter frequencies with the target vehicle. Specifically, the packet copy is forwarded to vehicles traveling in different road sections so that it can evenly scan the sub-area.

By avoiding searching the non-active sub-areas of the target vehicle as in the contact and centrality based routing algorithms, AAR greatly improve routing efficiency. Instead of pursuing the target vehicle as in the location based routing algorithms, each packet copy in an active sub-area of the target vehicle is relayed by vehicles with high encounter frequency with the target vehicle, thus bypassing the insufficiently accurate location prediction problem in location based routing algorithms.

1.2.3 DIAL: A Distributed Adaptive-Learning Routing Method in VDTNs

To deal with the problems mentioned in Section 1.1.3, we compare the performances of different routing methods in the analysis. We find two useful observations as follows: (i) The performances of different routing methods can be different on different vehicle pairs. (ii) It is true that there are some correlations between the routing performances of different methods and the features of vehicle pairs (e.g. geographic distance of the two vehicles and whether two vehicles are in the same community). Based on the observation (i), we hope to choose the routing method for each vehicle pair separately so that the routing performances on all the vehicle pairs can be optimal. However, based on the observation (ii), we find that it's difficult to choose the routing methods when it comes to a specific vehicle pair. Even if we can choose the routing methods based on features of vehicle pairs, centralized infrastructures are needed for sharing the necessary information among vehicles in order to calculate those features such as geographic distances and whether two

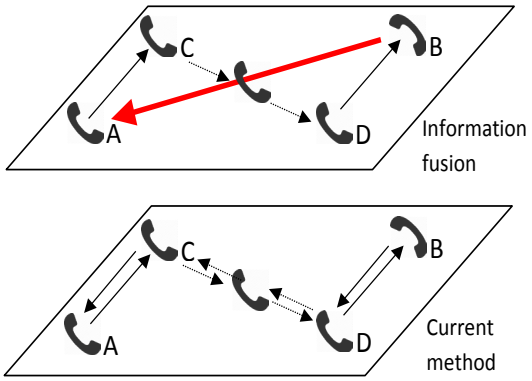


Figure 1.5: Current methods vs. information fusion based routing method.

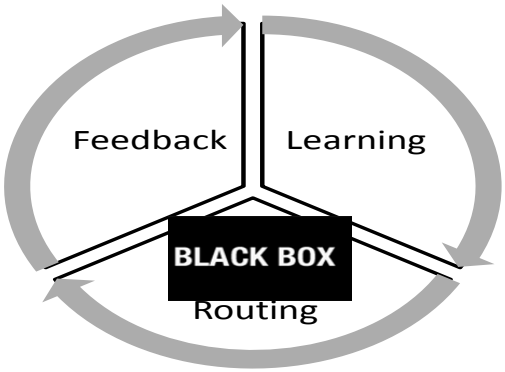


Figure 1.6: The adaptive-learning framework.

vehicles are in the same community. Fortunately, the routing of VDTNs is different from the general DTNs. General DTNs can consist of any kinds of moving objects, but VDTNs consist of vehicles which are driven by human beings. Hence, the communication on the top of VDTNs is actually the communication between human beings. Previous studies [51] show that most interactions are generated by pairs of people who interacted often previously. By taking advantages of this feature, we design a DIstributed Adaptive-Learning (DIAL) routing method which can not only share the important routing information of each vehicle without centralized infrastructures, but also design each vehicle pair with a personalized routing strategy. DIAL consists of two components: the information fusion based routing method and the adaptive-learning framework.

By taking advantages of the human beings communication feature we mentioned above, the information fusion based routing method enables DIAL to improve the routing performance by sharing and fusing multiple information without centralized infrastructures. In the information fusion based routing method, vehicle A attaches its personal information of frequently visited locations and frequently contact vehicles to the packet once vehicle A tries to deliver a packet to vehicle B. Then, based on centrality based method, the packet is delivered one relay vehicle by another relay vehicle. Lacking of the personal information (e.g. frequently visited locations and frequently contact vehicles) of target vehicle B, obviously, such kind of routing is inefficient. Once vehicle B gets all the personal information of vehicle A, vehicle B fuses all the personal information by setting different thresholds for adopting different routing methods (we introduce the detail of such a threshold in Section 5.2). Therefore, next time, vehicle B delivers the packet to vehicle A by choosing more efficient routing method. Since human beings tend to interact with others who they

interacted with before, the information fusion based routing method avoids the inefficient routing in the following communication between A and B and therefore, improves the routing performance without any centralized infrastructures. Figure 1.5 shows the difference between current method and the information fusion based routing methods.

With the help of the information fusion based routing method, we can share and fuse information of vehicles without centralized infrastructures. However, based on the observation (ii), we find that it's difficult to predict which routing method should be the best option for different vehicle pairs even though there are some correlations between the routing performances of different methods and the features of the vehicle pairs. Therefore, in order to design each vehicle pair with personalized routing strategy, we build an adaptive-learning framework. Instead of determining the thresholds for different methods statically, in the adaptive-learning framework, we consider the routing process as a black box and use feedback of success rates to determine the thresholds dynamically as shown in Figure 1.6. Similar as the information fusion based routing method, by taking advantages of the human beings communication feature, we can calculate the routing success rates of different routing methods which use different information. Then based on the feedback of the success rates, we can analyze the performances of different routing methods and adjust the routing strategies. For example, vehicle B frequently receives the packets sent from vehicle A. These packets can be delivered by contact based method, centrality based method or location based method. Once vehicle B sends packets to vehicle A, vehicle B sends the numbers of packets successfully delivered by different methods from vehicle A at the same time. Then vehicle A can calculate the success rates based on the numbers of packets successfully delivered by different methods and adjust thresholds for different routing methods accordingly to give more preference to the method that can lead to the highest success rate. The routing strategy can be self-adaptive in the adaptive-learning framework as shown in Figure 1.6. Therefore, we can design different vehicle pairs with different routing strategies and at the same time, the routing strategies can be continually improved according to the feedbacks of the routing performances.

1.3 Contributions

We summarize our contributions of the dissertation as follows:

1. Firstly, we study the complex network community application. We focus on the multi-copy

routing in Vehicle Delay Tolerant Networks (VDTNs). Multi-copy routing can balance the network congestion caused by broadcasting and the efficiency limitation in single-copy routing. However, the different copies of each packet search the destination node independently in current multi-copy routing algorithms, which leads to a low utilization of copies since they may search through the same path repeatedly without cooperation. To solve this problem, we propose a fractal Social community based efficient multi-coPy routing in VDTNs, namely SPread. First, we measure social network features in Vehicle NETworks (VNETs). Then, by taking advantage of weak ties and fractal structure feature of the community in VNETs, SPread carefully scatters different copies of each packet to different communities that are close to the destination community, thus ensuring that different copies search the destination community through different weak ties. For the routing of each copy, current routing algorithms either fail to exploit reachability information of nodes to different nodes (centrality based methods) or only use single-hop reachability information (community based methods), e.g., similarity and probability. Here, the reachability of node i to a destination j (a community or a node) means the possibility that a packet can reach j through i. In order to overcome above drawbacks, inspired by the personalized PageRank algorithm, we design new algorithms for calculating multi-hop reachability of vehicles to different communities and vehicles dynamically. Therefore, the routing efficiency of each copy can be enhanced. Finally, extensive trace-driven simulation demonstrates the high efficiency of SPread in comparison with state-of-the-art routing algorithms in DTNs.

2. Secondly, we find the following two observations: i) each vehicle has only a few active sub-areas in the entire VDTN area that it frequently visits, and ii) two frequently encountered vehicles usually have high probability to encounter each other in their active sub-areas, while have very low probability to encounter each other in the rest area on the entire VDTN area. We propose a traffic-considered shortest path spreading algorithm to spread different copies of a packet to different active sub-areas of the target vehicle efficiently. We propose a contact based scanning algorithm in each active sub-area of the target vehicle to relay the packet to the target vehicle. Furthermore, we propose an Advanced AAR (AAAR) by exploiting the spatio-temporal correlation of the visiting times of target vehicles on different road side units.

3. Finally, we further improve the routing efficiency of VDTNs by taking advantages of two fea-

tures: (i) Most interactions are generated by pairs of people who interacted often previously; (ii) The performances of different routing methods can be different on different vehicle pairs. We design the information fusion based routing method. The information fusion based routing method can distributedly share and fuse the vehicles' personal information in the routing process without the help of centralized infrastructures. Therefore, it is more practical and efficient than previous routing methods which are based on the help of centralized infrastructures. We design an adaptive-learning framework on the top of the information fusion based routing method which can design different routing strategies for different vehicle pairs for more efficient VDTN routing than the basic information fusion based routing method.

1.4 Dissertation Organization

The rest of this dissertation is structured as follows. Chapter 2 presents the related work. In Chapter 3, we propose SPread which is a multi-copy routing system in Vehicle Delay Tolerant Networks (VDTNs). In Chapter 4, by taking advantages of the unique features of VNETs, we propose AAR routing system which can further improve the routing efficiency. In Chapter 5, by taking advantages of the human beings communication feature that most interactions are generated by pairs of people who interacted often previously, we propose DIAL to improve the routing performance by sharing and fusing multiple information which include contact information, centrality information and location information. Finally, Chapter 6 concludes this dissertation with remarks on our future work.

Chapter 2

Related Work

Current routing algorithms in VDTNs can be classified to four categories: probabilistic [10, 47, 67, 77], centrality based [33, 53], community based routing [18, 34] and location based routing algorithms [14, 40, 48, 74].

In the category of probabilistic routing algorithms, PROPHET [47] simply selects vehicles with higher encounter frequency with target vehicles for relaying packets. PROPHET is improved by MaxProp [10] with the consideration of the successful deliveries history. R3 [67] considers not only the encounter frequency history, but also the history of delays among encounters to decrease the routing delay performance. Zhu *et al.* [77] found that two consecutive encounter opportunities drops exponentially and based on the observation, improved the prediction of encounter opportunity by Markov chain to design the routing algorithm in vehicle networks.

In the category of centrality based routing algorithms, PeopleRank [53] is inspired by the PageRank algorithm, which calculates the rank of vehicles and forwards packets to the vehicles with higher ranks. However, though vehicles with high centrality can encounter more vehicles, they may not have a high probability of encountering the target vehicle. Also, the main problem in both contact and centrality based routing algorithm is that packets may hardly encounter suitable relay vehicle due to the low encounter frequencies among vehicles in a large-scale VDTN.

In the category of community based routing algorithms [18, 34], SimBet [18] identifies some bridge nodes as relay nodes which can better connect the VNETs by centrality characteristics to relay packets. Instead of directly forwarding packets to target nodes, Bubble [34] clusters the nodes to different communities based on encounter history and still utilizes the bridge nodes to forward

packets to the destination community.

In the category of location based routing algorithms, GeOpps [40] directly obtains the future location of the target vehicle from GPS data and spreads packets to certain geographical locations for routing opportunities through shortest paths. GeoDTN [48] encodes historical geographical movement information in a vector to predict the possibility that two vehicles become neighbors. Wu *et al.* [74] exploited the correlation between location and time in vehicle mobility when they used trajectory history to predict the future location of the target vehicle in order to improve the prediction accuracy. Instead of predicting exact future location, DTN-FLOW [14] divides the map to different areas and predict the future visiting area of vehicles, which improves the routing performance since it is much easier to predict the future visiting areas than exact future locations. However, as indicated previously, the location based algorithms may lead to low routing efficiency due to insufficiently accurate location prediction due to traffic and vehicle speed variance.

A number of multi-copy routing algorithms also have been proposed. Spyropoulos *et al.* [65] introduced a "spray" family of routing schemes that directly replicate a few copies by source vehicle into the network and forward each copy independently toward the target vehicle. R3 [67] simply adopts the "spray" routing schemes based on its own single-copy routing. Bian *et al.* [32] proposed a scheme for controlling the number of copies per packet by adding an encounter counter for each packet carrier. If the counter reaches the threshold, then the packet will be discarded by the packet carrier. Uddin *et al.* [68] minimized the energy efficiency by studying how to control the number of copies in a disaster-response applications, where energy is a vital resource. However, in current multi-copy routing algorithms, different copies of each packet may search the same area on the entire VDTN area, which decreases routing efficiency. Our proposed AAR spreads different copies of each packet to different active sub-areas of the target vehicle, which greatly improves the routing efficiency.

Comparing to the previous methods, our proposed methods are novel for the following reasons: i) we consider the vehicle networks as complex networks, analyze the features and propose routing algorithms based on these features; ii) we observe that vehicles tend to be active in its own small active areas and then limit the routing in those areas; and iii) we propose an adaptive-learning method which can select different routing algorithms for different pairs of vehicles based on their unique features. These three algorithms significantly improve the performance of the previous routing algorithms.

Chapter 3

SPread: Exploiting Fractal Social Community For Efficient Multi-copy Routing in VDTNs

In this chapter, we focus on the multi-copy routing in Vehicle Delay Tolerant Networks (VDTNs). Multi-copy routing can balance the network congestion caused by broadcasting and the efficiency limitation in single-copy routing. However, the different copies of each packet search the destination node independently in current multi-copy routing algorithms, which leads to a low utilization of copies since they may search through the same path repeatedly without cooperation. To solve this problem, we propose a fractal Social community based efficient multi-coPy routing in VDTNs, namely SPread. First, we measure social network features in Vehicle NETworks (VNETs). Then, by taking advantage of weak ties and fractal structure feature of the community in VNETs, SPread carefully scatters different copies of each packet to different communities that are close to the destination community, thus ensuring that different copies search the destination community through different weak ties. For the routing of each copy, current routing algorithms either fail to exploit reachability information of nodes to different nodes (centrality based methods) or only use single-hop reachability information (community based methods), e.g., similarity and probability. Here, the reachability of node i to a destination j (a community or a node) means the possibility that a packet can reach j through i. In order to overcome above drawbacks, inspired by the personalized

PageRank algorithm, we design new algorithms for calculating multi-hop reachability of vehicles to different communities and vehicles dynamically. Therefore, the routing efficiency of each copy can be enhanced. Furthermore, we propose an Advanced SPread (denoted by ASPread) by exploiting the spatio-contact correlation of the community and considering the different sizes of communities. The spatio-contact correlation of the community means that the vehicles in the same community tend to meet each other in a certain small geographical area, which enables ASPread to forward a packet to vehicles that are more likely to meet the destination vehicle. Further, ASPread sends a larger number of copies of a packet to a community with a larger number of vehicles and vice versa. Finally, extensive trace-driven simulation demonstrates the high efficiency of SPread in comparison with state-of-the-art routing algorithms in DTNs. Also, ASPread generates higher success rate, lower average delay and average cost than SPread. The main contributions of this chapter are as follows:

1. We propose two densest subgraph discovering methods derived from the Goldberg's densets subgraph discovering algorithm.

2. We first measure the important nodes, community structure and fractal structure feature of the community in VNETs.

3. We design a weak tie based multi-copy routing algorithm to improve the multi-copy routing efficiency by taking advantage of the fractal structure feature of the community in VNETs.

4. We design personalized CommunityRank and VehicleRank algorithms to calculate the multi-hop reachability of vehicles to different communities and vehicles, which are inspired by the personalized PageRank algorithm. The new algorithms can enhance the single-copy routing efficiency by considering the multi-hop reachability of vehicles to different communities and vehicles.

5. We propose an Advanced SPread (ASPread) which improves the performance of the basic SPread routing algorithm.

3.1 An Improvement of Goldberg's Densest Subgraph Discovering Algorithm

In order to measure the features of VNET, we first study the community structure.

3.1.1 Previous Study of Community Discovering Algorithms

The studies of complex networks are developing with the coming observations of patterns of real-world networks. The psychologist Stanley Milgram conducted a series of experiments that indicated the average path length of peoples in human society is short [52] which is called *small world phenomenon*. Besides the small world phenomenon, evidences suggest that in most real-world networks, and particular social networks, nodes tend to create tightly knit groups characterized by a relatively high density of ties; this likelihood tends to be greater than the average probability of a tie randomly established between two nodes [70]. A *community* is a cohesive subset of nodes with denser inner links, relatively to the rest of the network [54]. It is discussed that a high clustering coefficient is the natural consequence of a community structure [57]. Studies have shown that the degree distribution of many real-world networks follow a power law [5]. Studies notice not only the degrees follow a power law, but also there is a scale-free distribution of communities [39, 44]. ER networks [22] are generated by picking nodes randomly and then connecting them by edges. ER networks exhibit a small average shortest path length which explained the small world phenomenon. The BA networks [5] proposed that structure emerges in network topologies as the result of two processes: *Growth* and *Preferential Attachment*. *Growth* means that there are continuing new nodes participate in the network; *Preferential Attachment* means that the probability of connecting to a node is proportional to the current degree of that node. The BA network is constructed as follows: starting from m_0 nodes, every time step a new node with m edges is added and each edge of the new node is attached to an existing node i with the probability proportional to the degree of the ith node. These two mechanisms result in power-law degree distribution with exponent $\gamma = 3$. The simplicity of the mechanism of BA graph make it an excellent model to explain the power law distribution of real-world graphs. However BA network has the deficiencies include the limited range of exponent value [19], the relation between node degree and age [37], the unavailable global information [30, 46] and so on. According to these deficiencies, PLRG model [61] is proposed to generate a degree power law with exponent in a range of $(0, + \propto)$. Fitness model [7] is proposed

that each node has a rank named fitness, and the preferential attachment is influenced by the rank. The fitness model can solve the deficiency of relation between degree and age of nodes. A step-by-step random walk network [30] is proposed that the links are created by random walk, but not preferential attachment to solve the deficiency of the unavailable global information in realities. The local-world evolving network [46] tried to solve the unavailable global information in realities by dividing the entire network to local-world evolving subnetworks. In local-world evolving network, each new coming nodes randomly select M nodes as their local-world, and they obey the preferential attachment rule in their local-world. Forest fire model [44] was proposed to explain a densification laws. It is based on having new nodes attach to the network by copying links of the neighbors of neighbors recursively with probability p. However, the mathematical analysis of this model is quite complex and unavailable. Also we found in our experiment that the degree does not follow power law when p is large enough. While ER networks exhibit a small average shortest path length, they have a small clustering coefficient (A *clustering coefficient* is a measure of the community structure defined as follows, "Suppose that a node v has k_v neighbors; then at most $\frac{k_v(k_v-1)}{2}$ edges can exist between them (this occurs when every neighbor of v is connected to every other neighbor of v). Let C_v denote the fraction of these allowable edges that actually exist. Define C as the average of C_v over all v" [70].). WS network [70] and NW network [55] are designed to study the generations of small world phenomenon and high clustering coefficient by an unified mechanism. The PLRG networks [61] try to generate the power law or other degree distribution of real-world network manually. The connected components [16], average distances [17] and so on are discussed in CL networks. Hierarchical network [59] is designed by hierarchically copying blocks to produce a large network based on the observation of hierarchical modularity. The degree of hierarchical network follows a power law with exponent 2.26. Kronecker network [41] is another complex network model which is based on the observation of hierarchical modularity. Kronecker network obeys properties which include densification law, multinormial degree distribution, short diameter and so on. Kronecker network proposed two interpretations: hierarchical modularity, and the similarities of nodes. However, the interpretations are too vague from simple rules which is easy to control. BTER model [63] was designed to generate a complex network with power law degree distribution, community structure, power law community size distribution by manually locating nodes to several blocks. Multiscale network [29] is a flexible network which considers the diversities of real networks. It synthesizes realistic ensembles of networks starting from a known network,

through a series of mappings that coarsen and later refine the network structure by randomized editing.

The densest subgraph problem was first formally introduced by Goldberg [26]. He gave an algorithm that requires $O(\log n)$ running time (n is the number of vertexes in the graph) to find the optimal solution by reducing the problem to a series of min-cut max-flow computations. Later on, different subproblems of the dense subgraph problem were proposed. Feige *et al.* [23] defined and studied the densest k-subgraph problem, which is to find a subgraph with the maximum density among subgraphs containing k vertices. Asahiro *et al.* [3] defined and studied the problem of discovering a k-vertex subgraph of a given graph G that has at least $f(k)$ edges. Saha *et al.* [62] defined the densest subgraph problems with a distance restriction or a specific subset restriction, and provided algorithms for these subproblems. However, this work neglects the connectivity of the returned graphs.

Various approximate and heuristic algorithms have been proposed to improve the time and space complexity of the initial algorithms for big data. Charikar [11] presented a simple greedy algorithm that leads to a 2-approximation to the optimum. This algorithm was improved by Bahmani *et al.* [4] in a MapReduce framework, which can lead to a $2(1+\varepsilon)$-approximation of the optimum. The most important problem in these previous algorithms [4, 11] is that they neglect the connectivity of the returned densest subgraph. Some heuristic algorithms [12, 24] for discovering dense subgraphs were also proposed based on different techniques such as shingling and matrix blocking.

The applications of dense subgraph problem are accompanied by theoretical works. Kumar *et al.* [38] proposed an approach to identify web page communities in the Internet based on dense subgraphs. Gibson *et al.* [24] applied the solution for discovering dense subgraphs problem to detect the link spam in World Wide Web. These works use a threshold to determine the returned subgraphs. These algorithms also neglect the connectivity problem of the detected subgraphs. Also, there are no previous works that find all dense subgraphs which do not contain denser subgraphs or are contained in denser subgraphs, which however are needed in many applications.

In addition to the neglect of the connectivity, there has been no previous works that find all significant dense subgraphs. Compared to previous works, our study of dense subgraphs is novel in that i) we define two new subproblems of the dense subgraph problem, which consider the connectivity of the outputs, and find all significant dense subgraphs, and ii) our algorithms can be easily applied for handling GB-level natural graphs with no approximations in one PC with high

time and memory efficiency.

3.1.2 Preliminaries

We first introduce the concepts used in this section. Let $G = (V, E)$ be an undirected graph. For a subset $S \subseteq V$, the *induced edge set* is defined as $E(S) = E \cap S^2$ and the *induced degree* of a node $i \in S$ is defined as $\deg_S(i) = |\{j|(i,j) \in E(S)\}|$. In the remaining parts of this section, we use vertex subset S to denote graph $G_S = (S, E(S))$ for short sometimes.

Definition 1 (Density of an undirected graph [4]:) *Let $G = (V, E)$ be an undirected graph. Given $S \subseteq V$, its density $\rho(S)$ is defined as $\rho(S) = \frac{|E(S)|}{|S|}$. The maximum density $\rho^*(S)$ of the graph is $\rho^*(S) = \max_{S \subseteq V}\{\rho(S)\}$.*

The densest subgraph is the subgraph that has the maximum density.

Definition 2 (Capacity of an edge [66]:) *Let $N = (V, E)$ be a network (directed graph) with s and t being the source and the sink of N respectively. The capacity of an edge (u, v) is denoted by $c(u, v)$. It represents the maximum amount of flow that can pass through an edge.*

Definition 3 (Capacity of a cut [66]:) *A cut $C = (V_1, V_2)$ is a partition of V, where $V_1 \cup V_2 = V$, and $V_1 \cap V_2 = \emptyset$. The cut-set of C is the set $\{(u, v) \in E|u \in V_1, v \in V_2\}$. The capacity of a cut $C = (V_1, V_2)$ is defined by $c(V_1, V_2) = \sum_{(u,v)\in V_1 \times V_2} c(u, v)$. It represents the sum of the capacities of the edges connecting two partitions V_1 and V_2 (i.e., cut-set).*

Definition 4 (Min-cut max-flow (min-cut in short) problem [66]:) *This problem is to minimize $c(V_1, V_2)$, that is, to determine V_1 and V_2 such that the capacity of cut $c(V_1, V_2)$ is minimal.*

When all the capacities of the edges in the graph are nonnegative, the min-cut problem can be solved in polynomial time [66]. However, if there are negative edges, the min-cut problem is an NP-hard problem [50]. The min-cut problem with nonnegative capacities was applied to solve the original densest subgraph problem [26]. In order to solve the LCDS problem in this section, we first propose a two connected partitions min-cut problem as follows:

Definition 5 (Two connected partitions min-cut problem (TCPM):) *This problem is to minimize $c(V_1, V_2)$, that is, to determine V_1 and V_2 such that the capacity of the cut is minimal, and meanwhile vertex subsets V_1 and V_2 are connected, respectively.*

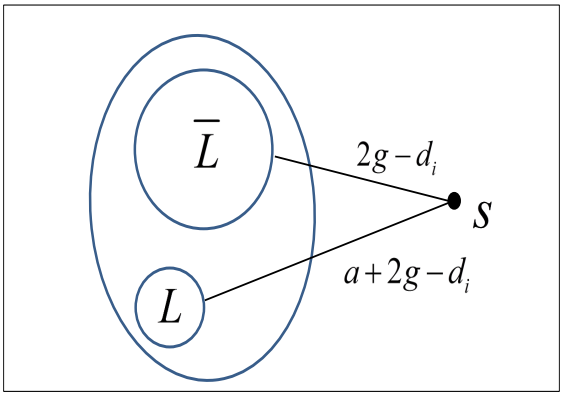


Figure 3.1: Graph construction

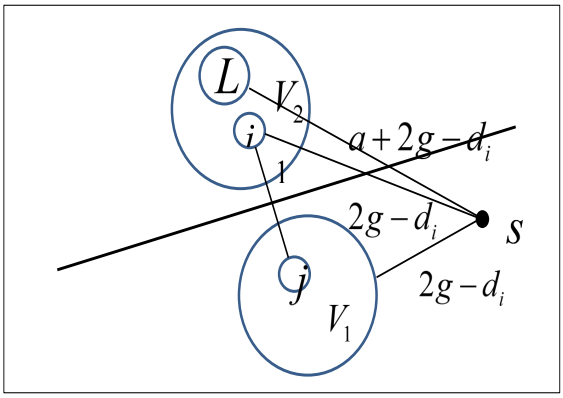


Figure 3.2: TCPM graph partition

This problem can be solved in polynomial time using a simple min-cut algorithm [66], since a simple min-cut algorithm just considers the min-cut of the two connected partitions and negative capacity does not influence the validity of the algorithm.

3.1.3 Discovering the LCDS

In this section, we define the LCDS problem and propose a polynomial-time solution.

Definition 6 (Local connected densest subgraph of a vertex subset L:) *Given a connected graph $G = (V, E)$, and a certain vertex subset L, where $L \subseteq V$. We call $G_L = (L^+, E(L^+))$ a local connected densest subgraph (LCDS) of L, if and only if that subgraph G_L is connected, $L \subseteq L^+$, and there are no such connected subgraph $G_S = (S, E(S))$ that satisfies $\rho(S) > \rho(L^+)$, and $L \subseteq S$.*

3.1.3.1 A Polynomial-time Algorithm

According to the interesting relationship between LCDS and TCPM problems shown later, the LCDS problem can be solved by trial and error in polynomial time. Given an estimated density of the LCDS of L (denoted by g), we can construct a specific graph (denoted by N). The LCDS can be reduced from one partition of TCPM of N if the estimation is right. Otherwise, we can judge whether g is too big or small based on the result whether the capacity of TCPM (denoted by c_{min}) is bigger than a certain value. Therefore, we can adjust the lower and upper bounds of the density of the LCDS of L (denoted by l_{min} and u_{min}, respectively) and give g a new value based on l_{min} and u_{min}. In this way, we apply a binary searching scheme to reach the "just right" g.

We first set $l_{min} = 0$ and $u_{min} = n$ where $n = |V|$. In the step of graph construction, given

Algorithm 1: Algorithm for discovering LCDS of L

```
Given: G = (V, E);
l_min ← 0, u_min ← n;
while (l_min − u_min) ≥ 1/(n(n−1)) do
    g ← (l_min + u_min)/2;
    Construct N = (V_N, E(V_N));
    Find TCPM (V_1 ∪ s, V_2);
    Calculate c_min;
    if c_min ≥ a|L| then
        u_min ← g
    end
    if c_min < a|L| then
        l_min ← g
    end
end
return subgraph of G induced by V_1;
```

a g, we convert the graph $G = (V, E)$ to graph $N = (V_N, E_N)$ as shown in Figure 3.1. We add a vertex s to the set of vertices of V, allocate each edge of E by a capacity of 1, connect every vertex i of $V \backslash L$ to vertex s by an edge of capacity $(2g - d_i)$, and connect every vertex i of L to vertex s by an edge of capacity $(a + 2g - d_i)$, where a is a negative constant smaller than the twice of the sum of other negative edge capacities in the graph N and d_i is the degree of vertex i of G. More formally,

$$
\begin{aligned}
V_N = \quad & V \cup \{s\} \\
E_N = \quad & \{(i,j) | \{i,j\} \in E(V)\} \cup \{(i,s) | i \in L\} \\
& \cup \{(i,s) | i \in V \backslash L\} \\
c_{ij} = \quad & 1 \qquad\qquad\qquad \{i,j\} \in E(V) \\
c_{si} = \quad & 2g - d_i \qquad\qquad i \in V \backslash L \\
c_{it} = \quad & a + 2g - d_i \qquad\qquad i \in V
\end{aligned}
$$

Then, we find a TCPM $(V_1 \cup s, V_2)$ of N. In Theorem 3.1.1, we find the relationship between c_{min} and the density bounds of the LCDS. Therefore, we can adjust g based on c_{min}. If $c_{min} \geq a|L|$, we update u_{min} with the current value of g; if $c_{min} < a|L|$, we update $l_{min}$with the current value of g. When the stop condition ($(l_{min} - u_{min}) < \frac{1}{n(n-1)}$ which is proved in Theorem 3.1.2) is satisfied, we get the final LCDS of L from V_2. Otherwise, we update g by $g = (l_{min} + u_{min})/2$ using the binary searching scheme, re-construct N based on the updated g, and repeat the above process until

the stop condition is satisfied.

In the following, we analyze the validity of this algorithm including the determination of lower and upper bounds of the density of the LCDS of L, the determination of the stop condition, the connectivity of the discovered LCDS of L and its time complexity.

3.1.3.2 Proving the Validity of the Algorithm

Relationship between capacity and density bounds of the LCDS: In order to continually narrow the scope of the density of the LCDS, we need a method to determine the lower and upper bounds based on the constructed TCPM in each loop in Algorithm 1. Therefore, we prove the relationship between the possible capacity of the constructed TCPM and the bounds of the LCDS below.

Lemma 3.1.1 *Suppose $(V_1 \cup \{s\}, V_2)$ is a TCPM of the above constructed graph $N = (V_N, E_N)$, then vertex subset $L \subset V_2$.*

Proof 3.1.1 *We know that any cut $c(V_1, V_2)$ where $L \subset V_2$ has capacity $c_L \leq (a|L| - \frac{a}{2})$, since a is a negative constant smaller than the twice of the sum of other negative edge capacities in the graph N. Suppose there is a cut with capacity c_{L^-} which a vertex subset L^- where $L^- \subset L$ and $L^- \subset V_1$, then $c_{L^-} > a|L \backslash L^-| + \frac{a}{2}$. Also, we know $(a|L \backslash L^-| + \frac{a}{2}) \geq (a|L| - \frac{a}{2})$, since $L^- \neq \emptyset$. Therefore, $c_{L^-} > c_L$. Hence, vertex subset $L \subset V_2$ for TCPM $(V_1 \cup \{s\}, V_2)$.*

Theorem 3.1.1 *Suppose $c(V_1 \cup s, V_2)$ is a TCPM of the graph $N = (V_N, E_N)$, which has capacity c_{min} and L^+ is the LCDS of L, then the g parameter in Algorithm 1 satisfies $g \geq \rho(L^+)$ if and only if $c_{min} \geq a|L|$, and it satisfies $g \leq \rho(L^+)$ if and only if $c_{min} \leq a|L|$.*

Proof 3.1.2 *As we can see from Figure 4.2, the capacity of the TCPM equals:*

$$\begin{aligned} c_{min} &= \sum_{i \in V_1, j \in V_2} c_{ij} \\ &= \sum_{j \in V_2} c_{sj} + \sum_{i \in V_1, j \in V_2} c_{ij} \end{aligned}$$

We know that vertex s is only connected to vertices in vertex subset L from the definition of TCPM. Also based on Lemma 3.1.1, we know that $L \subset V_2$. Hence, $\sum_{j \in V_1} c_{sj} = 0$. Therefore, we have:

$$
\begin{aligned}
c_{min} &= \sum_{i\in L}(a+2g-d_i) + \sum_{i\in V_2\setminus L}(2g-d_i) + \sum_{i\in V_1, j\in V_2} 1 \\
&= a|L| + 2|V_2|(g - \frac{\sum_{i\in V_2} d_i - \sum_{i\in V_1, j\in V_2} 1}{2|V_2|}) \\
&= a|L| + 2|V_2|(g-\rho(V_2))
\end{aligned}
$$

Since $|V_2| \geq |L|$ and $|L| > 0$, $2|V_2|(g-\rho(V_2)) = 0$ if and only if $g = \rho(V_2)$. Suppose $g \leq \rho(L^+)$, then we can find a cut $c(V_1, V_2)$ where $\rho(V_2) \geq g$. Such a cut can lead to $2|V_2|(g-\rho(V_2)) \leq 0$. Hence, capacity $c_{min} \leq a|L|$. Conversely, suppose $c_{min} \leq a|L|$, then, $2|V_2|(g-\rho(V_2)) \leq 0$. Then $g \leq \rho(V_2)$. We know $\rho(V_2) \leq \rho(L^+)$ from the definition. Therefore, $g \leq \rho(L^+)$.

Suppose $g \geq \rho(L^+)$, then we know $\rho(V_1) \leq \rho(L^+)$ from the definition. Hence, $2|V_2|(g-\rho(V_2)) \geq 0$. Therefore the TCPM $c_{min} \geq a|L|$. Conversely, suppose the TCPM $c_{min} \geq a|L|$. Then, $2|V_2|(g-\rho(V_2)) \geq 0$. Then, $g \neq \rho(V_2)$. Also $\rho(V_2) \leq \rho(L^+)$. Therefore, $g \leq \rho(L^+)$.

Based on the above property, we design the binary searching scheme of the algorithm.

Determining the stop condition of the algorithm:

Lemma 3.1.2 *Suppose there is a connected graph $N = (V_N, E_N)$, two vertex subsets S_1 and S_2 where $S_1, S_2 \subset V_N$. Then, $|\rho(S_1) - \rho(S_2)| \geq \frac{1}{n(n-1)}$, where $n = |V_N|$.*

Proof 3.1.3 *Suppose the number of the edges in subgraph $G_1 = (S_1, E(S_1))$ and $G_2 = (S_2, E(S_2))$ is m_1 and m_2, respectively. Then, $\rho(S_1) = \frac{m_1}{|S_1|}$, and $\rho(S_2) = \frac{m_2}{|S_2|}$. We have:*

$$
\begin{aligned}
|\rho(S_1) - \rho(S_2)| &= |\frac{m_1}{|S_1|} - \frac{m_2}{|S_2|}| \\
&= \frac{|m_1|S_2| - m_2|S_1||}{|S_1||S_2|}
\end{aligned}
$$

Since $\rho(S_1) \neq \rho(S_2)$, we can divide the above equation to three conditions: i) $|S_1| > |S_2|, m_1 \leq m_2$; ii) $|S_1| > |S_2|, m_1 \geq m_2$; and iii) $|S_1| = |S_2|, m_1 \leq m_2$.

In case i), $|S_1| \cdot |S_2| \leq \frac{1}{n(n-1)}$ and $m_1|S_2| - m_2|S_1| \geq m_1$. Then, we have $|\rho(S_1) - \rho(S_2)| = \frac{m_1}{n(n-1)}$. Hence, $|\rho(S_1) - \rho(S_2)| \geq \frac{1}{n(n-1)}$. Similarly, in case ii), $|\rho(S_1) - \rho(S_2)| \geq \frac{1}{n(n-1)}$. In case iii), $|S_1| < n$ since $\rho(S_1) \neq \rho(S_2)$. Therefore, $|\rho(S_1) - \rho(S_2)| \geq \frac{1}{(n-1)^2}$. In all the three conditions, $|\rho(S_1) - \rho(S_2)| \geq \frac{1}{n(n-1)}$.

Theorem 3.1.2 *Suppose* $|u_{min} - l_{min}| < \frac{1}{n(n-1)}$, *then we can ensure that the graph with density bigger than* l_{min} *is the LCDS solution.*

Proof 3.1.4 *Based on Lemma 3.1.2, we know we can guarantee that there is only one subgraph with a density bigger than* l_{min} *if* $|u_{min} - l_{min}| < \frac{1}{n(n-1)}$. *Then, proof completes.*

Connectivity guarantee of the LCDS of L:

Theorem 3.1.3 *Suppose* $(V_1 \cup s, V_2)$ *is a TCPM of the graph* $N = (V_N, E_N)$, *then vertex subset* V_2 *is connected.*

Proof 3.1.5 V_2 *is one partition of the TCPM. Therefore, we know that* V_2 *is connected from the definition of the TCPM problem.*

Theorem 3.1.3 indicates that the LCDS of L discovered by Algorithm 1 is connected.

Time complexity: Previous studies have proved that TCPM problem can be solved in polynomial time [66]. In the experiment, we apply a simple min-cut algorithm [36] with time complexity $O(|V||E|+|V|^2 log|V|)$. Also, the stop condition can be met in $O(log|V_N|)$ times of estimations (the time complexity of binary search) and each estimation requires one TCPM computation. Then, the total time complexity of the LCDS discovery algorithm is $O(|V||E|log|V| + |V|^2 log^2|V|)$.

3.1.4 Discovering the GSDSs

In this section, we define the GSDS problem, and propose a polynomial-time solution.

Definition 7 (Global significant dense subgraph:) *Given a connected graph* $G = (V, E)$, *and a connected subgraph* $G_S = (S, E(S))$ *where* $S \subset V$. G_S *is a GSDS if and only if there are no such connected subgraph* $G_{S^+} = (S^+, E(S^+))$ *that satisfies* $\rho(S^+) > \rho(S)$ *and* $S \subset S^+$, *and also there are no such connected subgraph* $G_{S^-} = (S^-, E(S^-))$ *that satisfies* $\rho(S^-) > \rho(S)$ *and* $S^- \subset S$.

3.1.4.1 A Basic Polynomial-time Algorithm

Based on the definition, the idea of this basic algorithm is to find candidates, which do not contain denser subgraphs, and then check whether they are contained in denser subgraphs to get the final results. This algorithm checks GSDSs based on the following three rules. First, the densest subgraph is a GSDS. Second, GSDSs are disjointed from each other. Third, if a candidate subgraph

is the LCDS of itself and does not contain denser subgraphs, then it is a GSDS. Accordingly, we reduce the GSDS problem of G to a series of computations of discovering the densest subgraph and the LCDS problems, which can be solved by the algorithm in [26] and Algorithm 1, respectively.

Algorithm 2 presents the pseudocode of this basic polynomial-time algorithm. In step 1 (block 3 - 4), we find the densest subgraph $G_1 = (V_1, E(V_1))$ of G, and add G_1 to GSDS list D_{list}. In step 2 (block 6), we find the densest subgraph $G_2 = (V_2, E(V_2))$ from the remaining graph $G \backslash G_1$. For subgraph G_2, there are no denser subgraphs inside G_2 based on the definition of the densest subgraph. However, to check whether there are denser subgraphs containing G_2, we find LCDS $G_{2L} = (V_{2L}, E(V_{2L}))$ of V_2. If $\rho(V_2) \geq \rho(V_{2L})$, then there are no denser subgraphs contain G_2, which we will prove in Theorem 3.1.6, and add G_2 to D_{list}. We continue this process in the remaining graph of G until the remaining of graph G becomes empty. It is obvious that the maximum times of the process is $|V|$.

Algorithm 2: Basic algorithm for discovering GSDSs

```
1: Given: G = (V, E);
2: G_0 = (V_0, E_0) ← G;
3: Find densest subgraph G_1 = (V_1, E(V_1)) of G;
4: Add G_1 to D_list;
5: G ← G\G_1, i ← 2;
6: while G ≠ ∅ do
     Find G_i = (V_i, E(V_i)) of G;
     // G_i = (V_i, E(V_i)) is the densest subgraph of G
     Find G_iL = (V_iL, E(V_iL));
     // G_iL = (V_iL, E(V_iL)) is the LCDS of V_i of G_0
     G ← G\G_i, i++;
     if ρ(V_i) ≥ ρ(V_iL) then
       Add G_i to D_list;
     end
   end
7: return D_list;
```

3.1.4.2 Proving the Validity and Properties of the Algorithm

In this section, we prove the three rules followed in this algorithm by studying the properties of GSDSs and analyze the time complexity.

Properties of GSDSs:

Theorem 3.1.4 *Suppose $G_1 = (V_1, E(V_1))$ is the densest subgraph of graph $G = (V, E)$, then G_1 is a GSDS.*

Proof 3.1.6 *From the definition of the densest subgraph, we know there are no subgraphs of G that are denser than G_1. Then, there are no subgraphs that contain G_1 are denser than G_1. Also, there are no subgraphs contained in G_1 that are more denser than G_1. Therefore, by the definition of the GSDS, G_1 is a GSDS.*

Theorem 3.1.4 supports our first rule.

Theorem 3.1.5 *Suppose $G_1 = (V_1, E(V_1))$ and $G_2 = (V_2, E(V_2))$ are two GSDSs of graph $G = (V, E)$, then we have $V_1 \cap V_2 = \emptyset$.*

Proof 3.1.7 *We prove it by contradiction. Suppose $V_1 \cap V_2 \neq \emptyset$, then we have a connected subgraph $G_{12} = ((V_1 \cup V_2), E(V_1 \cup V_2))$. Then we calculate the density $\rho(V_1 \cup V_2)$ as follows:*

$$\rho(V_1 \cup V_2) \quad = \frac{E(V_1 \cup V_2)}{|V_1 \cup V_2|}$$

Let $V_b = V_1 \cap V_2$. Then we have:

$$\rho(V_1 \cup V_2) \quad = \frac{\rho(V_1)|V_1| + \rho(V_2)|V_2| - \rho(V_b)|V_b|}{|V_1| + |V_2| - |V_b|}$$

Since G_1 and G_2 are GSDSs, we have $\rho(V_1) > \rho(V_b)$ and $\rho(V_2) > \rho(V_b)$. Suppose $\rho(V_1) \geq \rho(V_2)$, then we have:

$$\begin{aligned} \rho(V_1 \cup V_2) \quad &> \frac{\rho(V_2)|V_1| + \rho(V_2)|V_2| - \rho(V_2)|V_b|}{|V_1| + |V_2| - |V_b|} \\ &= \rho(V_2) \end{aligned}$$

Therefore, G_2 is not a GSDS from the definition. It contradicts with the assumption. Therefore, we must have $V_1 \cap V_2 = \emptyset$.

Theorem 3.1.5 supports our second rule, which guarantees that we can find all the GSDSs.

Theorem 3.1.6 *Suppose ${G_1}^* = ({V_1}^*, E({V_1}^*))$ is the densest subgraph of part of the graph G, then ${G_1}^*$ is the GSDS of G, if and only if ${G_1}^*$ is the LCDS of ${V_1}^*$ of G.*

Proof 3.1.8 *Firstly, we prove that ${G_1}^*$ is the GSDS if ${G_1}^*$ is the LCDS of ${V_1}^*$ of G. From the definition of the LCDS, we know there are no subgraph ${G_0}^* = ({V_0}^*, E({V_0}^*))$ where ${V_1}^* \subset {V_0}^*$ and $\rho({V_0}^*) > \rho({V_1}^*)$. Also, from the densest subgraph definition, we know there are no subgraph ${G_0}^* = ({V_0}^*, E({V_0}^*))$ where ${V_0}^* \subset {V_1}^*$ and $\rho({V_0}^*) > \rho({V_1}^*)$. Therefore, ${G_1}^*$ is the LCDS of ${V_1}^*$ of G. Secondly, we prove that ${G_1}^*$ is the GSDS only if ${G_1}^*$ is the LCDS of ${V_1}^*$ in G by contradiction. Suppose ${G_1}^*$ is not LCDS of ${V_1}^*$ in G, then there is a subgraph ${G_0}^* = ({V_0}^*, E({V_0}^*))$ where ${V_1}^* \subset {V_0}^*$ and $\rho({V_0}^*) > \rho({V_1}^*)$. Then ${G_1}^*$ is not a GSDS by the definition. Therefore, the condition that ${G_1}^*$ is the LCDS of ${V_1}^*$ of G is the necessary condition for that ${G_1}^*$ is a GSDS.*

Algorithm 3: Improved algorithm for discovering GSDSs

Given: $G = (V, E)$;
$S \leftarrow V$, $S_p \leftarrow \emptyset$, $\rho_{max} \leftarrow \rho(S)$;
while $S \neq S_p$ **do**
  $S_p \leftarrow S$;
  $S_c \leftarrow \{i \in S | deg_S(i) \leq \rho_{max}\}$;
  $S \leftarrow S \backslash S_c$;
  if $\rho(S) > \rho_{max}$ **then**
    $\rho_{max} \leftarrow \rho(S)$;
  end
end
$G_0 = (S_0, E(S_0)) \leftarrow G_S = (S, E(S))$;
Find densest subgraph $G_1 = (V_1, E(V_1))$ of G_S;
Add G_1 to D_{list};
${V_1}^* \leftarrow V_1 \cup \{j | (i, j) \in E(S), i \in V_1\}$;
${G_1}^* = ({V_1}^*, E({V_1}^*))$;
${G_S} \leftarrow {G_S} \backslash {G_1}^*$, $i \leftarrow 2$;
while ${G_S} \neq \emptyset$ **do**
  Find $G_i = (V_i, E(V_i))$;
  // it is the densest subgraph of G_S
  Find $G_{iL} = (V_{iL}, E(V_{iL}))$;
  // it is the LCDS of V_i of G_0
  ${V_i}^* \leftarrow V_i \cup \{j | (i, j) \in E(S), i \in V_i\}$;
  ${G_i}^* = ({V_i}^*, E({V_i}^*))$;
  ${G_S} \leftarrow {G_S} \backslash {G_i}^*$, $i++$;
  if $\rho(V_i) \geq \rho(V_{iL})$ **then**
    Add G_i to D_{list};
  end
end
return D_{list};

Theorem 3.1.6 supports our third rule. **Time complexity:** In order to discover the densest subgraph, we choose the push-relabel algorithm with dynamic trees [15]. The time complexity of this algorithm is $O(|V||E|log(|V|^2/|E|))$. For the TCPM problem, we choose a simple min-cut algorith-

m [66]. The time complexity of this algorithm is $O(|V||E| + |V|^2 log|V|)$. Therefore, the total time complexity for computing one LCDS is $O(|V||E|log(|V|^2/|E|)+|V||E|+|V|^2 log|V|)$. The total time complexity for computing one GSDS is $O(|V|^2|E|log(|V|^2/|E|)log|V| + |V|^2|E| + |V|^3 log|V|)$. For natural graphs, the time complexity is approximately $O(|V|^2|E|log^2|V|)$, since natural graphs are usually sparse graphs, which makes $|V| \approx |E|$.

Even though the basic GSDS algorithm can solve the problem in polynomial time, a time complexity of $O(|V|^2|E|log^2|V|)$ is still too high for large datasets especially in this big data era. To reduce its time complexity, we propose an improved GSDS algorithm below.

3.1.4.3 An Improved Algorithm for Large Datasets

It is well-known that the natural graphs usually follow a power-law degree distribution [5]. For graphs with such a feature, most of the vertices have low probabilities to be in the GSDSs, since they have very low degrees. Therefore, the basic idea of this improved algorithm is trying to reduce the initial size of the dataset by deleting the vertices with very low degrees. The detailed process is presented in Algorithm 3. For a given $G = (V, E)$, we first delete all the vertices, which have degrees equal or smaller than the maximum density of remaining graph during the deleting process (blocks 1-3) streamingly. Then, in addition to the same process as the basic algorithm, we delete the neighbors of the vertices of each densest subgraph found in the remaining graph (blocks 5-10). Block 11 returns the results. In the following, we prove the correctness of this algorithm.

Properties used for the improvement:

Lemma 3.1.3 *Suppose $G_S = (V_S, E(V_S))$ is the densest subgraph of graph G, then we have $deg_{V_S}(i) \geq \rho(V_S)$ for any vertex i, where $i \in V_S$.*

Proof 3.1.9 *We prove it by contradiction. Suppose there is at least one vertex i, where $i \in V_S$, and $deg_{V_S}(i) < \rho(V_S)$, then we delete vertex i from G_S, and get graph $G_{S^-} = (V_S \backslash \{i\}, E(V_S \backslash i))$. The density of graph G_{S^-} is:*

$$\begin{aligned} \rho(V_S \backslash \{i\}) &= \frac{|E(V_S \backslash i)|}{|V_S \backslash \{i\}|} \\ &= \frac{\rho(V_S)|V_S| - deg_{V_S}(i)}{|V_S| - 1} \end{aligned}$$

Since $deg_{V_S}(i) < \rho(V_S)$, we have:

$$\begin{aligned}\rho(V_S\backslash\{i\}) &> \frac{\rho(V_S)(|V_S|-1)}{|V_S|-1}\\ &> \rho(V_S)\end{aligned}$$

This contradicts with the precondition. Therefore, we have $deg_{V_S}(i) \geq \rho(V_S)$ *for any vertex* i*, where* $i \in V_S$*.*

Theorem 3.1.7 *After we delete all the vertices with degrees less or equal than the maximum density of the remaining graph of* G *in block 3 of Algorithm 3 to obtain* G_S*, all the GSDSs of* G *are in* G_S*.*

Proof 3.1.10 *We prove it by contradiction. Suppose there is one GSDS* $G_x = (V_x, E(V_x))$*, where* $G_x \not\subset G_S$*, then there is a vertex subset* I *where* $I \subset V_x$ *and* $I \not\subset V_s$*. There is a time in the deleting process that the first vertex* i *in* I *is deleted from the current* V_s *(denoted by* ${V_s}^+$*). Therefore, we have:*

$$\begin{aligned}\rho(V_x) &\leq deg_{V_x}(i)\\ &\leq deg_{{V_s}^+}(i)\\ &< \rho({V_s}^+)\end{aligned}$$

This implies that $\rho(V_x) < \rho({V_s}^+)$*, and at this moment, we have* $V_x \subset {V_s}^+$*. Hence, from the definition of GSDS, we know that* G_x *is not a GSDS of* G*. This contradicts with the assumption. Therefore, for any GSDS* G_x *of* G*, we have* $V_x \subset V_S$*.*

Based on Theorem 3.1.7, we design block 3 in Algorithm 3.

Theorem 3.1.8 *Suppose* $G_1 = (V_1, E(V_1))$ *and* $G_2 = (V_2, E(V_2))$ *are two GSDSs of graph* $G = (V, E)$*, then there is no such an edge* (v_1, v_2) *in graph* G*, where* $v_1 \in V_1$ *and* $v_2 \in V_2$*.*

Proof 3.1.11 *We prove it by contradiction. Suppose there is at least one edge* (v_1, v_2) *where* $v_1 \in V_1$ *and* $v_2 \in V_2$*, then we calculate the density of* $G_0 = (V_1 \cup V_2, E(V_1 \cup V_2))$ *as follows:*

$$\begin{aligned}\rho(V_1 \cup V_2) &= \frac{|E(V_1 \cup V_2)|}{|V_1 \cup V_2|}\\ &\geq \frac{\rho(V_1)|V_1| + \rho(V_2)|V_2| + 1}{|V_1| + |V_2|}\end{aligned}$$

Suppose $\rho(V_1) \geq \rho(V_2)$, then we have:

$$\begin{aligned}\rho(V_1 \cup V_2) &> \frac{\rho(V_2)|V_1| + \rho(V_2)|V_2|}{|V_1| + |V_2|} \\ &= \frac{\rho(V_2)(|V_1| + |V_2|)}{|V_1| + |V_2|} \\ &= \rho(V_2)\end{aligned}$$

Also, we know that $V_2 \subset V_1 \cup V_2$. Therefore, G_2 is not a GSDS. This contradicts with the precondition. Therefore, there is no such an edge (v_1, v_2) in graph G, where $v_1 \in V_1$ and $v_2 \in V_2$.

Based on Theorem 3.1.8, we design block 10 of Algorithm 3.

Theorem 3.1.9 *Suppose $G^* = (V^*, E(V^*))$ is a subgraph of $G = (V, E(V))$ and ${G_1}^* = ({V_1}^*, E({V_1}^*))$ is the densest subgraph of G^*, then there are no GSDSs of G in G^*, if $\rho({V_1}^*) < \rho(V)$.*

Proof 3.1.12 *Since $V^* \subseteq V$ and ${V_1}^* \subseteq V^*$, we have ${V_1}^* \subseteq V$. Also, we know $\rho({V_1}^*) < \rho(V)$. Also, by the definition of the GSDS, ${G_1}^*$ is not a GSDS of G. Based on the definition of the densest subgraph, we know that, for any subgraph ${G_i}^* = ({V_i}^*, E({V_i}^*))$ in G^*, $\rho({V_i}^*) < \rho({V_1}^*) < \rho(V)$. Also, ${V_i}^* \subseteq V$. Hence, ${G_i}^*$ is not a GSDS of G. Therefore, there are no GSDSs of G in G^*.*

Based on Theorem 3.1.9, we design the stop condition of the loop in block 10 of Algorithm 3.

3.2 Measurement

The social network features in human mobility networks have been widely applied in DTN routing algorithms [18,33,34,53]. However, it is not clear whether VNETs have these social network features since vehicles, especially taxies, may follow random customers' demands to move. Therefore, we analyze two real world VNET traces gathered by taxi GPS in different cities, referred to as *Roma* [49] and *SanF* [58]. The *Roma* trace contains mobility trajectories of 320 taxies in the center of Roma from Feb. 1 to Mar. 2, 2014. The *SanF* trace contains mobility trajectories of approximately 500 taxies collected over 30 days in San Francisco Bay Area.

We first transfer the traces to *contact graph*s based on the contact durations. The nodes of the graphs are the taxies in the traces, the edges are the contacts between pairs of taxies. We naturally think that if two vehicles encounter each other more often, they are in a closer relationship

and only the contacts which have accumulative durations long enough can be considered as edges. Therefore, in the following measurement, we define an accumulative contact duration threshold (3000s in *Roma* and 5000s in *SanF*) and the contacts with accumulative durations larger than the threshold can be considered as edges. In this way, we transform the mobility traces into complex networks and study the following social network features:

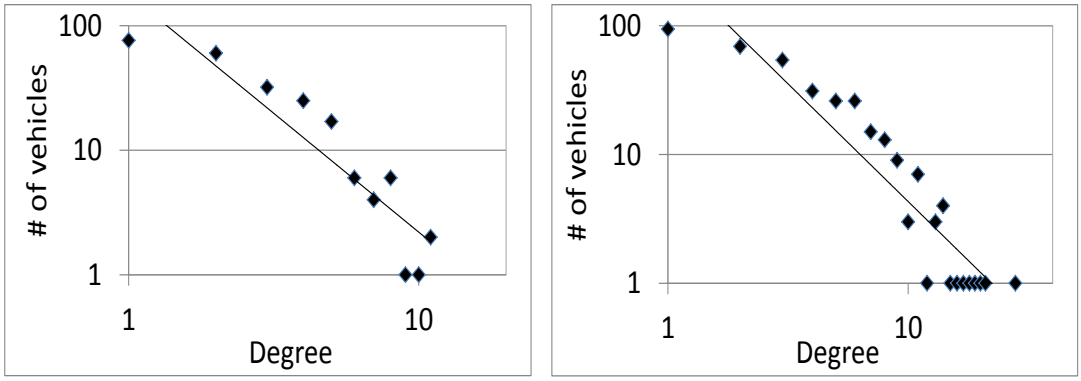
1. **Important nodes:** In a complex network, a few nodes play an important role in guaranteeing the connectivity of the network and spreading information [2]. Forwarding packets to such important nodes can enhance the routing efficiency. Here, we use degree and PageRank to measure the important vehicles and define important vehicles as vehicles with high degrees and PageRank values. The degree of each vehicle is calculated by counting the number of the edges the vehicle has in the contact graph. The PageRank value of each vehicle is calculated by counting the number and quality of edges to the vehicle.

2. **Community structure:** A network is said to have the community structure if nodes can be easily grouped into communities which are densely connected internally [25]. In this section, we use modularity [56] to measure the community structure. Modularity is the degree to which the network can be clustered as communities. The higher modularity the network has, the more obvious the community structure is.

3. **Fractal structure feature of the community:** The fractal structure feature of the community is the pattern that many small, highly connected communities combine in a hierarchical manner into larger, less cohesive communities recursively in networks [60]. Previous study [6] indicates that a power law clustering coefficient [71] distribution is the necessary and sufficient condition of the fractal structure feature of the community. Therefore, we measure the clustering coefficient distributions of the two traces for verifying fractal structure feature of the community in VNETs. Clustering coefficient of a vehicle is calculated by quantifying how close its neighbors are to be a clique (complete graph).

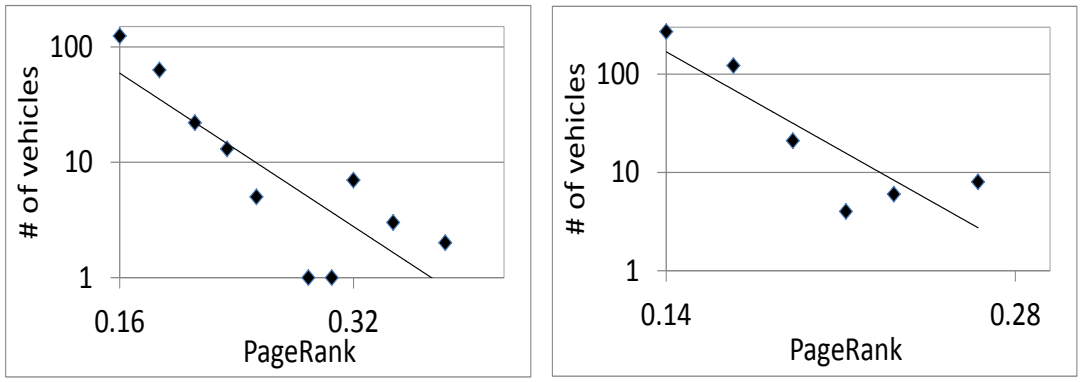
3.2.1 Important Nodes

Figure 3.3 shows the degree distributions of the two traces. We find that the degree distributions approximately follow power laws for both traces. Figure 3.4 shows the PageRank value distributions of the two traces. We find that the PageRank value distributions of the two traces also



(a) *Roma* (b) *SanF*

Figure 3.3: The degree distribution.



(a) *Roma* (b) *SanF*

Figure 3.4: The PageRank value distribution.

approximately follow power laws, which are consistent with degree distributions. Figure 3.3 and Figure 3.4 indicate that there are a few very important vehicles with high degrees and PageRank values. Therefore, we conclude our first observation (**O1**) as follows:

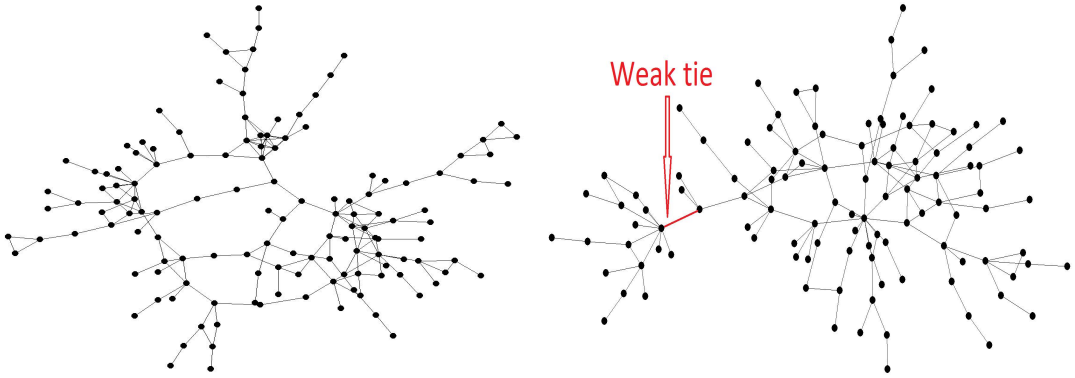
O1: *There are several very important vehicles with high degrees and PageRank values, which have high probability to encounter various and a large number of other vehicles.*

3.2.2 Community Structure in VNETs

Then, we measure the modularity [56] of each trace by Graphi [1] (A complex network analysis software). The results are shown in Figure 3.5. The modularity values of the two traces are 0.74 and 0.78, respectively. The social networks with the community structure in the real world usually have modularity values between 0.4 and 0.7 [56]. Therefore, the high modularity values of the two traces indicate their obvious community structures.

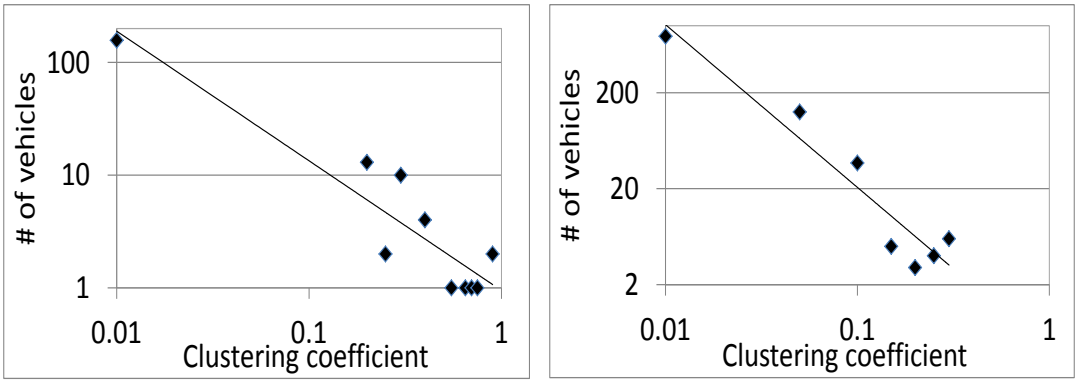
Trace	Modularity
Roma	0.74
SanF	0.78

Figure 3.5: Modularity



(a) *Roma* (b) *SanF*

Figure 3.6: The topologies of the largest connected subgraphs.



(a) *Roma* (b) *SanF*

Figure 3.7: The clustering coefficient value distribution.

In order to show the community structures clearly, we draw network topologies of the maximal connected components of the two traces in Figure 3.6. As shown in Figure 3.6, there are clearly dense components which are corresponding to different communities in VNETs. Also, we can clearly observe the weak ties, which connect different dense components. Therefore, we conclude our second observation (**O2**) as follows:

O2: *The VNETs have obvious community structures connected by weak ties.*

3.2.3 Fractal Structure Feature of the Community in VNETs

Figure 3.7 shows the clustering coefficient distributions of the two traces. The power law clustering coefficient distributions shown in Figure 3.7 indicate that two traces both have fractal structure feature of the community. Therefore, we conclude our third observation (**O3**) as follows:

O3: *The communities in VNETs present fractal structure feature.*

The above three observations show that, even for the taxi networks in which the mobilities

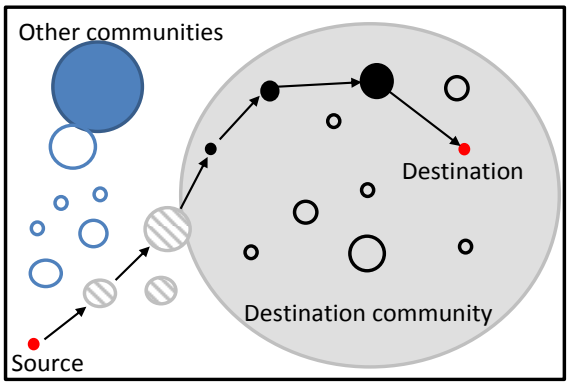


Figure 3.8: An example of the routing process.

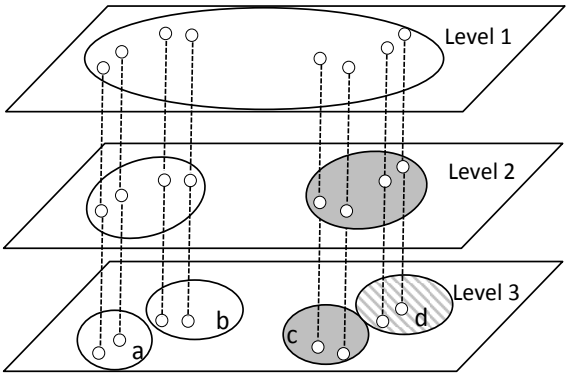


Figure 3.9: FCT.

are determined by random customers' demands, there are clear social network features as other human mobility networks [18, 34].

3.3 System Design

Before introducing the detailed design of SPread, we give an overview of the routing process for a packet as follows:

1. First, we scatter different copies of the packet to communities which are close to the destination community through weak ties as shown in Figure 1.1, which improves the multi-copy routing efficiency. Then the routing of each copy is divided into two phases: inter-community and intra-community.

2. In the inter-community routing, each copy is gradually forwarded to the vehicles with higher multi-hop reachability to the destination community as shown in Figure 4.8. Once a copy encounters the destination community, the copy is forwarded or replicated to the destination community according to different situations.

3. In the intra-community routing, for the copies which have reached the destination community, they are gradually forwarded to the vehicles with higher multi-hop reachability to the destination vehicle as shown in Figure 4.8.

Based on the overview, in this section, we first provide the method to identify the fractal structure feature of the community in VNETs and build Fractal Community structure Tree (FCT) by static GPS history data. FCT is used for scattering copies of each packet to different communities.

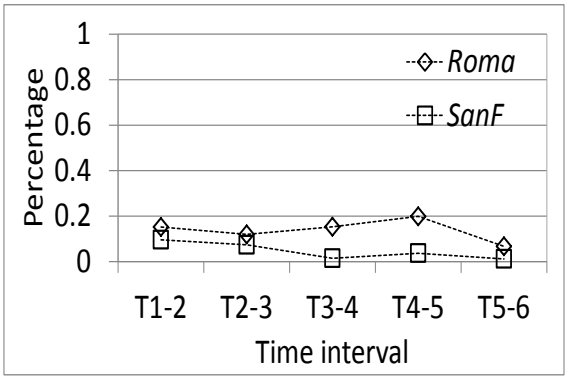


Figure 3.10: The stability of encounter frequencies.

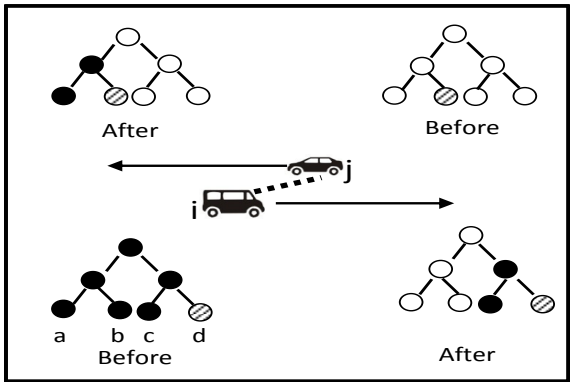


Figure 3.11: Multi-copy process.

Then, we describe the process of constructing IntEr-community personalized Rank Table (IERT) and IntrA-community personalized Rank Table (IART) for each vehicle dynamically. IERT is used for the routing in the inter-community routing phase. IART is used for the routing in the intra-community routing phase. IERT and IART both consider the multi-hop reachability of vehicles to different communities and vehicles. Next, we explain the details of our weak tie based multi-copy routing algorithm. Finally, the pseudocode and routing process descriptions of SPread from the micro-scope are given as a summary.

3.3.1 Building Fractal Community Structure Tree (FCT)

In order to scatter copies of each packet to communities which are close to the destination community, there are few questions. How can we identify communities? How can we calculate the distances among communities? In order to answer the questions, we need to understand the structure of VNETs. Therefore, we build Fractal Community structure Tree (FCT) by the static GPS history data for solving these questions. FCT saves the fractal structure feature of the community in a tree structure. Figure 3.9 shows an FCT for example. In the FCT, lowest level communities a, b, c and d present fractal structure feature, which combine themselves into higher level communities recursively. For the routing part, we only need the information of the communities in the lowest level. For example, as shown in Figure 3.9, we only need the information of communities a, b, c and d in level 3 for the inter-community and intra-community routing. However, for calculating the distances among communities and weak tie based multi-copy routing algorithm, we need to consider the tree information, which will be further explained in subsection 3.3.3.

3.3.1.1 Stability

VNETs are usually sparse distributed and building FCT is more complicated than identifying communities only. Therefore, it is preferable to build FCT by static GPS history data which can be easily obtained. However, the FCT is mainly determined by encounter frequencies among vehicles. Whether the encounter frequencies stay stable from time to time significantly influences the accuracy of FCT built by static GPS history data. Therefore, we first analyze the stability of vehicle encounter frequencies. We divide each of the two traces to 6 time intervals with equal length and count the top 5 frequently encountered vehicles of all the vehicles at the end of each time interval. Then we calculate the percentage of changes from one time interval to the next time interval and draw Figure 4.1. The $Ti - j$ on the x axis in Figure 4.1 means the changes from time interval i to time interval j. The y axis is the percentage of the changes from one time interval to the next time interval. As shown in Figure 4.1, the encounter frequencies among vehicles tend to be stable, which means that we can apply the recent GPS history data to construct the FCT for future system design.

3.3.1.2 BGLL algorithm

BGLL algorithm is an algorithm for fast discovering fractal communities [8] and has been widely applied to different applications. There are various community detecting algorithms. However, we need to not only detect the communities, but also find the fractal structure feature of the community easily. At the same time, we hope the algorithm is fast and precise. Therefore, BGLL algorithm is the most suitable algorithm. However, BGLL algorithm does not save the FCT by itself. Therefore, we introduce an improved BGLL algorithm for building the FCT as follows.

1. First, we consider each vehicle as a community and calculate the increased modularity ΔQ for adding any vehicle i to its neighbor j and add node i to its neighbor's community which has the positive maximum increased modularity ΔQ by

$$\Delta Q = \left[\frac{\sum_{in} + k_{i,in}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m} \right)^2 \right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right]$$

where $\sum_{in}$ is the sum of the weights (contact durations) of the edges inside the community that

j belongs to. $\sum_{tot}$ is the sum of the weights of the edges incident to nodes in the community that j belongs to, k_i is the sum of the weights of the edges incident to node i, $k_{i,in}$ is the sum of the weights of the edges from i to nodes in C and m is the sum of the weights of all the edges in the network. If all the ΔQ are negative, vehicle i stays in its initial community. This process is applied repeatedly for all nodes until the modularity cannot be further increased.

2. Then, we consider each community as a new node and reconstruct the network where the edge weight of any two communities is the sum of edge weight connecting two communities. If the number of nodes is more than 1, we go to step 1.

3. We construct the final FCT by traversing the community level by level until the community sizes reach a suitable size and record the lowest level community set as S.

3.3.2 Building IntEr-community Personalized Rank Table (IERT)

In the inter-community routing phase, in order to forward each copy to vehicles which are close to the destination community, we need to define a criterion for selecting relay vehicles. Probability, community and centrality based utilities all have their own drawbacks as mentioned in Chapter 2. Therefore, inspired by the personalized PageRank algorithm, we design the personalized CommunityRank algorithm and build IntEr-community personalized Rank Table (IERT) for each vehicle dynamically. IERT records the multi-hop reachability of a vehicle to different communities (called CommunityRank). The CommunityRank can count the multi-hop reachability to different communities, which enhance the inter-community routing efficiency.

3.3.2.1 Calculating one hop encounter frequencies to different communities

Building IERT needs the information of encounter frequencies of vehicles with different communities. Therefore, each vehicle is responsible for maintaining the encounter frequencies with different communities in the IEter-community Frequency Table (IEFT) by Formula (3.1):

$$\begin{cases} FC_{ia}^{t} = fc_{ia}^{t} & t = 0, \\ FC_{ia}^{t} = \alpha FC_{ia}^{t-1} + (1-\alpha) fc_{ia}^{t} & else \end{cases} \tag{3.1}$$

where FC_{ia}^{t} denote the encounter frequencies between vehicle i and community a at time t. fc_{ia}^{t} denotes the encounter frequencies during the interval $[t-1, t)$ (the number of times that vehicle i

encounters the vehicles in community a). α is a damping factor where the higher value of α is, the more FC_{ia}^t count the recent encounters.

3.3.2.2 CommunityRank algorithm

The personalized CommunityRank value is given by Formula (3.2):

$$\begin{cases} CR_{ij}^t = FC_{ij}^t & t = 0, \\ CR_{ia}^t = (1-\beta) + \beta \sum\limits_{k \in E_i^t} \frac{FC_{ki}^t CR_{ka}^t}{\sum\limits_{l \in C} FC_{kl}^t} & else \end{cases} \tag{3.2}$$

where CR_{ij}^t denotes the CommunityRank from vehicle i to vehicle j at time t; E_i^t denotes the vehicle set that vehicle i encountered during time interval $[t-1, t)$ which belong to the same community with vehicle i; C denotes the community which vehicle i belongs to and β is the damping factor. Whenever two vehicles encounter each other, they exchange their current IERT and IEFT tables. Then, the two vehicles update their CommunityRank values using Formula (3.2). Implicitly, CommunityRank algorithm exploits the mobility and contact behavior of vehicles since the CommunityRank values are updated every time when vehicles encounter each other.

3.3.3 Building IntrA-community Personalized Rank Table (IART)

In the intra-community routing phase, in order to forward each copy to the vehicles which are close to the destination vehicles, we need to define another criterion for selecting relay vehicles. Similar as the personalized CommunityRank algorithm, we design the personalized VehicleRank algorithm and build IntrA-community personalized Rank Table (IART) for each vehicle dynamically. IART records the reachability of a vehicle to different vehicles (called VehicleRank) in the same community. The VehicleRank can count the multi-hop reachability to different vehicles, which enhances the intra-community routing efficiency.

3.3.3.1 Calculating one hop encounter frequencies to vehicles in the same community

Building IART needs the information of encounter frequencies of vehicles with other vehicles in the same community. Therefore, each vehicle is also responsible for maintaining the encounter frequencies with other vehicles in the same community in the IntrA-community Frequency Table

(IAFT) by Formula (3.3):

$$\begin{cases} FV_{ij}^{t} = fv_{ij}^{t} & t = 0, \\ FV_{ij}^{t} = \gamma FV_{ij}^{t-1} + (1-\gamma) fv_{ij}^{t} & else \end{cases} \tag{3.3}$$

where FV_{ij}^{t} denotes the encounter frequencies between vehicle i and j at time t; fv_{ij}^{t} denotes the encounter frequencies during time interval $[t-1, t)$ (the number of times that vehicle i encounters vehicle j). γ is a damping factor where the higher value γ is, the more FV_{ij}^{t} counts the recent encounters. The IAFT table will be applied for the VehicleRank algorithm later. Since the communities are divided based on the encounter frequencies, vehicles in different communities usually have very low encounter frequencies with each other. Therefore, we just discard the encounter frequencies among vehicles in different communities for saving memory.

3.3.3.2 VehicleRank algorithm

As shown in Figure 3.3, Figure 3.4 and Figure 3.6, VNETs are dominated by some important vehicles which can guarantee the connectivity of the network. Therefore, we hope the relay vehicles have not only high probability to reach destination vehicles, but also high probability to reach the other important vehicles which can reach destination vehicles with high probability. Centrality based routing algorithms [33,53] lack the capability to measure the vehicle reachability to different vehicles. Therefore, we design a personalized VehicleRank algorithm which is inspired by the personalized PageRank algorithm [31] to calculate the multi-hop reachability of vehicles to different vehicles in the same community dynamically. Consequently, the personalized VehicleRank value is given by Formula (3.4):

$$\begin{cases} VR_{ij}^{t} = FV_{ij}^{t} & t = 0, \\ VR_{ij}^{t} = (1-d) + d \sum\limits_{k \in E_i^t} \frac{FV_{ki}^{t} VR_{kj}^{t}}{\sum\limits_{l \in C} FV_{kl}^{t}} & else \end{cases} \tag{3.4}$$

where VR_{ij}^{t} denotes the VehicleRank from vehicle i to vehicle j at time t and d is the damping factor. Whenever two vehicles encounter each other, first they check whether they belong to the same community. If they belong to the same community, they exchange their current IART and IAFT tables. Then, the two vehicles update their VehicleRank values using Formula (3.4). Implicitly, VehicleRank algorithm also exploits the mobility and contact behavior of vehicles since the VehicleRank values are updated every time that vehicles encounter each other.

3.3.4 Weak Tie based Multi-copy Routing Algorithm

In Chapter 1, we have introduced the concept of weak ties and its important role in information spreading. Recall that we define the infrequent contacts in VDTNs as weak ties. To be more specific, we further define the contact between vehicles from two different communities as a weak tie (which is usually with a low encounter frequency) and design our weak tie based multi-copy routing algorithm. In the weak tie based multi-copy routing algorithm, the different copies of each packet are first scattered to different communities in a VNET through weak ties. Then different copies can search the destination vehicle through different weak ties. Therefore, we can improve the multi-copy routing efficiency since there is a high probability that at least one weak tie is connected during the routing time. For example, as shown in Figure 1.1, we scatter copies of a packet to communities which are close to the destination community. Then, different copies search the destination community through different weak ties (A, B, C, D) simultaneously. Finally, one of the copy encounters a connected weak tie D to destination community, which improves the multi-copy routing efficiency.

Besides the problem of the low utilization of different copies of each packet mentioned in Chapter 1, the multi-copy routing algorithms which rely on source nodes for replicating are overdependent on the capability of source nodes themselves, though the capabilities of source nodes themselves are various as shown in Figure 3.3. The source node may barely encounter any suitable nodes for replicating. Therefore, besides the basic weak tie multi-copy routing idea mentioned above, we hope to improve the efficiency of scattering copies by replicating copies from different relay vehicles simultaneously. However, there are challenges to control the number of copies per packet, since the frequent communications among vehicles are too expensive. How can we control the number of copies? How can we properly replicate copies of each packet to different communities? In order to handle the challenges, we use the FCT and develop the following detailed algorithm.

3.3.4.1 Basic concepts

Before we describe the detailed multi-copy routing algorithm, we first introduce three concepts: multi-copy community scope, multi-copy capability and multi-copy community scope splitting. Multi-copy community scope is used to identify the communities where different relay vehicles can scatter copies. Multi-copy capability is used as a criterion for selecting vehicles with stronger capability for scattering copies in the same community. Multi-copy community scope splitting is

used to separate and distribute the multi-copy community scope to vehicles in different communities in order to accelerate the scattering. The details are as follows.

Multi-copy community scope identifies the communities that a vehicle can replicate copies to, which is stored in the FCT. As shown in Figure 3.9 for example, the tree like FCT saves the communities the vehicle i can replicate copies to in gray hierarchically. The striped community presents the community that i belongs to.

Multi-copy capability presents the capability of a vehicle to replicate copies to a specific multi-copy community scope, which is calculated by Formula (3.5):

$$MCC(i,S) = \sum_{a \in S} CR_{ia}^{t} \tag{3.5}$$

where $MCC(i,S)$ is the multi-copy capability of vehicle i to multi-copy community scope S.

Multi-copy community scope splitting is that given a multi-copy community scope S and two vehicles i and j, we split the multi-copy community scope according to the distances of each community in the multi-copy community scope and the communities vehicle i and j belong to by Formula (3.6):

$$\begin{cases} S_i = \{\forall a \in S | level(i,a) > level(j,a)\} \\ S_{j=} S \backslash S_i \end{cases} \tag{3.6}$$

where S_i is the split multi-copy community scope for vehicle i; $level(i,a)$ are the levels of the lowest level community which contains the community a and the community that vehicle i belongs to. For example, as shown in Figure 3.5, $level(a,b) = 2$ and $level(a,c) = 1$.

3.3.4.2 Detailed multi-copy routing process

Based on these concepts, we introduce our weak tie based multi-copy routing algorithm. Initially, when the packet is produced by source vehicle s, s is authorized with the multi-copy community scope of all the communities. Then the weak tie multi-copy routing algorithm works as follows:

1. Once a relay vehicle i encounters another vehicle j without the same packet, we check whether vehicle i and j belong to the same community. If yes, go to step 2). Otherwise, go to step 3);

2. we calculate the multi-copy capabilities of i and j to the multi-copy community scope S stored in vehicle i by Formula (3.5). If $MCC(j,S) > MCC(i,S)$, vehicle i forwards its packet with its multi-copy community scope to j. Therefore, the packet and its multi-copy community

scope will be forwarded to the vehicles with higher multi-hop reachability to the corresponding multi-copy community scope;

3. vehicle i checks whether vehicle j's community belongs to vehicle i's multi-copy community scope. If yes, vehicle i splits the multi-copy community scope by Formula (3.6), replicates a copy with multi-copy community scope S_j to vehicle j and updates its own multi-copy community scope by S_i. Therefore, the multi-copy community scope is carefully split and allocated to the communities which are closer to the multi-copy community scope.

A simple example of multi-copy community scope splitting process is shown in Figure 3.11. When the relay vehicle i encounters another vehicle j, the multi-copy community scope is split according to the distances between communities that i and j belong to and the multi-copy community scope. In Figure 3.11, the black nodes present the multi-copy community scope and the striped nodes present communities that i and j belong to.

3.3.4.3 Manage the number of copies

Algorithm 4: Detailed Process of SPread

$\forall k, VR_{ik_0} \leftarrow FV_{ik_0}$;
while $S \neq S_p$ **do**
  $S_p \leftarrow S$;
  $S_c \leftarrow \{i \in S | deg_S(i) \leq \rho_{max}\}$;
  $S \leftarrow S \backslash S_c$;
  if $\rho(S) > \rho_{max}$ **then**
    $\rho_{max} \leftarrow \rho(S)$;
  end
end
$G_0 = (S_0, E(S_0)) \leftarrow G_S = (S, E(S))$;
$G_1 = (V_1, E(V_1))$ of G_S;
G_1 to D_{list};
${V_1}^* \leftarrow V_1 \cup \{j | (i,j) \in E(S), i \in V_1\}$;

A large number of copies may lead to a congestion in the network and meanwhile, the copies located in the communities which are far from the destination community have little chance to reach the destination community. Therefore, instead of initializing the multi-copy community scope to all the communities, we only initialize the multi-copy community scope to the communities which are close to the destination community. To be more specific, we define the initial multi-copy community scope S_{init} by Formula (4.1):

$$S_{init} = \{\forall a \in S | level(t, a) > r\} \tag{3.7}$$

where S is the set of the communities at the lowest level and r is a threshold for managing the number of copies of a packet.

3.3.5 Detailed Process of SPread

Based on the above descriptions, the detailed process of SPread is shown in Algorithm 4, where $packet(k)$ is the kth packet in vehicle i, $c(packet(k))$ is the destination community of $packet(k)$, $c(i)$ is the community that vehicle i belongs to, $MCC(i, S_i)$ is the multi-copy capability of i on multi-copy community scope S_i, $t(packet(k))$ is the destination vehicle of $packet(k)$ and $VR(j, t(packet(k)))$ is the VehicleRank of j to $t(packet(k))$. At the beginning of the routing, $packet(k)$ is produced by the source vehicle and the multi-copy community scope is calculated by a threshold r for controlling the number of copies. Suppose a relay vehicle i of $packet(k)$ encounters another vehicle j, then we have:

1. If $c(j) \neq c(packet(k))$ and $c(j) \in S_i$, $packet(k)$ is replicated from i to j and the multi-copy community scope for i is split and distributed to j by Formula (3.6);

2. If $c(j) \neq c(packet(k))$, $c(i) = c(j)$ and $MCC(j, S_i) > MCC(i, S_i)$, $packet(k)$ is forwarded to j with its multi-copy community scope S_i;

3. If $c(j) = c(packet(k))$, $c(i) \neq c(j)$ and $S_i = \emptyset$, $packet(k)$ is forwarded to j with its multi-copy community scope S_i;

4. If $c(j) = c(packet(k))$, $c(i) \neq c(j)$ and $S_i \neq \emptyset$, $packet(k)$ is replicated from i to j and the multi-copy community scope for i is split and distributed to j by Formula (3.6);

5. If $c(i) = c(j) = c(packet(k))$ and $VR(j, t(packet(k)))$ is larger than $VR(i, t(packet(k)))$, $packet(k)$ is forwarded to j with its multi-copy community scope S_i.

3.4 Advanced SPread

In SPread, we translate the vehicles to a contact graph by their contact information, and then classify the contact graph to hierarchical communities by a general community identification algorithm. However, such a strategy may cause some problems. First, as the previous contact based

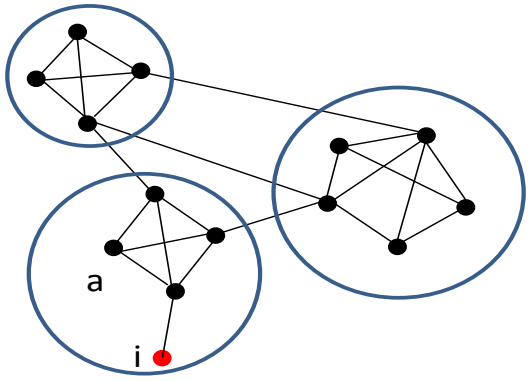
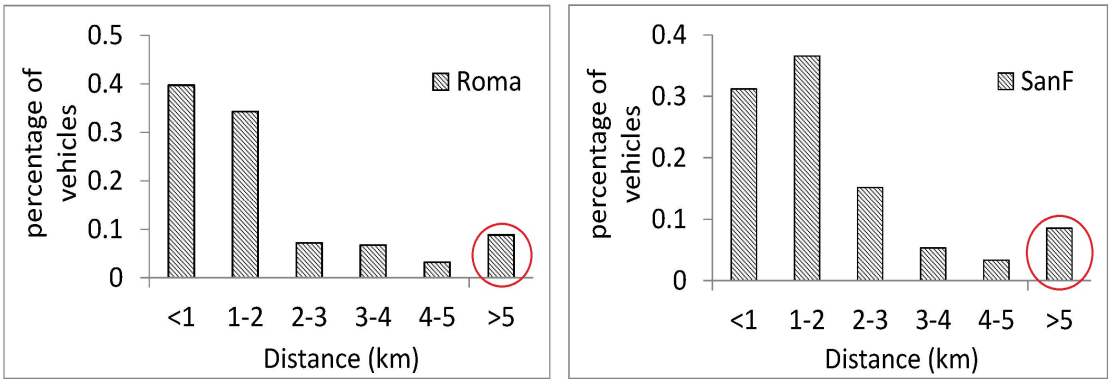


Figure 3.12: An example of the community identified by a general community discovery method.

routing algorithms, the packet delivery in SPread is still based on passively waiting for the next suitable relay vehicle instead of actively searching the target vehicle, which influences the performance of SPread since the trajectories of vehicles are not predictable. Second, the general community identification algorithm fails to consider some useful information (e.g., the relationship between contact locations and contact frequencies) that can help decrease the success rate and average delay of the routing. Finally, SPread evenly distributes one copy to each community. However, it may lead to a imbalance of utilization of copies since communities discovered by the general community identification algorithms are with different sizes. Therefore, in this section, we further exploit the spatio-contact correlation of the community to improve the efficiency of the basic SPread and propose an Advanced SPread (ASPread). The spatio-contact correlation means that the vehicles in the same community tend to meet each other in a certain small geographical area comparing to the whole VNET map.

The general community discovery method classifies each vehicle to a community. However, in reality, not all the vehicles are very related even though they are in the same community. For example as shown in Figure 3.12, vehicle i belongs to the community a only because vehicle i is a little more close to community a than all the other communities. But in fact, if vehicle i is selected as a relay vehicle for community a, the routing performance will be decreased since there is only one link from vehicle i to the rest vehicles in community a. Therefore, if a vehicle that is not very related to its own community is selected, the routing performance may be adversely influenced. We define *core vehicles* as the vehicles that frequently visit the locations that the majority of the vehicles in a community visit frequently, such as the black nodes in Figure 3.12. By taking advantage of the relationship between communities and geographic locations, we improve the routing performance by



(a) *Roma* (b) *SanF*

Figure 3.13: The trajectory of a vehicle in the entire VDTN map.

identifying the core vehicles in the communities and try to deliver the packets to the core vehicles in the communities.

3.4.1 Measuring Spatio-contact Correlation of the Community

In order to verify the spatio-community correlation of vehicles, firstly, we define the concepts of the *road section* and the *active road section*. A *road section* is the road part that does not contain any intersections and it is denoted by the two IDs of intersections on its two ends in the entire VDTN map. Then, we define the *active road sections* of vehicle v as road sections where vehicle v visits frequently. To be more specific, we define the set of active road sections of vehicle j (denoted by S_j) by:

$$S_j = \{\forall s \in S | f(s, j) > rv\} \tag{3.8}$$

where S_j is the set of active road sections of vehicle j, $f(s, j)$ is the frequency that vehicle j visits road section s; and rv is a visit frequency threshold. A smaller threshold rv leads to more road sections in the S_j and vice versa.

We set r to 4 and find the active road sections of each vehicle in the *Roma* and *SanF* traces. Figure 3.13 shows the distributions of the average distances of the active road sections between vehicles and all the other vehicles in the same community in the *Roma* and *SanF* traces. As shown in Figure 3.13, the active road sections of most vehicles in the same community have average distances less than 2km. Since previous community is defined based on contact but not location, it is possible that the contact location of the vehicles in the community is on the whole map. Comparing to the diameters of the whole maps which are approximately 100km, 2km is a very

small distance and hence, we can restrain the searching of copies of packets to a small geographic area. Therefore, this phenomenon verifies that most vehicles in the same community tend to have close active road sections, and there are high correlation between the location and the community. While at the same time, as shown in the red circles, there exist some vehicles whose active road sections far away from most vehicles in the same community. Therefore, we conclude our third observation (**O3**) and fourth observation (**O4**) as follows:

O3: *Most vehicles in the same community tend to be active in the same location.*

O4: *There exist some vehicles whose active road sections far away from most vehicles in the same community.*

Based on this observation, we further enhance the routing efficiency of SPread. We present the enhanced routing algorithm in the following.

3.4.2 Mapping Communities to Geographic Locations

Based on **O3** that vehicles in the same community tend to be active in the same location, we try to map different communities to different geographic locations so that it becomes easier to find the target vehicle's community.

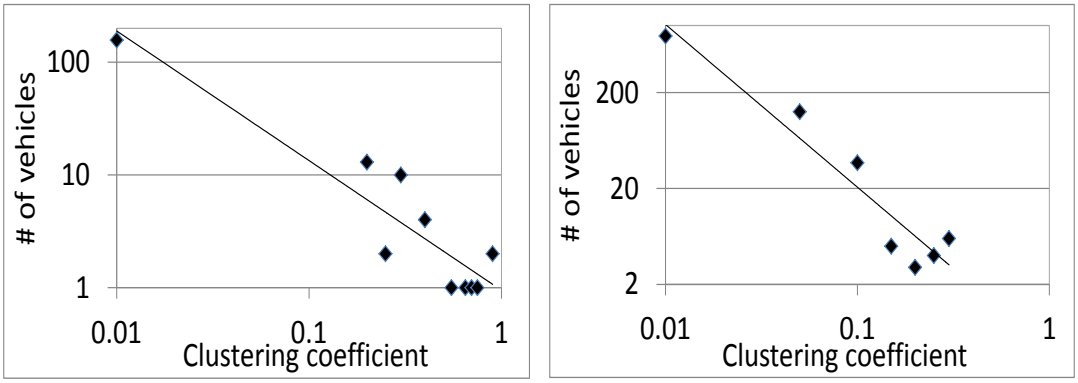
Based on the definition of active road sections of vehicles, we define the set of active road sections of community a (denoted by S_a) by:

$$S_a = \{\forall s \in S | \sum_{i \in V} f(s,i) > rc\} \tag{3.9}$$

where S_a is the set of active road sections of community a, V is the set of vehicles in community a and rc is a visit frequency threshold for communities. A smaller threshold rc leads to more road sections in the S_a and vice versa.

However, it is not possible to calculate S_a in a distributed manner since it is unlike for a vehicle to collect $f(s,i)$ for each $i \in V$. Therefore, we let each vehicle i calculate the set of active road sections of community a (denoted by S_{a_i}) distributedly. We use $f(s, S_{a_i})$ to denote the visiting frequency on road section s in S_{a_i} at current time collected by vehicle i in the distributed manner. It is calculated by:

$$f(s, S_{a_i}) = \begin{cases} 0 & \text{if } \textit{initialize} \\ \max(f(s, S_{a_i}) + f(s,j), f(s, S_{a_j})) & \text{if } \textit{meet} \text{ j} \end{cases} \tag{3.10}$$



(a) *Roma* (b) *SanF*

Figure 3.14: The community size distribution.

When vehicle i meets vehicle j which belongs to community a, vehicle i updates $f(s, S_{a_i})$. $f(s, S_{a_i})+$ $f(s, j)$ means that the visiting frequency of road section s in S_{a_i} is incremented by the visiting frequency of vehicle j on road section s. Vehicle j also maintains $f(s, S_{a_j})$. Then, vehicle i chooses the higher visiting frequency value as its updated $f(s, S_{a_i})$. Vehicle i determines the active road sections of community a by:

$$S_{a_i} = \{\forall s \in S | f(s, S_{a_i}) > rc\} \tag{3.11}$$

Based on Formula (3.11), each vehicle i distributedly calculates the set of active road sections of each community. When vehicle i meets vehicle j, if vehicle j belongs to the community of the destination (say community a) and its active road sections, S_j (calculated by Formula (3.8)), belongs to S_{a_i}, then vehicle j is a core vehicle of community a. In this case, vehicle i forwards its packet to vehicle j, which has a high probability of meeting the target vehicle.

3.4.3 Adaptive Determination of the Number of Copies based on Community Size

Another problem of SPread is that copies are evenly distributed to each community in SPread. However, previous studies [25] show that the community size follows a power law distribution. In order to verify it in VNET, we draw Figure 3.14 which shows the distribution of numbers of communities with different sizes. As shown in Figure 3.14, the community size also follows a power law distribution in VNET, which is consistent with previous studies. However, in SPread, we equally distribute one copy to each community. This strategy can lead to an imbalance of the utilization of copies which means that some of the copies are responsible for searching among many

Road section ID	Frequency
1	0
2	0
3	2
...	...

Figure 3.15: An example of the active road section table.

vehicles, while some other copies are responsible for searching among only a few vehicles. Therefore, in this section, we calculate the number of copies of community a by:

$$N_a = \frac{N \cdot |V_a|}{|V|} \tag{3.12}$$

where N_a is the number of copies which can be sent to community a, N is the total number of copies of a packet that can be sent, V_a is the set of vehicles in community a and V is the set of all the vehicles in the entire network.

In this way, we can make sure that the communities with larger sizes are allocated with more copies and the communities with smaller sizes are allocated with fewer copies. Therefore, we can avoid the situation in which some copies search among too many vehicles, while some other copies search among only a few vehicles. As a result, the tradeoff between the routing overhead and routing efficiency can be better achieved.

3.4.4 Routing Algorithm of Advanced SPread

In ASPread, a packet copy arrives at an active sub-area of the target vehicle by the same method used in SPread. Then, ASPread forwards the packet copy to the core vehicles in the community that are more likely to meet the destination node. For this purpose, each node additionally maintains two tables: active road section table and community-location mapping table. A vehicle's active road section table records its visiting frequencies on its active road sections as shown in Figure 4.16, where 1, 2, 3,... denote road IDs. The active road section table is updated by recording the vehicle's visited road sections and its visiting frequency (i.e., the number of visits in a certain time period T). Then, based on Formula (3.8), each vehicle determines its active road sections and updates its active road section table periodically. To be more specific, for a vehicle i, it updates its active road section table as follows:

1. Vehicle i initializes all the road sections with frequency 0;

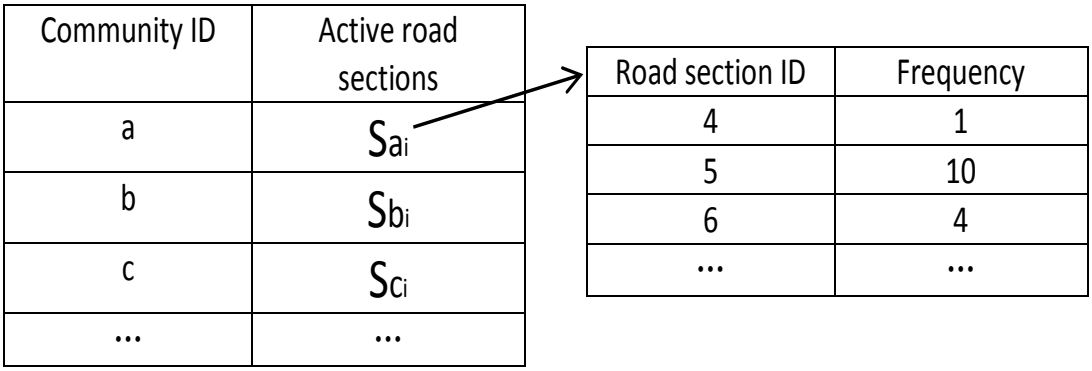


Figure 3.16: An example of the community-location mapping table.

2. Once vehicle i reaches a road section, it increases the visiting frequency of the road section by 1 and go to Step 3);

3. If the frequency of the road section is larger than the frequency threshold rv, add the road section to vehicle i's active road section table (Formula (3.8)).

Each vehicle also stores a community-location mapping table as shown in Figure 3.16. In the table, a, b and c denote community IDs, and S_{a_i}, S_{b_i} and S_{c_i} denote the set of active road sections of community a, b and c, respectively, calculated by Formula (3.11). The arrow means that S_{a_i} includes active road sections 4, 5 and 6 and their visiting frequencies. These frequencies are used to find the active road sections of a community based on Formula (3.11).

For vehicle i, it updates its community-location mapping table as follows:

1. Vehicle i initializes its own community-location mapping table by setting the active road section sets of all communities to empty sets;

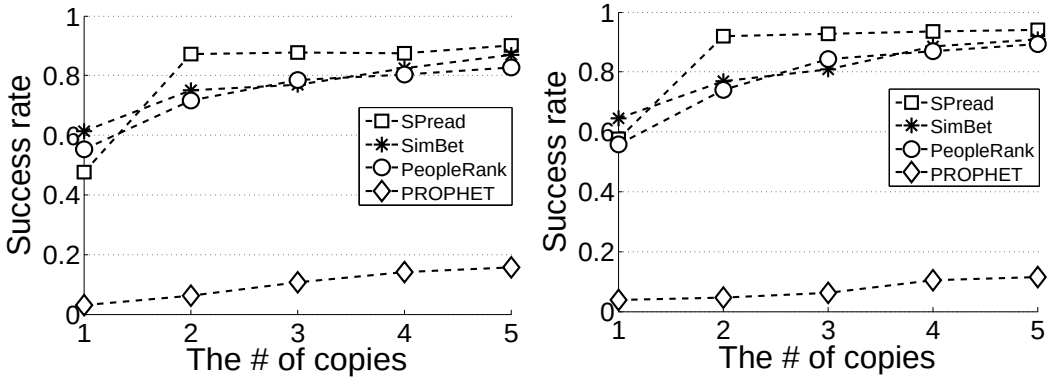
2. Once vehicle i meets another vehicle j, vehicle i gets the active road section table of vehicle j and the community-location mapping table of vehicle j;

3. Based on the active road section table of vehicle j, vehicle i recalculates the visiting frequency of road section s in community $c(j)$ (denoted by $f(s, S_{c(j)_i})$) by Formula (3.10);

4. If the frequency of a road section is larger than the frequency threshold rc, vehicle i adds the road section to community $c(j)$'s active road section set in the community-location mapping table by Formula (3.11).

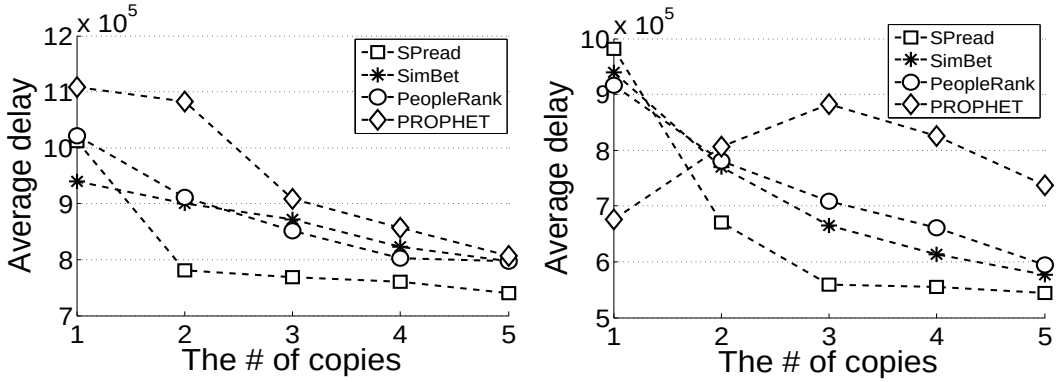
Algorithm 5: Detailed Process of ASPread

```
1: ∀k, VR_{ik_0} ← FV_{ik_0};
2: if vehicle i meets vehicle j then
   |  if j ∈ S_{c(j)_i} then
   |  |  Call Algorithm 4;
   |  end
   |  if vehicle i is the first relay vehicle in c(i) then
   |  |  while CopyNum(i) > 0 and c(i) == c(j) do
   |  |  |  copy packet(k) to j;
   |  |  |  CopyNum(i) − −;
   |  |  end
   |  end
   |  end
```



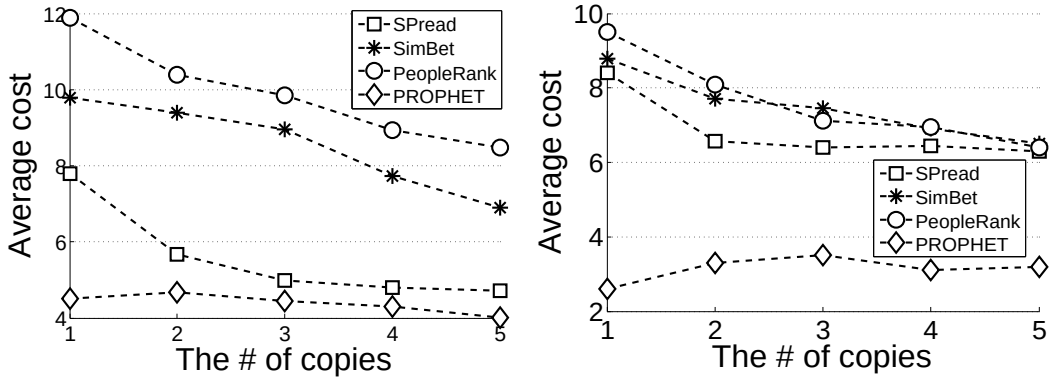
(a) *Roma* (b) *SanF*

Figure 3.17: The success rate vs. different number of copies.



(a) *Roma* (b) *SanF*

Figure 3.18: The average delay vs. different number of copies.



(a) *Roma* (b) *SanF*

Figure 3.19: The average cost vs. different number of copies.

When a relay vehicle i meets another vehicle, say vehicle j, if the destination community is in vehicle j's multi-copy community scope, vehicle i further checks if vehicle j is a core vehicle in the destination community. To do this, vehicle i refers to its own community-location mapping table, S_{a_i}, and vehicle j's active road section table S_j. If $S_j \in S_{a_i}$, vehicle j is a core vehicle in the destination community and vehicle i forwards the packet to vehicle j.

Based on the above description, the detailed process of ASPread is shown in Algorithm 5. In the algorithm pseudocode, $packet(k)$ is the kth packet in vehicle i, $c(i)$ is the community that vehicle i belongs to and $CopyNum(i)$ is the number of copies vehicle i can create in community $c(i)$. At the beginning of the routing, $packet(k)$ is produced by the source vehicle and the multi-copy community scope is calculated by a threshold r and the number of copies of each community is calculated by Formula (3.12). Different from SPread, ASPread distributes multiple copies to a community and the number of copies is determined by the community size. Also, we only select core vehicles in the destination communities as relay vehicles in ASPread instead of any vehicles in the destination communities. Suppose a relay vehicle i of $packet(k)$ encounters another vehicle j, then the following steps are executed in the routing algorithm based on SPread:

1. If vehicle j belongs to the destination community of $packet(k)$, go to Step 2);

2. Vehicle i checks whether vehicle j is the core of its own community. If yes, vehicle i follows the same steps of SPread; otherwise, vehicle i ignores vehicle j directly without taking any action;

3. In order to allocate different numbers of copies based on different sizes of communities as we mentioned above, if relay vehicle i is the first relay vehicle in its own community, vehicle i is

responsible for distributing the specified number of copies in its own community calculated by Formula (3.12) and go to Step 4);

4. Every time when the relay vehicle i meets another core vehicle in its community, it replicates a copy to the vehicle and decreases the remaining number of copies by 1 until the remaining number of copies equals to 0.

Based on this process, we can guarantee that the packet can be efficiently relayed by considering not only the contact distances between different communities, but also the geographic distances between the relay vehicles and target communities. Also, the community with a larger size is allocated with more copies to make sure that the target vehicle can be more efficiently searched.

3.5 Performance Evaluation

3.5.1 Performance Evaluation and Analysis of Massive Natural Graphs

Recall that the precise algorithm [62] (denoted by *LocPreAlg*) that finds the densest subgraph containing a specific vertex subset and the approximate algorithms [4,11] (denoted by *GloAppxAlg1* and *GloAppxAlg2*) that find the densest subgraph neglect the connectivity of the returned subgraphs. Although the previous precise algorithm [26] (denoted by *GloPreAlg*) can guarantee the connectivity of the returned subgraph, it is at the cost of high time and memory complexity. Also, no previous work can find GSDSs. In this section, we conduct experiments to show the effectiveness and efficiency of our proposed LSDS (Algorithm 1) and GSDS (Algorithm 3) algorithms in solving these problems in comparison with these previous algorithms. We also show the enhanced efficiency of our improved GSDS algorithm (Algorithm 3). We use two groups of massive datasets in Table 3.1 and Table 3.2 from different domains in our experiments. We use big datasets in Table 3.2 to test the capacity of manipulating big data of Algorithm 3. The algorithms are implemented by C language. The testing platform is one PC with 2.1GHz Intel core i3 processor with 2 cores, and a 4GB memory. The Operating System is Ubuntu 10.0. In all the following figures, we use the results of our algorithms as a baseline and shows the ratios of the results of other algorithms to our algorithms. Further, we use Algorithm 3 to extract the GSDSs of the massive natural graphs from social networks, technology, chemistry, and biology and conduct simple analysis to reveal the physical significance of GSDSs in natural graphs from different domains.

Table 3.1: Description of the datasets from different domains

ID	Description	Domain
Dataset 1 [43]	Collaboration network	Social network
Dataset 2 [42]	Facebook	Social network
Dataset 3 [72]	Power Grid	Technology
Dataset 4 [35]	Protein interaction	Chemistry
Dataset 5 [28]	E-mail interchanges	Information
Dataset 6 [20]	Metabolic network	Biology

Table 3.2: Description of the big datasets

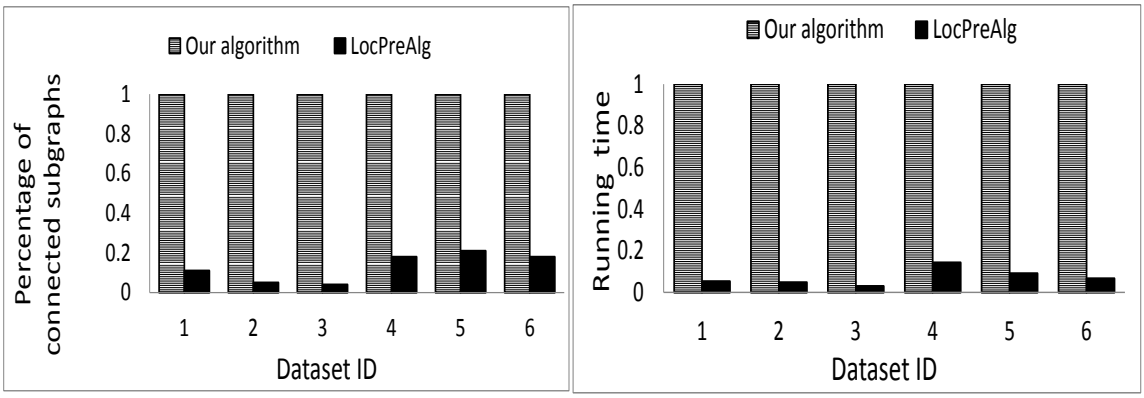
ID	Description	Domain
Dataset 7 [75]	LiveJournal	Social network
Dataset 8 [75]	Orkut	Social network

3.5.1.1 Performance Evaluation of the LSDS Algorithm

We randomly select a specific vertex subset from each dataset, and then use Algorithm 1 and *LocPreAlg* to find the LCDS of the selected vertex subset. Since both algorithms are not suitable for large datasets, we conduct experiments on datasets in Table 3.1. We repeated the experiment on each dataset for 100 times and calculated the percentage of connected subgraphs in the 100 returned subgraphs.

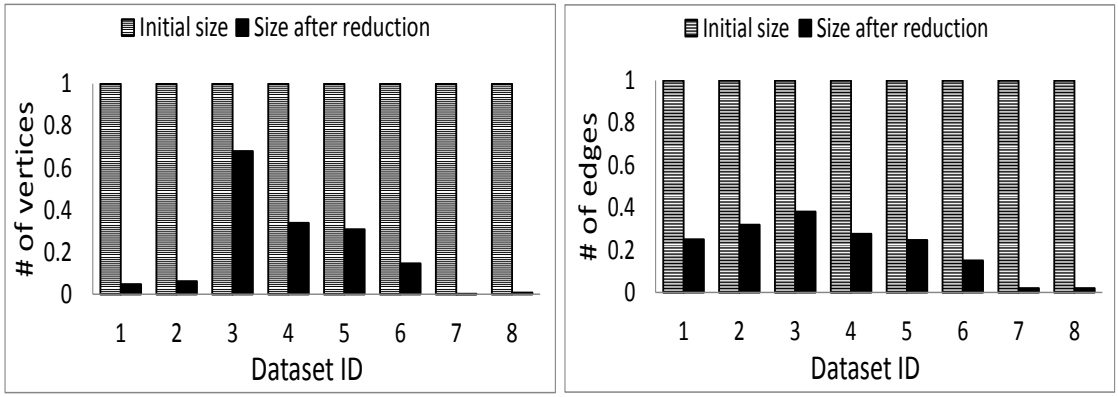
Figure 3.20(a) shows ratio of the percentage of connected subgraphs in *LocPreAlg* compared to Algorithm 1. For the previous algorithm, only 13% (for average) of the outcomes are connected, while for Algorithm 1, 100% of the outcomes are connected. We further analyze the 13% of the connected outcomes of the previous algorithm. We find that it is connected just because the specific vertex subset only contains one vertex, which is contained in the densest subgraph of the initial graph. For other outcomes of *LocPreAlg*, they are disconnected in order to increase the density. This results show that Algorithm 1 can guarantee the connectivity of the returned graphs for the LCDS problem.

Since Algorithm 1 and *LocPreAlg* both require to store all the data into the memory, the memory usages for both algorithms are same. Figure 3.20(b) shows the ratio of the time used by *LocPreAlg* compared to Algorithm 1. We see that *LocPreAlg* is far more efficient than Algorithm 1. The reason is because that we apply a simple min-cut algorithm [66] for solving the TCPM problem with time complexity $O(|V||E| + |V|^2 log|V|)$, while *LocPreAlg* applied a push relabeled algorithm [15] for solving the min-cut max-flow problem with time complexity $O(|V||E|log(|V|^2/|E|))$. This is the cost to guarantee the connectivity of the returned subgraphs. We will improve the efficiency of Algorithm 1 in our future work.

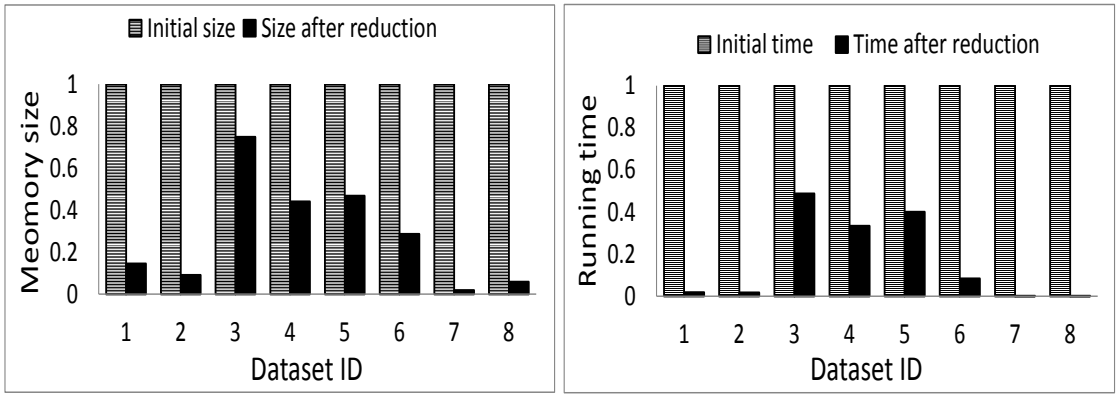


(a) Discovering connected subgraphs (b) The running time

Figure 3.20: The performances of the LCDS algorithm compared to previous algorithms



(a) The number of vertices (b) The number of edges



(c) The memory size (d) The running time

Figure 3.21: The performance of the improved GSDS algorithm compared to the basic GSDS algorithm

Table 3.3: Comparison of datasets before and after reduction

Datasets	# of vertices		# of edges		size (KB)	
	Before	After	Before	After	Before	After
Dataset 1	5,242	244	28,980	7,238	344	50
Dataset 2	9,877	599	25,998	8,264	644	59
Dataset 3	4,941	3,353	13,188	5,006	107	80
Dataset 4	1,846	624	4,406	1,121	34	15
Dataset 5	1,133	349	10,902	2,681	79	37
Dataset 6	453	66	2,066	301	14	4
Dataset 7	3,997,962	4,136	34,681,189	650,724	489,799	8,818
Dataset 8	3,072,441	20,723	117,185,083	2,087,932	1,728,293	100,352

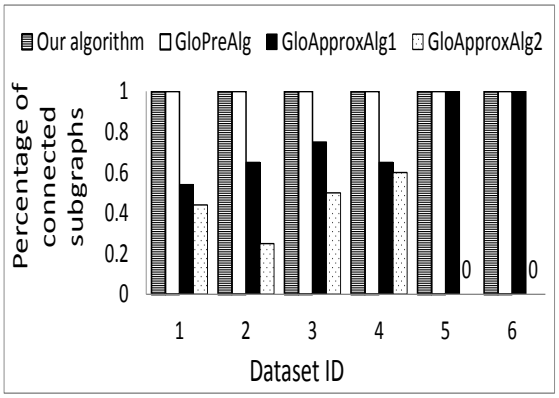
3.5.1.2 Comparison of Basic and Improved GSDS Algorithms

We then show the effectiveness of Algorithm 3 compared to Algorithm 2. Recall that Algorithm 3 removes unnecessary edges and vertices. Table 3.3 shows the comparisons of the number of vertices, the number of edges, and the memory size of the datasets before and after the reduction. In order to show the reduction performances clearly, we draw Figures 3.21(a), (b) and (c) that show the ratio of the number of vertices, the number of edges and the memory in Algorithm 3 compared to Algorithm 2. From the table and figures, we see that the reduction in Algorithm 3 is significant. Especially for the big dataset 7 and big dataset 8 which have a GB level data size, the number of vertices and edges, and memory occupation are reduced to about 2% (for average) of the initial size, which makes it possible to run the polynomial algorithm on one PC.

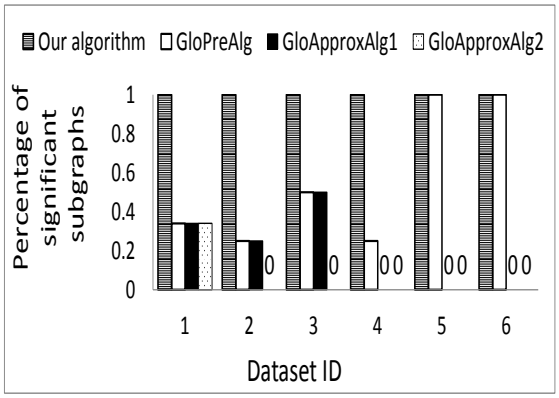
Figure 3.21(d) shows the percentage of running time of Algorithm 2 compared to Algorithm 3. Unsurprisingly, the running time of Algorithm 3 is much faster than Algorithm 2 since the data sizes after reduction are much smaller than the initial sizes due to the significant size reduction in Algorithm 3. Since dataset 7 and dataset 8 are too large to be computed by Algorithm 2, we only estimate the results based on the theoretical time complexity. We see the actual running time of Algorithm 3 are only 0.01% of the estimated running time of Algorithm 2.

3.5.1.3 Performance Evaluation of the Improved GSDS Algorithm

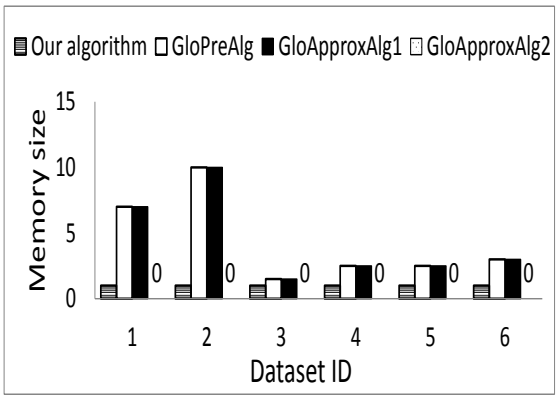
In this section, we compare the performance of discovering GSDSs between Algorithm 3 and *GloPreAlg*, *GloAppxAlg1* and *GloAppxAlg2*. First, we use Algorithm 3 to discover all the GSDSs. Then, we use *GloPreAlg*, *GloAppxAlg1* and *GloAppxAlg2* to find the same number of densest subgraphs by recursively running these algorithms in the remaining graph. Then, we check whether



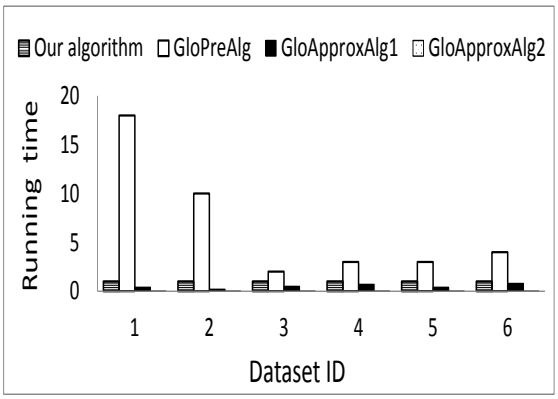
(a) Discovering connected subgraphs



(b) Discovering significant subgraphs



(c) The memory size



(d) The running time

Figure 3.22: The performance of the improved GSDS algorithm compared to previous algorithms

these dense subgraphs are connected or significant. We repeat 100 experiments on each dataset in Table 3.1.

Figure 3.22(a) shows the percentage of connected subgraphs of *GloAppxAlg1*, *GloAppxAlg2* and *GloPreAlg* compared to Algorithm 3. We find that for *GloAppxAlg1* and *GloAppxAlg2*, there are a lot of disconnected subgraphs in the results, while for Algorithm 3, all the output subgraphs are connected. Also, we find *GloAppxAlg1* performs a little better than *GloAppxAlg2* since *GloAppxAlg1* greedily searches the densest subgraph by deleting the vertices one by one, while *GloAppxAlg2* greedily searches the densest subgraph by deleting the vertices batch by batch. *GloPreAlg* does not have the connectivity problem since it is a precise algorithm. The results confirm that *GloAppxAlg1* and *GloAppxAlg2* neglect the connectivity problem and Algorithm 3 can solve it.

Figure 3.22(b) shows the percentage of significant subgraphs in *GloAppxAlg1*, *GloAppxAlg2* and *GloPreAlg* compared to Algorithm 3. We find that for *GloAppxAlg1*, *GloAppxAlg2* and *Glo-*

PreAlg, there are a lot of insignificant subgraphs in the results, while for Algorithm 3, all the output subgraphs are significant. Also, similar as the connectivity comparison results, we find *GloAppxAlg1* performs a little better than *GloAppxAlg2*. Although *GloPreAlg* is a precise algorithm and does not have the connectivity problem, it still cannot guarantee that its discovered subgraphs are significant. The results confirm that *GloAppxAlg1*, *GloAppxAlg2* and *GloPreAlg* cannot guarantee that the discovered subgraphs are significant and Algorithm 3 can solve it.

Figure 3.22(c) shows the ratios of memory sizes of *GloAppxAlg1*, *GloAppxAlg2* and *GloPreAlg* compared to Algorithm 3. We see that Algorithm 3 needs much smaller memory size than *GloPreAlg* and *GloAppxAlg1* since it significantly reduces the data size. *GloAppxAlg2* almost does not need any memory since it is a streaming algorithm. Figure 3.22(d) shows the ratio of running time of *GloAppxAlg1*, *GloAppxAlg2* and *GloPreAlg* compared to Algorithm 3. Algorithm 3 is much faster than *GloPreAlg*, while *GloAppxAlg1* and *GloAppxAlg2* are much faster than Algorithm 3. This is because Algorithm 3 is precise algorithm but *GloAppxAlg1* and *GloAppxAlg2* are approximate algorithms applying a greedy strategy. Although *GloAppxAlg1* is more time efficient and *GloAppxAlg2* is both more time and memory efficient than Algorithm 3, both the algorithms cannot guarantee that the returned subgraphs are connected and significant as shown previously. Algorithm 3 can handle these problems with the capability of handling big data.

3.5.1.4 Analysis on the Natural Graphs

When it comes to the physical significances, the physical significance of the LCDS is clear and been discussed [62]. Instead, we focus on revealing the physical significance of the GSDSs based on our limited information about the datasets.

Table 3.4 shows the actual number of GSDSs discovered by our algorithm. Datasets 1 and 2 are paper collaboration networks from the categories of gr-qc and hep-th in ArXiv, respectively. For the papers in each category, they have sub-categories. Table 3.5 shows the top-10 most frequent sub-categories of 1000 randomly selected papers in each of the categories gr-qc and hep-th, which correspond to datasets 1 and 2, respectively. We find that the number of GSDSs in datasets 1 and 2 can precisely reflect the number of the most frequent sub-categories (emphasized with bold type) of their corresponding categories. Dataset 3 is the power grid network of western states. Interestingly, there are just 31 major metropolitan areas in western states [9], which is very close to the number of GSDSs (28) in dataset 3. The protein network in dataset 4 has 13 GSDSs, while there should

Table 3.4: # of GSDSs

ID	# of GSDS (Actual)	ID	# of GSDS
Dataset 1	3	Dataset 5	1
Dataset 2	4	Dataset 6	1
Dataset 3	28	Dataset 7	2
Dataset 4	13	Dataset 8	4

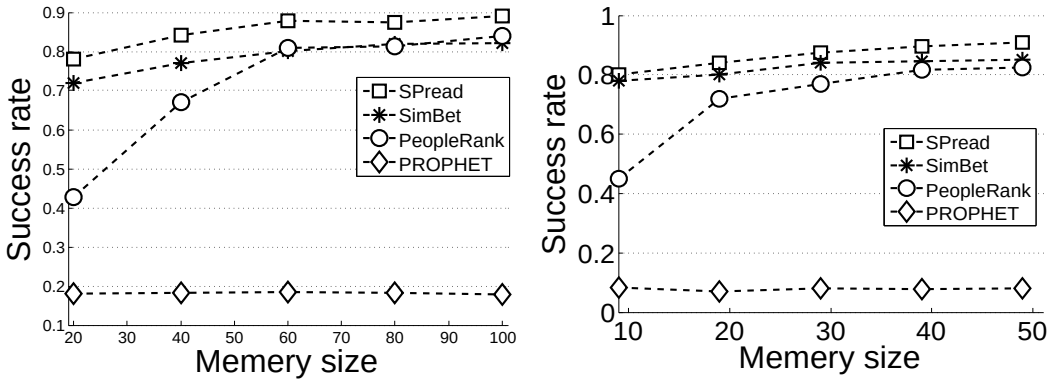
Table 3.5: Top-10 sub-categories in Dataset 1 and 2

Dataset 1		Dataset 2	
Sub-category	Frequency	Sub-category	Frequency
hep-th	**327**	**hep-ph**	**184**
astro-ph	**304**	**gr-qc**	**167**
math-ph	**175**	**math-ph**	**91**
quant-ph	31	**astro-ph**	**85**
cond-mat	28	hep-lat	39
physics.atom-ph	27	math.DG	35
stat-mech	11	nlin.SI	21
nucl-th	7	nucl-th	19

be hundreds of functional modules in the network [13]. The massive functional modules are highly clustered into several GSDSs separately. We are interested in the reasons behind such a clustering of functional modules to different GSDSs. The email network (dataset 5) from University Rovira i Virgili and metabolic network (dataset 6) from C.elegans both only have 1 GSDS, while there are 13 and 10 modularities [20] in each of the datasets, respectively. which means they are highly centralized to single important circle (e.g. a university or a simple organism). For the big datasets 7 and 8, we lack the information to reveal the physical significance of GSDSs.

Massively further information and more comprehensive analysis are needed for deeper understanding of the physical significance of GSDSs in different domains. However, from the simple analysis above, we find that the GSDS problem is different from traditional community detecting in that GSDS problem naturally ignore the unimportant parts and focus on the important parts of the graph (while the concepts of important parts may depend on the specific knowledge from the corresponding domains). On the contrary, the traditional community detecting methods [26, 55, 69] (e.g. modularity) return all the subgraphs and lack the capacity to purify the significant parts. Therefore, GSDS can be treated as a good complement in real applications when there is a desire to focus on the core parts of the graph. For example, we can apply GSDS to detect important academic circles in a research field, major metropolitan areas of a country, important functional module groups in a protein interaction network and so on.

To sum up, in this section, we revealed two problems existing in previous studies, which are



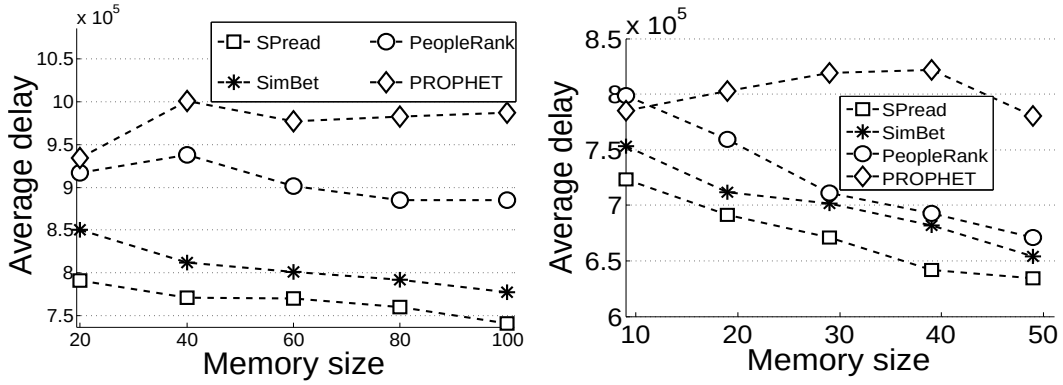
(a) *Roma* (b) *SanF*

Figure 3.23: The success rate vs. different memory size.

important in various applications in different domains. One problem is the neglect of the connectivity of returned subgraphs in previous approximate and precise algorithms for finding the densest subgraph without and with the restriction of containing a vertex subset, respectively. The other problem is the lack of an algorithm for finding multiple connected and significant dense subgraphs. To handle these problems, we defined two subproblems of discovering dense subgraphs: the LCDS and GSDS problems, and proposed algorithms to solve the problems in polynomial time. Also, based on the feature of natural graphs, we provided an improved algorithm to reduce the time and space complexity of the basic GSDS algorithm, which can easily handle data with GB-level size in one PC. In the experiments, we applied our algorithms on massive natural graphs, evaluated the efficiencies of our algorithms in comparison with previous algorithms, and analyzed the structure of these natural graphs. The experimental results showed the high effectiveness and efficiency of our algorithms in solving the problems. In the future, we will focus on improving the time complexity of the algorithms. Also, we will find more evidences to reveal the physical significance of GSDSs in natural graphs from different domains.

3.5.2 Trace-driven Experiment Setup

In order to evaluate the performance of SPread, we conduct the trace-driven experiments on both the *Roma* and *SanF* traces in comparison with SimBet, PeopleRank [53] and PROPHET [47] algorithms. SimBet represents community based routing algorithms. PeopleRank represents centrality based routing algorithms. PROPHET represents probabilistic routing algorithms. The details of the algorithms are introduced in Chapter 2. We measure the following metrics:



(a) *Roma* (b) *SanF*

Figure 3.24: The average delay vs. different memory size.

1. *Success rate:* The percentage of packets that successfully arrive at their destination vehicles.

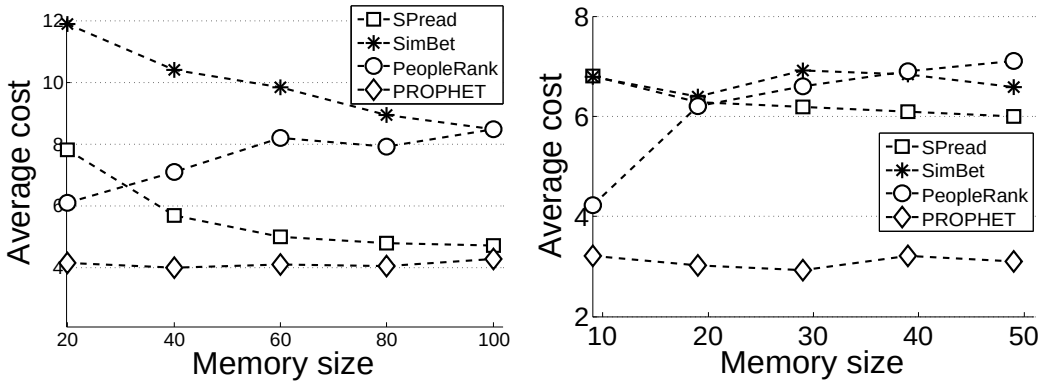
2. *Average delay:* The average time per packet for successfully delivered packets to reach their destination vehicles.

3. *Average cost:* The average number of hops per packet for successfully delivered packets to reach their destination vehicles. The more hops per packet are needed for successfully delivered packets, the more energy will be cost.

3.5.3 Performance with Different Number of Copies

Since our algorithm is designed for multi-copy routing, we compare SPread with the other three algorithms with multiple copies of each packet replicated by the spray and wait multi-copy routing algorithm [65] for fair comparisons. In order to compare the performance of ASPread with SPread, firstly, we set the multi-copy scope and total number of copies as the same as SPread. Then, we reallocate the number of copies to the communities in the multi-copy community scope based on Formula (3.12).

Figure 3.17 show the success rates with different numbers of copies per packet, respectively. Generally, the performances follow SPread>SimBet>PeopleRank>PROPHET. The performance of SimBet is a litter better than PeopleRank, since SimBet considers not only the centrality of vehicles, but also the one-hop reachability of vehicles to different communities. SPread performs better than SimBet since it carefully allocates the copies and consider the multi-hop reachability of vehicles to different communities at the same time. PROPHET performs the worst, since it is difficult to



(a) *Roma* (b) *SanF*

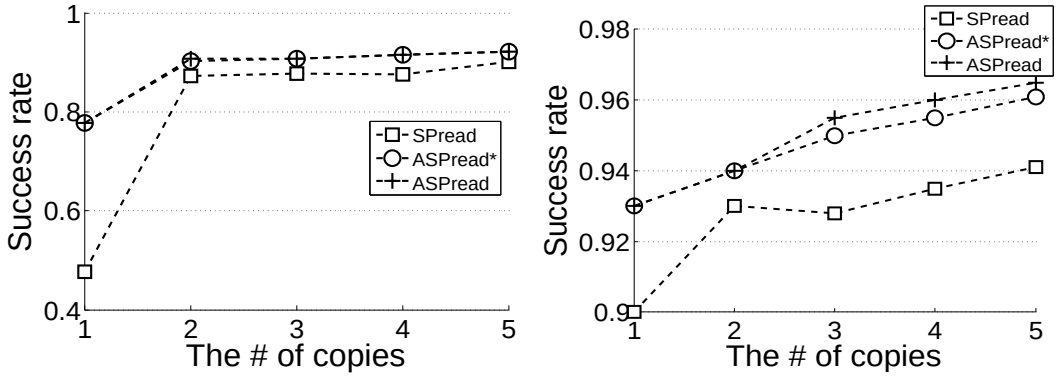
Figure 3.25: The average cost vs. different memory size.

encounter a vehicle that has a high probability to encounter the destination vehicles in the VDTNs.

Figure 3.18 show the average delays with different numbers of copies per packet. Generally, the average delays follow PROPHET>PeopleRank>SimBet>SPread. The delay of PROPHET is the largest, since the relay vehicles need to wait a long time to encounter a vehicle that has a high probability to encounter the destination vehicles in the VDTNs. The delay of SPread is the smallest, since we limit the searching scope of each copy to its own community and search through different weak ties simultaneously.

Figure 3.19 show the average costs with different numbers of copies per packet. Generally, the average costs follow PeopleRank>SimBet>SPread>PROPHET. The cost of PeopleRank is the largest, since the packets are forwarded only by the PeopleRank value without any reachability information to different vehicles. The cost of PROPHET is the smallest, since the packets are directly forwarded to the vehicles with high probability to encounter the destination vehicles. However, PROPHET has very low success rate due to the same reason. SPread performs better than SimBet and PeopleRank.

Then, we analyze the influence of the number of copies per packet to different algorithms. As shown in Figure 3.23, Figure 3.24 and Figure 3.25, when there is only 1 copy, the performance (include success rate, average delay and average cost) of SPread is a little worse than PeopleRank and SimBet, since SPread is designed for multi-copy only and each copy can search in its community only before it encounters the destination community. However, when the number of copies is slightly increased, the performance of SPread is improved significantly and exceeds the other three algorithms. This is because our weak tie multi-copy based routing algorithm carefully allocates the different copies and



(a) *Roma* (b) *SanF*

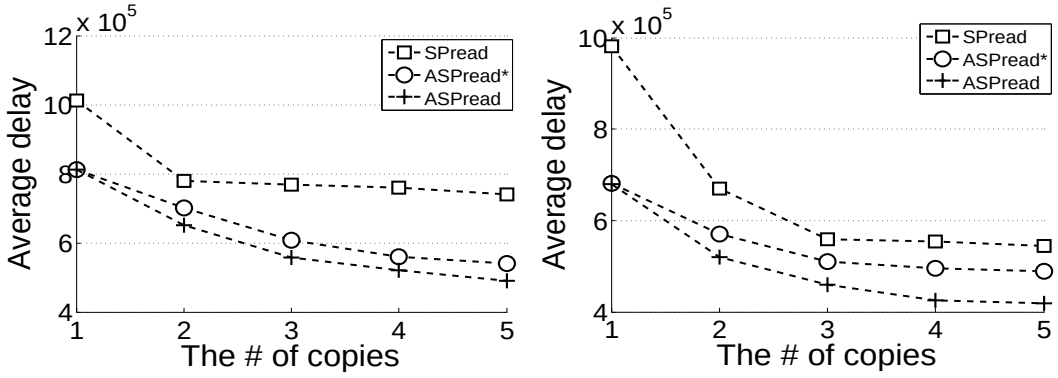
Figure 3.26: The success rate vs. different number of copies.

fully utilizes each of the copies.

3.5.4 Performance with Different Memory Sizes

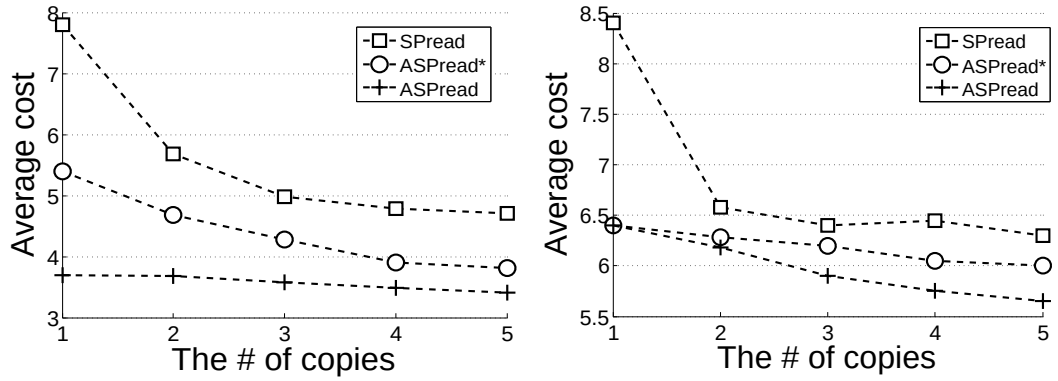
Besides the number of copies per packet, the memory size of each vehicle also influences the performance. Therefore, we analyze the influence of memory size to different algorithms. Figure 3.23, Figure 3.24 and Figure 3.25 show the success rates, average delays, and average costs with different memory sizes, where we suppose that 1 unit memory (horizontal axis) can save 1 packet. Generally, the sensitivities of different algorithms to the memory sizes follow PeopleRank>SimBet>SPread> PROPHET. The performance of PeopleRank is very sensitive to the memory size, since all the packets tend to be forwarded to few vehicles with very high PeopleRank values and the limited memory size can significantly influence the routing process negatively. PROPHET is insensitive to the memory size, since the packets only tend to find those specific vehicles with high probability to encounter the destination vehicles, which guarantees load balance. However, PROPHET generates low success rate due to the same reason. Also, as shown in Figure 3.24, the average delay of PROPHET is not stable, since the success rate is very low and the average delay is randomly influenced by very few success routings. The sensitivities of SPread and SimBet are similar which are between PeopleRank and PROPHET.

To sum up, SPread has the highest success rate, lowest average delay and medium average cost. Also, SPread is sensitive to the number of copies per packet when the number of copies is very small, and insensitive to the memory size. SimBet and PeopleRank have the medium success rate, average delay and average cost. PeopleRank is very sensitive to the memory size. PROPHET



(a) *Roma* (b) *SanF*

Figure 3.27: The average delay vs. different number of copies.



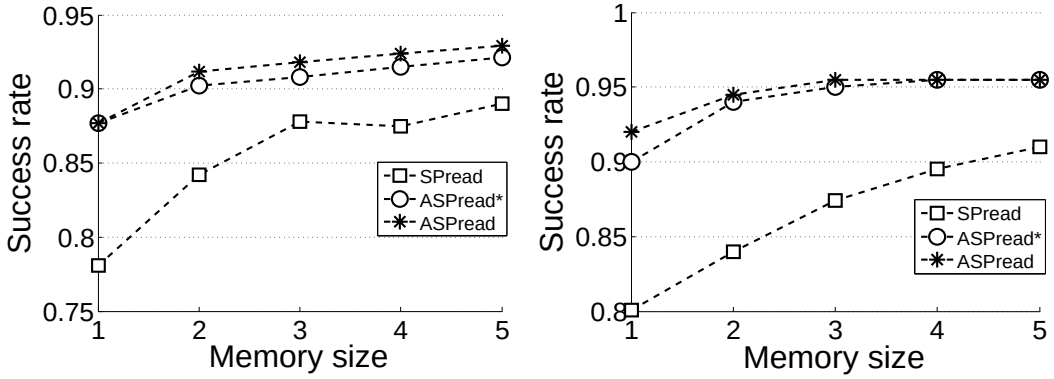
(a) *Roma* (b) *SanF*

Figure 3.28: The average cost vs. different number of copies.

has the lowest average cost due to its very specific requirement for forwarding packets. However, PROPHET has very low success rate and high average delay.

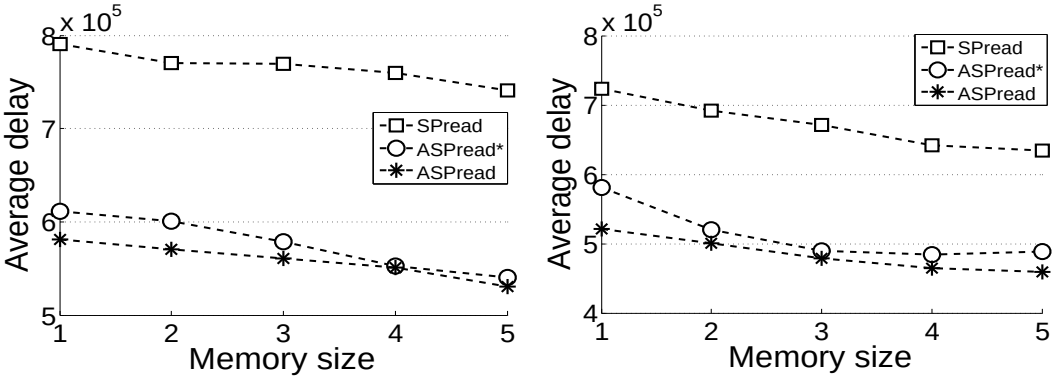
3.5.5 Performance of ASPread Compared with basic SPread

In this section, we compare SPread, ASPread without the method of adaptive determination of the number of copies based on community size (denoted by ASPread*) and ASPread with different number of copies per packet and different memory sizes of each vehicle. We measure the performance of ASPread* in order to show the individual effectiveness of the method of adaptive determination of the number of copies based on community size and the method of forwarding to core vehicles in ASPread. Figure 3.26, Figure 3.27 and Figure 3.28 show the success rates, average delays and costs with different number of copies per packet. Figure 3.29, Figure 3.30 and Figure 3.31 show the success rates, average delays and costs with different memory sizes of each vehicle. Generally, the success



(a) *Roma* (b) *SanF*

Figure 3.29: The success rate vs. different memory size.

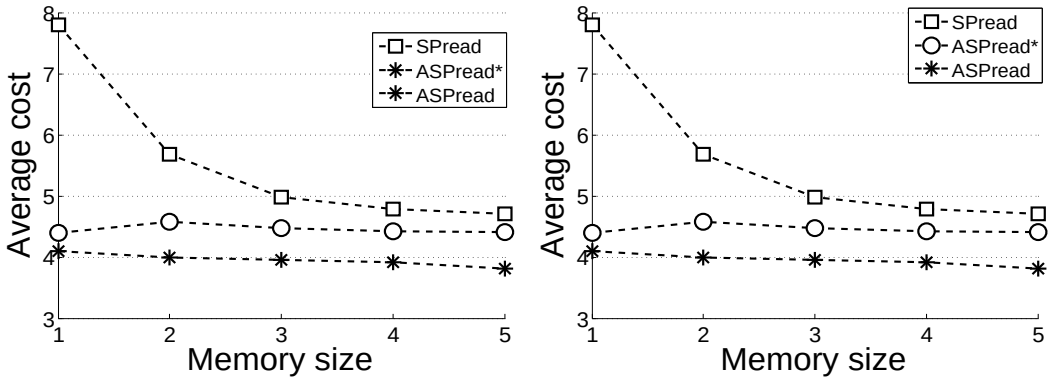


(a) *Roma* (b) *SanF*

Figure 3.30: The average delay vs. different memory size.

rate follows ASPread≈ASPread*>SPread, the average delay follows ASPread<ASPread*<SPread and the cost follows ASPread<ASPread*<SPread. The improvement of success rate of ASPread and ASPread* over SPread is significant at the beginning when the number of copies is limited since ASPread and ASPread* can more precisely deliver the copies to the core vehicles which are not only in the target vehicle' community but also are more likely to meet the target vehicle, especially when the number of copies is limited. SPread only delivers the copies to the relay vehicles in the target vehicle's community, which may not be likely to meet the target vehicle. When the number of copies is sufficient, the improvement becomes less significant since the chance to deliver to the core vehicles in the communities in SPread is increased as the number of copies is increased.

Because of the same reasons, SPread generates relatively high average delay and much higher cost due to the possibility of selecting relay vehicles which are far away from other vehicles in the same communities. Furthermore, it may take a copy a longer time to find the target vehicle if the



(a) *Roma* (b) *SanF*

Figure 3.31: The average cost vs. different memory size.

community size is larger. In ASPread and ASPread*, since the packet is directly forwarded to the core vehicles in communities which are more likely to meet other vehicles in the same communities, the number of relays is reduced and hence the performance on average delay and cost is significantly improved. Comparing ASPread and ASPread*, we see that they have similar success rates since both of them select core vehicles in the communities. However, ASPread has lower average delay and cost than ASPread* since ASPread sends more copies to communities with larger sizes and therefore it can find the target vehicles more quickly.

We also see that the average delay and average cost of ASPread are lower than ASPread*. Since ASPread distributes different number of copies to different communities based on the community size, it takes approximately same time for each copy to find the target vehicle. As a result, the average delay and average cost of ASPread are lower than ASPread* since ASPread balances the number of vehicles each copy needs to search and therefore avoids the situation that some copies search in communities with too many vehicles, which may cause delay and high cost.

To sum up, ASPread has a higher success rate and a lower average delay and cost compared with SPread since it considers the spatio-contact correlation of the community and distributes different number of copies to different communities based on the community size.

Chapter 4

Exploiting Active Sub-areas for Multi-copy Routing in VDTNs

In this chapter, we aim to improve the routing performance in VDTNs. We first analyze vehicle network traces and observe that i) each vehicle has only a few active sub-areas that it frequently visits, and ii) two frequently encountered vehicles usually encounter each other in their active sub-areas. We then propose Active Area based Routing method (AAR) which consists of two steps based on the two observations correspondingly. AAR first distributes a packet copy to each active sub-area of the target vehicle using a traffic-considered shortest path spreading algorithm, and then in each sub-area, each packet carrier tries to forward the packet to a vehicle that has high encounter frequency with the target vehicle. In addition to the basic AAR, we further propose an Advanced AAR (AAAR). In the AAAR, we improve the routing efficiency in each sub-area by exploiting spatio-temporal correlation and developing three strategies for calculating spatio-temporal correlation. Extensive trace-driven simulation demonstrates that AAR produces higher success rates and shorter delay in comparison with the state-of-the-art routing algorithms in VDTNs. Also, the simulation shows that our advanced AAAR has better performances than our AAR. The main contributions of this chapter are as follows:

1. We measure two real VNET *Roma* and *SanF* traces, which serves as the foundation for our proposed routing algorithm for VNETs.

2. We propose a traffic-considered shortest path spreading algorithm to spread different copies of

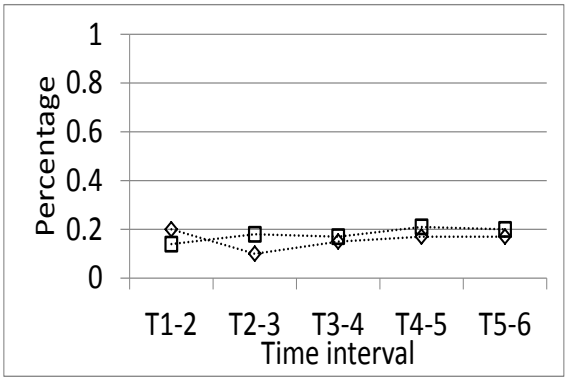


Figure 4.1: The stability of active sub-areas

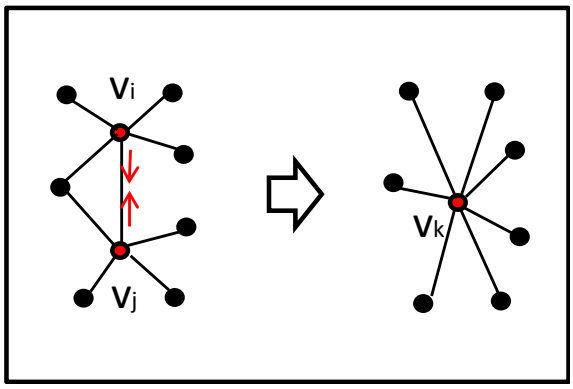


Figure 4.2: Active sub-area identification.

a packet to different active sub-areas of the target vehicle efficiently.

3. We propose a contact based scanning algorithm in each active sub-area of the target vehicle to relay the packet to the target vehicle.

4. We propose an Advanced AAR (AAAR) by exploiting the spatio-temporal correlation of the visiting times of target vehicles on different road side units.

4.1 Identification of Each Vehicle's Active Sub-areas

Current routing algorithms search the target vehicle in the entire VDTN map, which leads to low routing efficiency since some routing paths may be outside of the active sub-areas of the target vehicle. Using multi-copies to search in different active sub-areas of the target vehicle can improve routing efficiency. Because our trace measurement uses the concept of each vehicle's active sub-areas, we first introduce our method of identification of each vehicle's active sub-areas in this section before we introduce our trace measurement.

4.1.1 Stability

Nodes in a VNET are usually sparsely distributed. Therefore, it is preferable to identify active sub-areas by static GPS history data which can be easily obtained. However, whether the active sub-areas of vehicles stay stable from time to time significantly influences the efficiency of routing in the system design. Therefore, we first analyze the stability of the active sub-areas of vehicles. We divide each of the two traces to 6 time intervals with equal length and count the top 5 frequently visited road sections of all the vehicles at the end of each time interval. Then, we calculate

the percentage of changes from one time interval to the next time interval and draw Figure 4.1. The $Ti-j$ on the x axis in Figure 4.1 means the changes from time interval i to time interval j. The y axis is the percentage of the changes from one time interval to the next time interval. As shown in Figure 4.1, the active sub-areas of vehicles tend to be stable, which means that we can apply the recent GPS history data to identify active sub-areas for future system design.

4.1.2 Active Sub-Area Identification

In the entire VDTN map, a *road section* is the road part that does not contain any intersections and it is denoted by the two IDs of intersections on its two ends such as *ab* in Figure 1.4. Then, instead of searching target vehicle v on the entire VDTN map, we only direct packets to search in the road sections where the target vehicle v visits frequently. We call these road sections the *active road sections* of vehicle v. To be more specific, we define the set of active road sections of vehicle v (denoted by S_v) by:

$$S_v = \{\forall s \in S | f(s, v) > r\} \tag{4.1}$$

where S is the set of all road sections, $f(s, v)$ is the frequency that vehicle v visits road section s; and r is a visit frequency threshold. A smaller threshold r leads to more road sections in the S_v and a larger routing area and vice versa. In this chapter, we set $r = 7$ and $r = 5$ in *Roma* and *SanF* traces, respectively.

Sending a packet copy to each active road section of the target vehicle generates many packet copies and high overhead. Actually, sending a packet copy to a set of connected active road sections is sufficient because the copy can be forwarded to vehicles travelling along all these road sections to search the target vehicle. We define an active sub-area of a vehicle as a set of connected active road sections of the vehicle. We propose a method to create the active sub-areas of each vehicle by following rules.

(1) Each sub-area of a vehicle consists of connected road sections of the vehicle so that a packet copy can scan the entire sub-area for the target vehicle without the need of traveling on the inactive road sections.

(2) Each sub-area of a vehicle should have similar number of road sections so that the load balance on the size of scanning sub-areas of multiple copies can be guaranteed.

Specifically, our active sub-area identification algorithm works as follows:

(1) First, we transform the entire VDTN map to a graph. We consider each road section as a node and connect two nodes if the corresponding two active road sections share the same road intersection. Also, we tag each node with weight 1. Then, the areas division problem is translated to a graph partition problem.

(2) Next, as shown in Figure 4.2, we continually select a directly connected nodes with the smallest sum of weights, remove the edges between these two nodes, merge them to one node, and set its weight to the sum. If all pairs of directly connected nodes have equal sum of weights, we randomly select a node pair to merge.

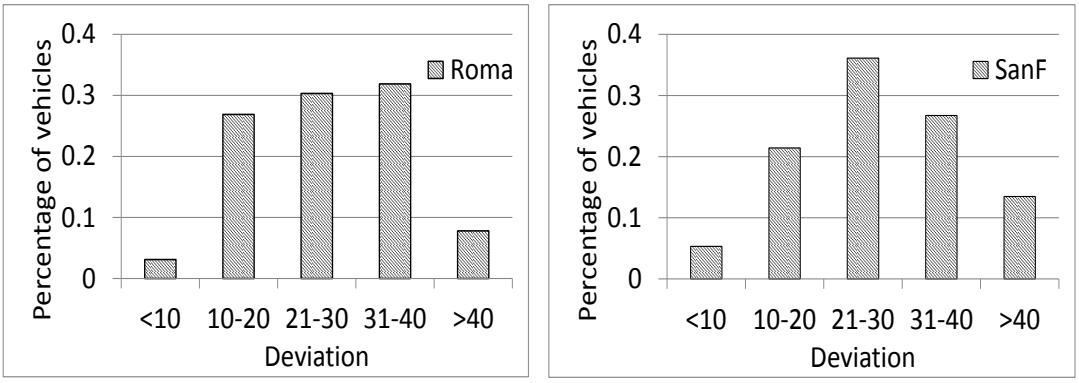
(3) We repeat step (2) until the number of nodes equals the number of active sub-areas required. Then, the corresponding road sections in one node constitute an active sub-area.

In the above process, two disconnected nodes cannot be merged, which guarantees that the road sections in each active sub-area are connected. Also, since the weight of a node represents the number of road sections corresponding to the node, merging two nodes with the smallest sum of weights can constrain the difference between the number of road sections in different active sub-areas. As a result, the above two rules are followed, which facilitates the execution of our proposed routing algorithm. The active sub-areas of each vehicle and the entire VDTN map are stored in each vehicle in VDTN. When a vehicle joins in the VDTN, it receives this information.

4.2 Trace Measurement

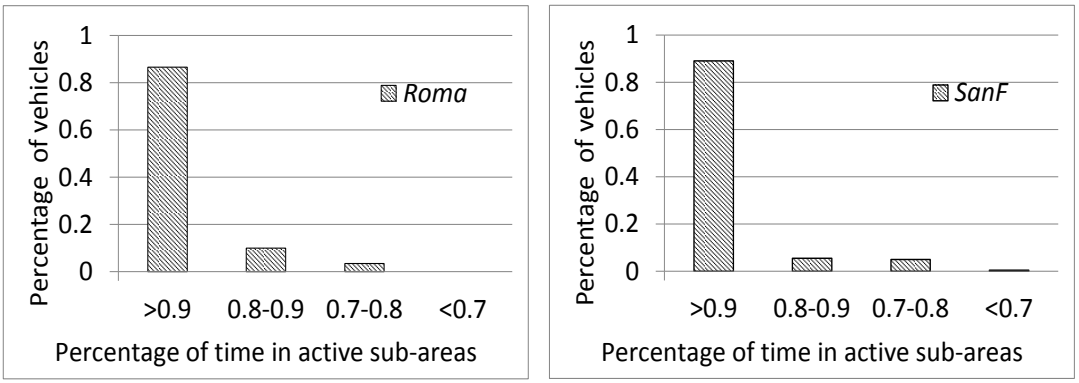
In order to design a new routing algorithm to improve the performance of current routing algorithms, we first need to better understand the pattern of vehicles' trajectories and the relationship between vehicle contact and location. Therefore, we analyze the real-world VNET *Roma* and *SanF* traces gathered by taxi GPS in different cities, referred to as *Roma* [49] and *SanF* [58]. The *Roma* trace contains mobility trajectories of 320 taxies in the center of Roma from Feb. 1 to Mar. 2, 2014. The *SanF* trace contains mobility trajectories of approximately 500 taxies collected over 30 days in San Francisco Bay Area. Our analysis focuses on following two aspects:

1. *Vehicle mobility pattern.* We expect to find out whether the movement of each vehicle exhibits a certain pattern. If each vehicle frequently visits a few sub-areas in the entire VDTN area,



(a) *Roma* (b) *SanF*

Figure 4.3: Deviation of visiting time of vehicles.

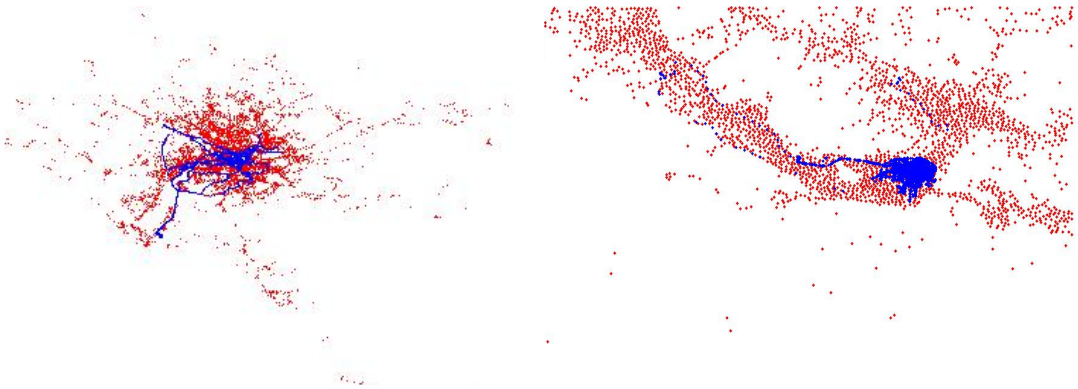


(a) *Roma* (b) *SanF*

Figure 4.4: Percentage of time spent on active sub-areas.

then our routing algorithm only needs to search these sub-areas of a target vehicle in order to improve routing efficiency.

2. *Relationship between contact and location.* Contact and centrality based routing algorithms search vehicles that have high encounter frequency with the target vehicle in the entire VDTN area. If we can identify the locations that the vehicles frequently meet the target vehicle, we can reduce the relay search area to improve the routing efficiency. Therefore, we expect to find out if such locations can be identified.



(a) *Roma* (b) *SanF*

Figure 4.5: The trajectory of a vehicle in the entire VDTN map.

4.2.1 Vehicle Mobility Pattern

In order to measure the pattern of vehicles' mobility on the entire VDTN area, we normalize the total driving time of each vehicle to 100 hours and normalize its real visiting time on each road section by:

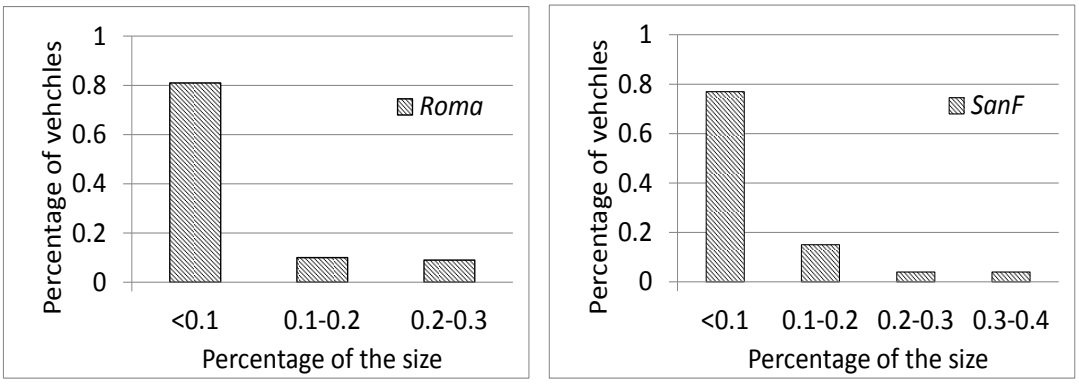
$$\bar{t}(s_i, v_i) = \frac{100 \times t(s_i, v_i)}{t_{v_i}} \tag{4.2}$$

where s_i denotes road section s_i, v_i denotes vehicle i, $\bar{t}(s_i, v_i)$ is the normalized visiting time of vehicle v_i on road section s_i, t_{v_i} is the real total driving time of vehicle v_i and $t(s_i, v_i)$ is the real visiting time of vehicle v_i on road section s_i. We then calculate the deviation of visiting time of vehicle v_i (D_{v_i}) by:

$$D_{v_i} = \frac{1}{|S|} \sum_{i \in S} \left(\bar{t}(s_i, v_i) - \bar{t}_{v_i}\right)^2 \tag{4.3}$$

where S is the set of all the road sections in the entire VDTN map and $\bar{t}_{v_i}$ is the average visiting time of vehicle v_i per road section. Since the total visiting time of each vehicle is normalized to 100 hours and S is fixed, $\bar{t}_{v_i}$ is a fixed value $\frac{100}{|S|}$ for any vehicle v_i. Figure 4.3 shows the distributions of the deviation of visiting time of vehicles in the *Roma* and *SanF* traces. The high deviations of most vehicles indicate that these vehicles' trajectories are unevenly distributed among road sections, and they frequently visit a few road sections.

Figure 4.5 shows two random vehicles' trajectories in the entire VDTN area in the *Roma* and *SanF* traces, respectively. The blue points are the vehicles' trajectories and the red points are the road intersections on the entire VDTN areas. We can find that the vehicles' trajectories are concentrated in small sub-areas in the entire VDTN areas, which confirm the results in Figure 4.3.



(a) *Roma* (b) *SanF*

Figure 4.6: Percentage of size of the entire map.

Next, we measure the percentage of time of vehicles spent on their active sub-areas. Figure 4.4 shows the distribution of the percentage of time of vehicles spent on active area. As we can see, most vehicles spent more than 90% of time on their active sub-areas. Also, as shown in Figure 4.6, Our measurement shows that usually the total size of active sub-areas of each vehicle is smaller than 10% of the size of the entire VDTN area. As shown in Figures 4.3, 4.5 and 4.4, we conclude our first observation (**O1**) as follows:

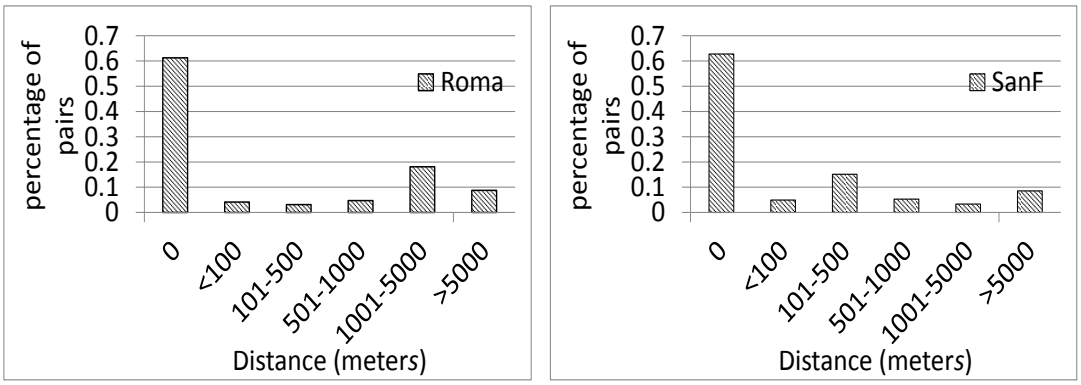
O1: *Each vehicle has its own active sub-areas which are usually very small comparing to the entire VDTN map.*

Based on this observation, we can constrain the areas of searching the target vehicle to its active sub-areas. Then, routing of packets on inactive areas of the target vehicle can be avoided and the routing efficiency can be improved.

4.2.2 Relationship between Contact and Location

First, we define a pair of vehicles as frequently encountered pair of vehicles if they encounter more than 10 times. Then, for any frequently encountered pair of vehicles v_i and v_j that frequently meet each other, we calculate the average distance $\overline{d(v_i, v_j)}$ by:

$$\overline{d(v_i, v_j)} = \frac{\sum_{i=1}^{n} d_i(v_i, v_j)}{|n|} \tag{4.4}$$



(a) *Roma* (b) *SanF*

Figure 4.7: Distance from encounter locations to active sub-areas.

where $d_i(v_i, v_j)$ is the shortest distance between the ith encounter location and the shared active sub-areas of vehicles v_i and v_j, and n is the number of encounters happened between these two vehicles. Figure 4.7 shows the distribution of the average distances of pairs of vehicles that encountered frequently in the *Roma* and *SanF* traces. We find that for most pairs of vehicles, their encounter locations are near by their active sub-areas. Actually, most encounters are happened in their active sub-areas (i.e., encounter locations with average distance 0). Therefore, we conclude our second observation (**O2**) as follows:

O2: *The frequently encountered vehicles usually encounter each other in their active sub-areas.*

Based on this observation, we can use contact based routing in each active sub-area of the target vehicle rather than the entire VDTN map, which will greatly improve the routing efficiency.

4.3 Active Area Based Routing Method

Before introducing the detailed design of AAR, we first give an overview of the routing process for a packet in AAR.

1. In the traffic-considered shortest path spreading algorithm, the source vehicle spreads different copies of the packet to the target vehicles' active sub-areas through paths with short distance and more traffic, as shown in the left part of Figure 4.8. Different from current location based routing algorithms, we identify the spreading paths with the consideration of not only the

physical distance of the paths but also the traffic condition in order to have enough relay vehicle candidates in spreading, which improves the spreading efficiency.

2. In the contact-based scanning algorithm, a packet copy in an active sub-area continually scans the sub-area until it encounters the target vehicle or a vehicle that can encounter the target vehicle more frequently as a relay vehicle, as shown in the right part of Figure 4.8. Since the scanning focuses on the active sub-areas of the target vehicle that it visits frequently and also encounters its frequently encountered vehicles, the routing efficiency is improved.

In the following, we introduce these two algorithms. As the work in [64], we assume that each road intersection is installed with a road side unit. The road side unit can send information to and receive information from nearby vehicles and store information. The road side units help to calculate traffic, receive and forward packets.

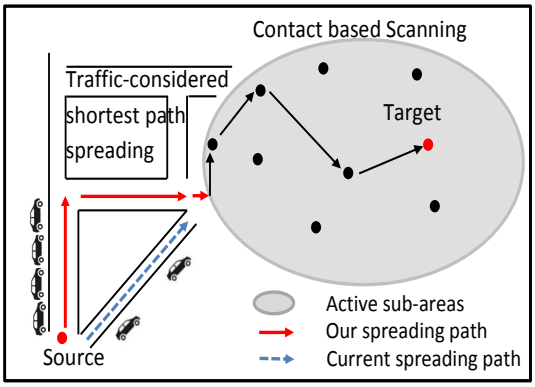


Figure 4.8: An example of the routing process.

4.3.1 Traffic-Considered Shortest Path Spreading

To spread the copies of a packet to different active sub-areas of the target vehicle, as we indicated previously, if we directly calculate the shortest path only based on the distance, the identified path may have few vehicles to function as relays, which leads to low routing efficiency. Therefore, our traffic-considered shortest path spreading algorithm jointly considers distance and traffic in selecting a spreading path. To spread the multiple packet copies, the source vehicle can send a copy to each sub-area individually, which however generates high overhead. To handle this problem, our spreading algorithm builds the spreading path tree that combines common paths of different copies in copy spreading. For example, Figure 4.10 shows an example of such a spreading path tree to spread packet copies to active sub-areas as shown in Figure 4.10, where letters $a - i$ represents the road side units. Then, the copies of a packet are relayed to their responsible active sub-areas one road side unit by one road side unit through vehicles. Each source vehicle needs the traffic information in determining the spreading path. Below, we first introduce how the traffic is calculated and dynamically updated in each vehicle in Section 4.3.1.1. We then introduce the details

of our spreading algorithm in Section 4.3.2.

4.3.1.1 Road Traffic Measurement

Road traffic varies from time to time. Therefore, it is necessary to measure the road traffic dynamically. Each the road side unit in each intersection measures the traffic of each road section as follows:

(1) When a vehicle v passes intersection a, vehicle v sends ID of the previous road side unit it passed to road side unit in intersection a (denoted by u_a).

(2) Road side unit u_a periodically updates the traffic in road section ab by:

$$T_{ba}^t = \alpha T_{ba}^{t-1} + (1-\alpha) N_{ba}^t \tag{4.5}$$

where T_{ba}^t is the traffic from intersection b to a at time t and N_{ba}^t is the number of vehicles that have pass through intersection b to a during the time period. T_{ba}^t is determined by current traffic and recent traffic in the past time and α is a damping factor where the lower value α is, the more T_{ba}^t counts the current traffic.

Also, each vehicle updates its road traffic information via two ways as follows:

1. When a vehicle v_a passes road side unit u_a, road side unit u_a sends its stored traffic information to vehicle v.

2. When vehicles v_a and v_b encounter each other, they exchange their stored traffic information. Then, the vehicles compare and update the traffic information of each road section with updated information.

4.3.1.2 Building Traffic-considered Shortest Path Tree

Based on the traffic information, we introduce our algorithm for each vehicle to build the traffic-considered shortest path tree to spread packet copies to different active sub-areas.

(1) First, we introduce a metric called *traffic-considered distance* that jointly considers the distance and traffic of a road section:

$$D_{ba} = \frac{d_{ba}}{T_{ba}} \tag{4.6}$$

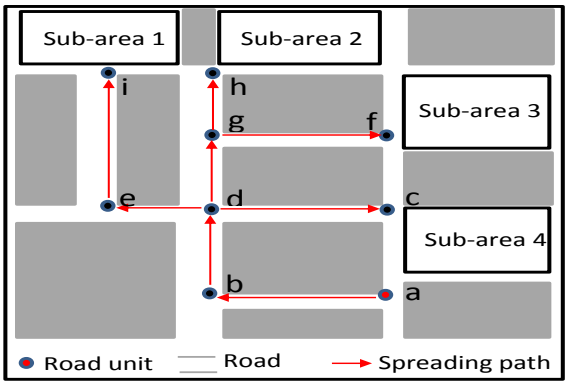


Figure 4.9: The shortest path to different sub-areas.

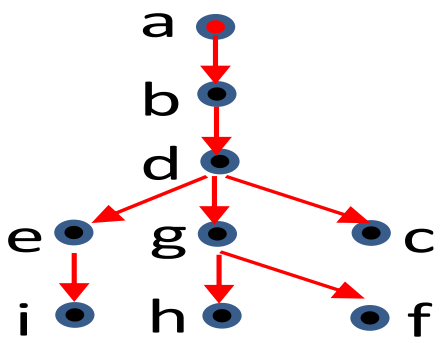


Figure 4.10: An example of spreading path tree.

where T_{ba} is the updated traffic from intersection b to a, d_{ba} is the physical distance length between b and a and D_{ba} is traffic-considered distance from b to a. It is not necessary that $D_{ba} = D_{ab}$. Using this metric in selecting spreading path, we can find path with shorter distance and higher traffic (i.e., more relay candidates), which can improve the routing efficiency.

(2) Recall that an active sub-area of a target vehicle consists of several connected road sections. In order to successfully sends a packet copy to a sub-area, a source vehicle must send the copy to an intersection of one road section in the sub-area. To find the path with minimum traffic-considered distance for a sub-area, the source vehicle first builds a graph, in which the nodes are the intersection it will pass and all the intersections of the road sections in the sub-area, two nodes are connected if their corresponding intersections are connected by a road section, and the weight of each edge equals the traffic-considered distance. It then finds the shortest path to each road intersection using the Dijkstra algorithm. Among these paths, it further picks up the shortest path as the shortest path to the sub-area.

(3) After the source vehicle calculates the shortest paths to all the active sub-areas of the target vehicle, it combines the common paths in these shortest paths to build the spreading path tree. For example, paths $a \to b \to d \to c$, $a \to b \to d \to e$, $a \to b \to d \to g \to h$, $a \to b \to d \to g \to f$ and $a \to b \to d \to e \to i$ are combined to a tree by merging the same road intersections on different paths (as shown in Figure 4.9 and Figure 4.10), so that copies can be spread efficiently.

When the source vehicle arrives at the next road side unit, i.e., u_a, it drops the packet to u_a. When a vehicle travelling to road side unit u_b passes u_a, u_a sends a packet copy to the vehicle,

Road section	Time stamp	Road section	Time stamp
ac	1:01	fg	1:14
bd	1:12	fh	1:03
cd	0:00	gi	1:09
cf	1:02	hi	1:06
dg	1:13	hj	1:05
ef	1:16	ik	1:08

Figure 4.11: An example of the scanning history table.

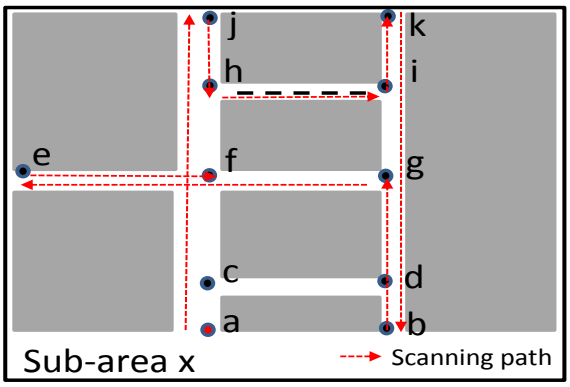


Figure 4.12: An example of the scanning road section selection.

which will drop the packet to u_b. u_b will send a copy to u_d through a vehicle travelling to u_d. Then, u_d sends packet copies to three vehicles travelling to u_e, u_g and u_c respectively. This process repeats until all road side units in the spreading path tree receive a packet copy.

4.3.2 Contact-based Scanning in Each Active Sub-area

After a packet copy arrives at an active sub-area of the target vehicle, the packet carriers (i.e., road side units and vehicles) use the contact-based scanning algorithm in the sub-area to forward the packet to the target vehicle. As in current contact based routing algorithms, each vehicle records its contact frequency to others and exchange such information upon entering. Therefore, a packet carrier can judge if its encountered vehicle is a better packet carrier, i.e., has a higher encounter frequency with the target vehicle. In our contact-based scanning algorithm, the packet is being forwarded to vehicles travelling in different road sections in order to evenly scan the sub-area to meet the target vehicle. During the scanning process, if the packet carrier meets a vehicle which is a better packet carrier or can lead to more even scanning, it forwards the packet to this entered vehicle. Once a packet carrier is about to leave the active sub-area, it drops the copy to the boundary road side unit of the sub-area, which will forward the packet to the vehicle whose traveling direction is the road section that should be scanned.

4.3.2.1 Maintaining Scanning History Table

In order to ensure that the entire active sub-area can be scanned by a packet, each packet maintains a scanning history table. The scanning history table records the scanning history of the

packet. For example, as shown in Figure 4.11, each road section in the sub-area has a time stamp which is its last scanning time. A road side unit chooses the road section that has the oldest time stamp among the reachable road sections as the next scanning road section. For example, as shown in Figure 4.12, from intersection c, the road sections that can be scanned are road sections ac, cd and cf, while road section ef cannot be scanned. Since road section cd has the oldest time stamp, it is the next scanning road section. Once a packet finishes scanning a road section, the time stamp of this road section in its scanning history table is updated with the current time.

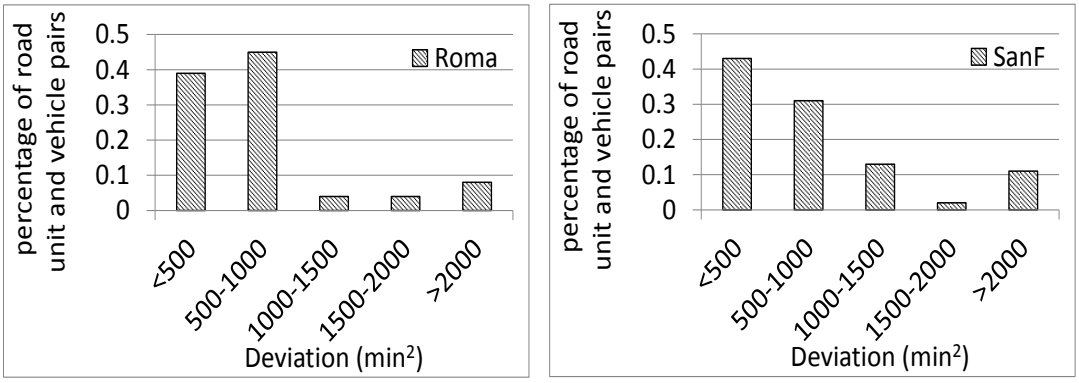
4.3.2.2 Routing Algorithm in a Sub-area

We adopt the method in [47] to measure the encounter frequency of each pair of vehicles. Specifically, the contact utility is calculated every time when once two vehicles encounter by:

$$C(v_i, v_j) = C_{old}(v_i, v_j) + (1 - C_{old}(v_i, v_j)) \times C_{init}(v_i, v_j) \tag{4.7}$$

where $C(v_i, v_j)$ is the updated encounter frequency utility; $C_{old}(v_i, v_j)$ is the old encounter frequency utility and $C_{init}(v_i, v_j)$ is the initial value of contact utility of all the pairs of vehicles, which is set to a value selected from $(0, 1)$. This definition ensures that the two vehicles with a high encounter frequency have a larger encounter frequency utility.

Below, we explain the contact-based scanning algorithm. Recall that the traffic-considered shortest path algorithm sends a packet copy to a road side unit in an active sub-area of the target vehicle. Then, the contact-based scanning algorithm is executed. First, the road side unit determines the road section that the packet should scan, which is the road section that has the oldest scan time stamp among the reachable road sections, as explained previously. Then, the road side unit will forward the packet to the passing vehicle, say v_i, with the direction to the selected road section. When v_i travels along the road section, for each of its encountered vehicle v_j, if v_j has a higher contact utility to the target vehicle or v_j's direction has a smaller time stamp then v_i's direction in the scanning history table of the packet among all the reached road sections, v_i forwards the packet to v_j. After a vehicle finishes scanning a road section, it drops the packet to the road side unit on the intersection in the end of this road section. If a vehicle is leaving the active sub-area, it also drops the packet to the boundary road side unit. Then, the road side unit decides the next scanning road section and the process repeats until the packet meets the target vehicle.



(a) *Roma* (b) *SanF*

Figure 4.13: Deviation of visiting times of vehicles in the same day.

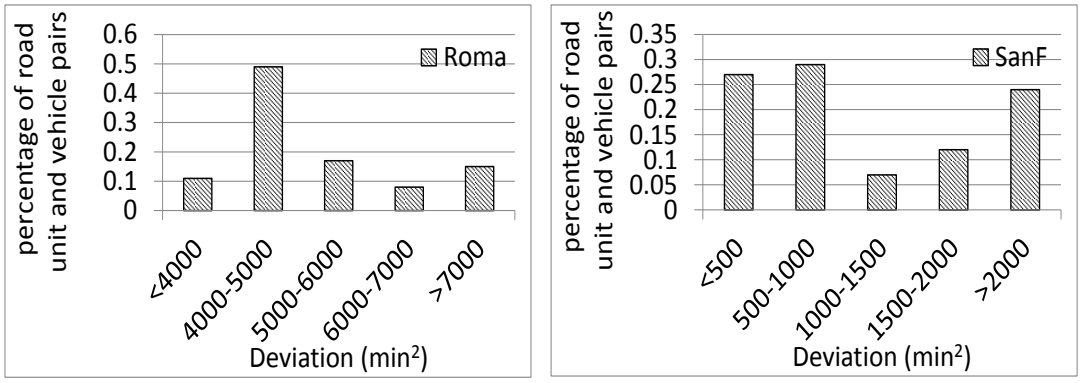
As shown in Section 5.1, the target vehicle spends most of traveling time in its active sub-areas and also it meets its frequently encountered vehicles its active sub-areas. Therefore, by scanning the target vehicle's active sub-areas and relying on vehicles with high contact utilities with the target vehicle in routing can greatly improve routing efficiency and success rate.

4.4 Advanced Active Area Based Routing Method

In AAR, we adopt a simple scanning strategy in each active sub-area when a packet copy does not encounter any relay vehicles with high contact utility with target vehicles. However, the scanning strategy may cause some problems. First, the scanning can lead to frequent packet relays among different vehicles since each vehicle usually can scan only a few road sections and then needs to drop the packet copy frequently. Such frequently relays will waste a lot of energy. Second, the scanning strategy fails to consider some useful information (e.g., the spatio-temporal correlation of vehicles) that can help decrease the success rate and average delay of the routing. Therefore, in this section, we further exploit the spatio-temporal correlation of the target vehicles to improve the efficiency of the basic AAR and propose the Advanced AAR (AAAR).

4.4.1 Measuring the Spatio-Temporal Correlation of Vehicles

For easy analysis, first, we use different road side units in the active sub-areas to denote different locations in the active sub-areas and translate the standard time to minutes in a day. For example, time 13:12 is translated to time 792 ($13 \times 60 + 12 = 792$). Then, we normalize the average



(a) *Roma* (b) *SanF*

Figure 4.14: Deviation of visiting times of vehicles in different days.

visiting time of each vehicle in a day on each road side unit by:

$$\bar{t}(r_i, v_i) = \frac{\sum_{t_i \in T(r_i, v_i)} t_i(r_i, v_i)}{|T(r_i, v_i)|} \tag{4.8}$$

where r_i denotes road side unit i, v_i denotes vehicle i, $\bar{t}(r_i, v_i)$ is the average visiting time of vehicle v_i on road side unit r_i, $t_i(r_i, v_i)$ is the ith visiting time of vehicle v_i on road side unit r_i and $T(r_i, v_i)$ is the set of visits of vehicle v_i on road side unit r_i during a certain time period. Then, we calculate the deviation of visiting time of vehicle v_i on road side unit r_i by:

$$D(r_i, v_i) = \frac{1}{|T(r_i, v_i)|} \sum_{t_i \in T(r_i, v_i)} \left(\bar{t}(r_i, v_i) - t_i(r_i, v_i)\right)^2 \tag{4.9}$$

$D(r_i, v_i)$ denotes the deviation of visiting time of vehicle v_i on road side unit r_i. If we consider the time period as one day, the deviation can reflect the differences between the visiting times during a day. A high deviation indicates the visiting times of vehicle v_i on road side unit r_i differ largely from each other, while a low deviation indicates vehicle v_i tends to visit road side unit r_i in the same time periods in a day.

The spatio-temporal correlation can be measured by the unit of each pair of road side unit and vehicle that visited the road side unit. Based on the above definition of deviation, first we calculate $D(r_i, v_i)$ of visiting times for each pair of road side unit and vehicle for each day in the trace. That is, $T(r_i, v_i)$ is the set of $t_i(r_i, v_i)$ for a given pair of (r_i, v_i) during a day.

Figure 4.13 shows the distributions of the deviation of visiting times of different road side unit and vehicle pairs in the same day in the *Roma* and *SanF* traces. The figure shows that most

road side unit and vehicle pairs have deviations in a range from 0 to 1000. If we assume each visiting time has an equal difference with the average visiting time, then the difference is less than about 30 minutes ($30 \times 30 = 900 \approx 1000$), which indicates that the visiting times in the same day have high correlation with each other for a pair of road side unit and vehicle. The low deviations of most road side unit and vehicle pairs indicate that vehicles tend to visit the same road side unit at the close times in the same day, and there exists high correlation between vehicles' location and the time.

Then, we define the representative visiting time of a pair of road side unit and vehicle in one day ($t_i(r_i, v_i)$ in Equation (4.8)) as the average visiting time in that day and the average visiting time of a pair of road side unit and vehicle in the whole time period in the traces ($\bar{t_i}(r_i, v_i)$ in Equation (4.8)) as the average of the representative visiting times in different days in the whole time period. Using the representative visiting time in each day and average visiting time in different days in the whole time period, we calculated the $D(r_i, v_i)$ for in the whole time period of each trace and plotted Figure 4.14. Figure 4.14 shows the distributions of the deviation of visiting times of different road side unit and vehicle pairs in different days in the whole time period in the *Roma* and *SanF* traces. As shown in the figure, most road side unit and vehicle pairs have deviations in a range from 4000 to 5000. If we assume each visiting time has an equal difference with the average visiting time, then the difference is less than about 70 minutes ($70 \times 70 = 4900 \approx 5000$) which is still relatively small. The low deviations of most road side unit and vehicle pairs indicate that vehicles tend to visit the same road side unit at the close time in each day, and there are high correlation between vehicles' location and the time. Therefore, we conclude our third observation (**O3**) as follows:

O3: *In the active sub-areas, most vehicles' locations have a high correlation with their visiting times.*

Based on this observation, we can improve the routing strategy in active sub-areas, which will further enhance the routing efficiency of AAR.

4.4.2 Spatio-Temporal Information based Location Visiting Time Prediction

The challenge and the key to improve the routing efficiency is to let a packet forwarder know the location of the target when the forwarder receives the packet. Based on the spatio-temporal information, we can estimate the location of the target according to the current time. In the following, we

Vehicle ID	Time period
1	14 (13:00-14:00)
2	2 (01:00-02:00)
3	13 (12:00-13:00)
4	12 (11:00-12:00)
5	1 (00:00-01:00)
...	...

Figure 4.15: An example of an active vehicle visiting time table stored on road side units.

Road unit	Time period
a	1 (00:00-01:00)
b	12 (11:00-12:00)
c	13 (12:00-13:00)
d	2 (01:00-02:00)
e	14 (13:00-14:00)
...	...

Figure 4.16: An example of an active road side unit visiting time table stored on packet copies.

propose three different strategies to predict the future visiting time dynamically. In the first strategy, we consider the most recent visiting time during a day at a road side unit as its future visiting time at other days at this road side unit. In the second strategy, we consider the average time of historical visiting times in the past as the future visiting time. In the third strategy, we consider the most frequently appeared visiting time in the historical data as the future visiting time. For example, in the historical visiting times at a road side unit, visiting time 2:00 appears most frequently. Then, we consider 2:00 as the future visiting time. We separate each day to 24 time periods as shown in Figure 4.17. For example, time 23:21 in a day is in the 24 time period. As shown in Figure 4.15, each road side unit in active sub-areas maintains an active vehicle visiting time table, where it records the vehicles that visit this road side unit and their visiting time periods. We do not consider the vehicles that are not active in the corresponding active sub-area since they barely visit any road side units in the sub-area. Using one of the three strategies, each road side unit updates the estimated future visiting time of each vehicle in its active vehicle visiting time table. Each packet's carrier collects the information from the active vehicle visiting time tables from road side units and maintains its active road side unit visiting time table as shown in Figure 4.16. Below, we introduce each of the three strategies for estimating the visiting time of a vehicle at a road side unit.

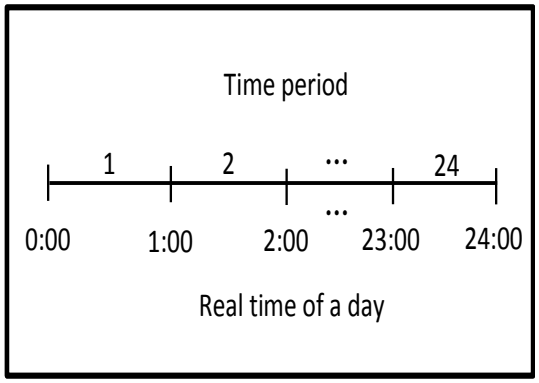


Figure 4.17: The partition of a time period.

4.4.2.1 Most recent time based location visiting time prediction

In our first strategy, we suppose that the most recent visiting time of a vehicle on a road side unit can reflect its future visiting time on this road side unit at other days. Then, each road side unit updates the visiting time in its active vehicle visiting time tables by the most recent visiting time. We present the table maintenance process below. Each road side unit r stores an active vehicle visiting time table. The table records the IDs of vehicles which are active in at least one road section that consists of road side unit r. When a vehicle v passes road side unit r, road side unit r checks whether the ID of vehicle v is recorded in its active vehicle visiting time table. If yes, road side unit r updates the visiting time in its active vehicle visiting time table by:

$$t_{new}(r, v) = t_{current}(r, v), \tag{4.10}$$

where $t_{new}(r, v)$ is the updated visiting time of vehicle v on road side unit r in the table and $t_{current}(r, v)$ is the current visiting time.

4.4.2.2 Average time based location visiting time prediction

In the above strategy of maintaining the spatio-temporal information, we dynamically update the active vehicle visiting time table by the most recent visiting time. However, without considering the historical visiting times, the accuracy of the prediction of the future visiting time may be influenced. For example, if a vehicle passes a road side unit at 6:00 though it always passes this road side unit as 2:00 in the past, and then the predicted future visiting time of 6:00 based on the first strategy is not accurate. Therefore, we propose our second strategy for predicting the future visiting time by the average time. Specifically, the predicted visiting time of vehicle v on road side unit r is calculated every time when vehicle v visits road side unit r by:

$$t_{new}(r, v) = \frac{\sum_{t(r,v) \in T_{recent}(r,v)} t(r, v)}{|T_{recent}(r, v)|} \tag{4.11}$$

where $t_{new}(r, v)$ is the updated visiting time of vehicle v on road side unit r, $T_{recent}(r, v)$ is the visiting time set of vehicle v on road side unit r during a time period and $t(r, v)$ is an element in set $T_{recent}(r, v)$.

4.4.2.3 Frequent appeared time based location visiting time prediction

Our third strategy finds a tradeoff between the most recent visiting times and the history of visiting times. To be more specific, suppose there are visiting time set $D(v,r) = t_1, t_2, ..., t_n$ of vehicle v on road side unit r, then the predicted visiting time is calculated by:

$$t_{new}(r,v) = \max \sup(t_i) \tag{4.12}$$

where $\sup(t_i)$ is the number of times of time period t_i appeared in visiting time set $D(v,r)$. For example, suppose $D(v,r) = \{1,1,1,2,2,4\}$, then $\sup(1) = 3$, $\sup(2) = 2$ and $\sup(4) = 1$, and $t_{new}(r,v) = 1$.

In Section 4.5.3.1, we will evaluate and compare the performance of the above three strategies.

4.4.3 Routing Algorithm in AAAR

In AAAR, a packet copy arrives at an active sub-area of the target vehicle by the same method used in AAR. At the same time, each packet copy maintains an active road side unit visiting time table which records the visiting times of target vehicle on road side units in the corresponding active sub-area of the packet copy as shown in Figure 4.16. In the active road side unit visiting time table, the visiting times of target vehicle on road side units are all initialized as not available (NA). Then, the active road side unit visiting time table is updated by checking the vehicle visiting time tables stored on the road side units every time when the packet copy reaches a road side unit in its corresponding active sub-area. The active road side unit visiting time tables help improve the routing efficiency. When a packet copy arrives a road side unit, if the current time is the same as the visiting time of the target vehicle at this road side unit, the packet copy can stay on the road side unit to meet the target. Also, the packet carrier can forward the packet to the road side unit where the target is visiting at the current time.

Based on the active road side unit visiting time table, after a packet copy arrives at an active sub-area of the target vehicle, it searches the target vehicle in the active sub-area by the following process.

1. First, the packet copy checks its active road side unit visiting time table of its target vehicle

on road side units in the corresponding active sub-area. If there is one road side unit that has the same time period as the current time, then go to Step 2; otherwise, go to Step 3.

2. The packet carrier uses the traffic-considered shortest path spreading method introduced in Section 4.3.1 to spread the packet copy to the road side unit with the same time period as the current time. Then, go to Step 4.

3. The packet carrier uses the contact-based scanning method introduced in Section 4.3.2 to forward the packet to the next road side unit. Then, go to Step 1;

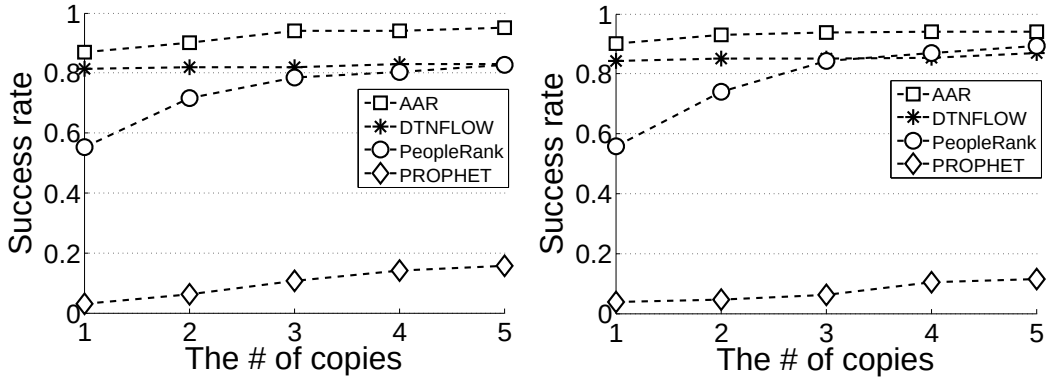
4. Once the packet copy arrives at the road side unit with the same time period as the current time, it stays on the road side unit until it encounters the target vehicle, a vehicle with a high contact utility (introduced in Section 5.2) or the time period has elapsed. Once the time period has elapsed, go to Step 1; Once the packet copy meets a vehicle with a high contact utility with the target vehicle, go to Step 5.

5. The vehicle is selected as relay vehicle and the packet begins to scan road sections until the vehicle leaves the active sub-area or encounters another vehicle with a higher contact utility. Once the vehicle leaves the active sub-area, the copy is relayed to the boundary road side unit and go to Step 1. Once the vehicle encounters another vehicle with a higher contact utility, go to Step 5 again.

Based on this process, we can guarantee that the packet can be efficiently relayed by considering not only the target's frequently encountered vehicles and its frequently encountered road side units at different times, but also the predicted visiting times on different road side units. AAAR can decrease the frequent deliveries of the packet copy between different vehicles and save energy.

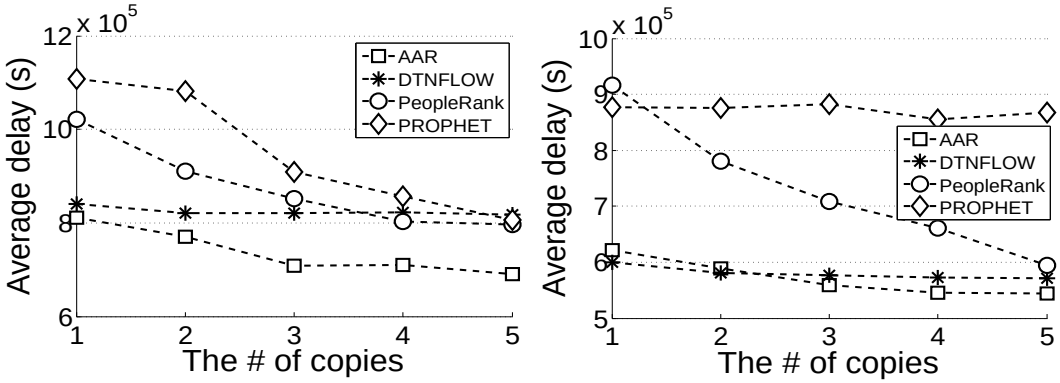
4.5 Performance Evaluation

In order to evaluate the performance of AAR, we conduct the trace-driven experiments on both the *Roma* and *SanF* traces in comparison with DTN-FLOW [14], PeopleRank [53] and PROPHET [47] algorithms. DTN-FLOW represents location based routing algorithms, PeopleRank represents centrality based routing algorithms, and PROPHET represents contact based routing algorithms. The details of the algorithms are introduced in Chapter 2. We measure the following metrics:



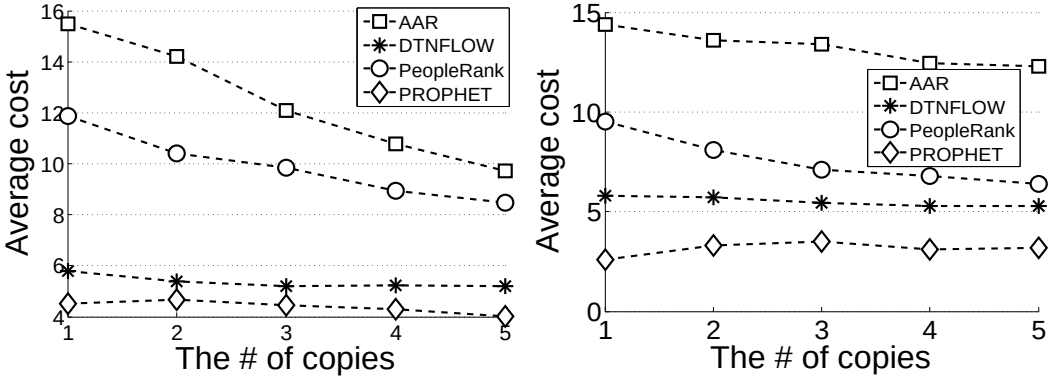
(a) *Roma* (b) *SanF*

Figure 4.18: The success rate vs. different number of copies.



(a) *Roma* (b) *SanF*

Figure 4.19: The average delay vs. different number of copies.



(a) *Roma* (b) *SanF*

Figure 4.20: The average cost vs. different number of copies.

1. *Success rate:* The percentage of packets that successfully arrive at their destination vehicles.

2. *Average delay:* The average time per packet for successfully delivered packets to reach their destination vehicles.

3. *Average cost:* The average number of hops per packet for successfully delivered packets to reach their destination vehicles. The more hops per packet are needed for successfully delivered packets, the more energy will be cost.

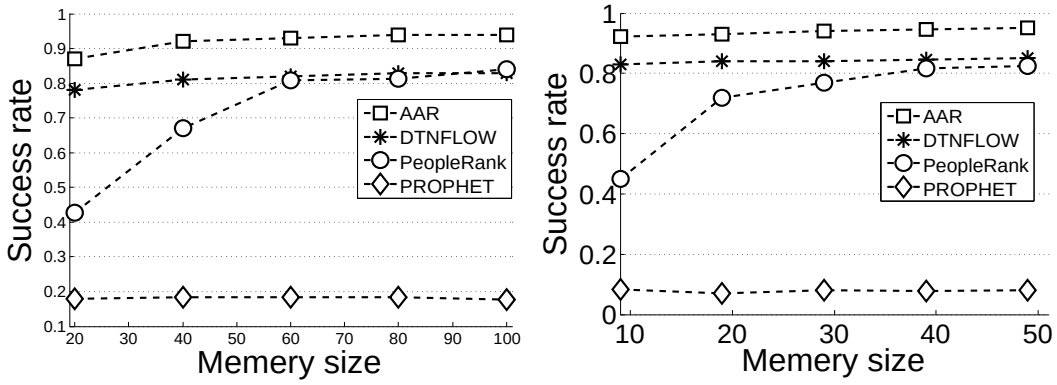
In our experiments, the number of active sub-areas of the target vehicle depends on the number of multiple copies of the packet. To be more specific, we spread one copy of a packet to each active sub-area and therefore, the number of active sub-areas equals the number of multiple copies.

4.5.1 Performance with Different Number of Copies

Since our algorithm is designed for multi-copy routing, we compare AAR with the other three algorithms with multiple copies of each packet replicated by the spray and wait multi-copy routing algorithm [65] for fair comparisons. Figure 4.18 show the success rates with different numbers of copies per packet in the *Roma* and *SanF* traces , respectively. Generally, the success rate follows AAR>DTN-FLOW>PeopleRank>PROPHET. The performance of DTN-FLOW is better than PeopleRank since DTN-FLOW divides the very large area to sub-areas and avoid to search the target vehicles on a very large area. AAR performs better than DTN-FLOW since AAR considers the encounter history. PROPHET performs the worst, since it is difficult to encounter a vehicle that encounters the target vehicle frequently in the very large area.

Figure 4.19 show the average delays with different numbers of copies per packet. Generally, the average delays follow PROPHET>PeopleRank>DTNFLOW>AAR. The delay of PROPHET is the largest, since the copies of a packet waste most time outside of active sub-areas where target vehicle barely visits brought by relay vehicles, as shown in the left part of Figure 1.3. The delay of DTN-FLOW is smaller than PROPHET since DTN-FLOW limits the routing paths in certain sub-areas. The delay of AAR is the smallest, since AAR not only spreads each copy to its responsible active sub-area efficiently by traffic-considered paths, but also scans different active sub-areas with the help of vehicles that encounter target vehicles frequently simultaneously.

Figure 4.20 show the average costs with different numbers of copies per packet. Generally, the average number of hops follow PeopleRank > AAR> DTNFLOW > PROPHET. The number



(a) *Roma* (b) *SanF*

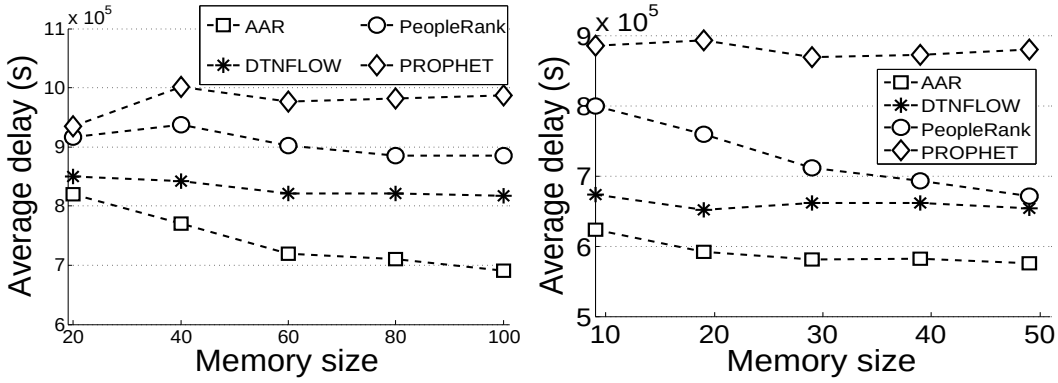
Figure 4.21: The success rate vs. different memory size.

of hops of PeopleRank is the largest, since the packets are forwarded only by the PeopleRank value without any reachability information to different vehicles. The number of hops of PROPHET is the smallest, since the packets are directly forwarded to the vehicles with high probability to encounter the target vehicles. However, PROPHET has very low success rate due to the same reason. The number of hops of DTNFLOW is also very small since the packets are waiting on the landmarks in the most time. AAR performs better than PeopleRank.

Then, we analyze the influence of the number of copies per packet to different algorithms. As shown in Figure 4.18, Figure 4.19 and Figure 4.20, when there is only 1 copy, the performance (include success rate, average delay and average cost) of AAR is a little worse than PeopleRank and SimBet, since AAR is designed for multi-copy only and each copy can search in its community only before it encounters the destination community. However, when the number of copies is slightly increased, the performance of AAR is improved significantly and exceeds the other three algorithms. This is because our weak tie multi-copy based routing algorithm carefully allocates the different copies and fully utilizes each of the copies.

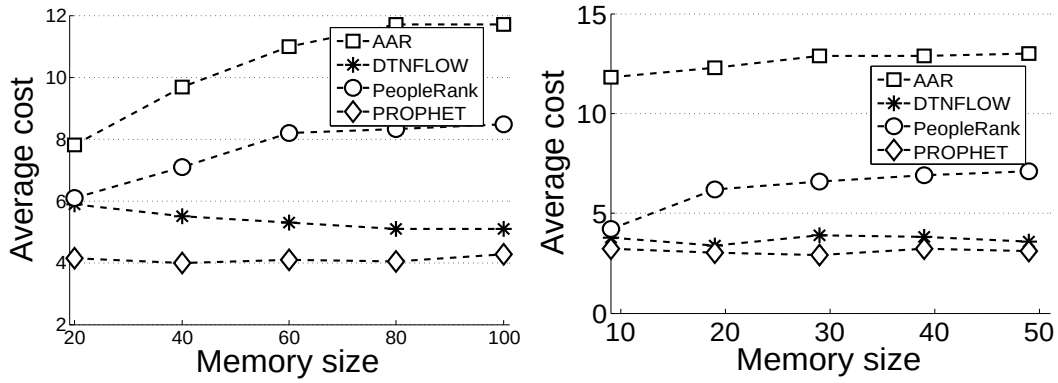
4.5.2 Performance with Different Memory Sizes

Besides the number of copies per packet, the memory size of each vehicle also influences the performance. Therefore, we analyze the influence of memory size to different algorithms. Figure 4.21, Figure 4.22 and Figure 4.23 shows the success rates, average delays and average costs with different memory sizes, where we suppose that 1 unit memory (horizontal axis) can save 1 packet. Generally, the sensitivities of different algorithms to the memory sizes follow PeopleRank>AAR>DTNFLOW>



(a) *Roma* (b) *SanF*

Figure 4.22: The average delay vs. different memory size.

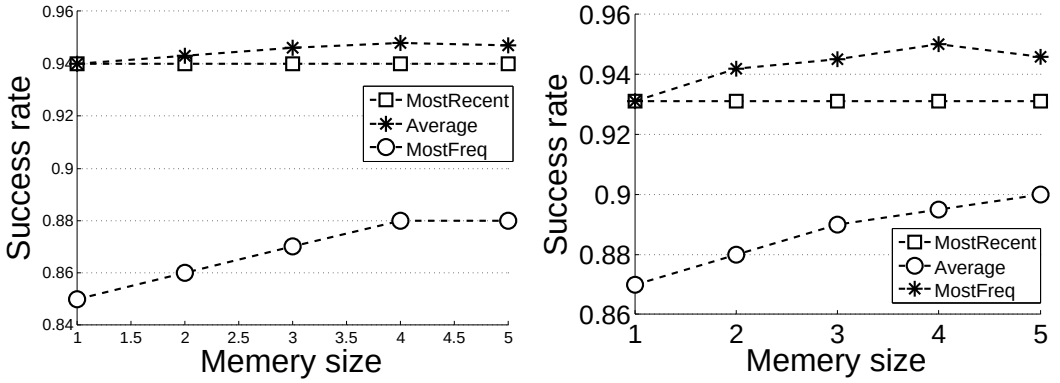


(a) *Roma* (b) *SanF*

Figure 4.23: The average cost vs. different memory size.

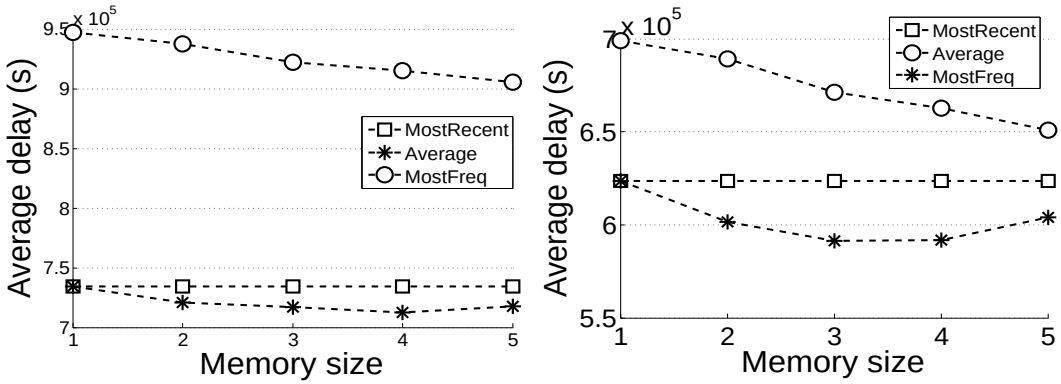
PROPHET. The performance of PeopleRank is very sensitive to the memory size, since all the packets tend to be forwarded to few vehicles with very high PeopleRank values and the limited memory size can significantly influence the routing process negatively. PROPHET is insensitive to the memory size, since the packets only tend to find those specific vehicles with high probability to encounter the target vehicles, which guarantees load balance. However, PROPHET generates low success rate and long delay due to the reasons we mentioned in Section 4.5.1. DTNFLOW is also not sensitive to the memory size since each packet is relayed in limited times from one landmark to another landmark. The performance success rate and average delay of AAR is slightly improved with the increasing memory size since a larger memory size allows packets to scan sub-areas more frequently.

To sum up, AAR has the highest success rate and the lowest average delay. However, AAR is a little sensitive to the number of copies and the memory size. DTNFLOW and PeopleRank have



(a) *Roma* (b) *SanF*

Figure 4.24: The success rate vs. different number of copies.



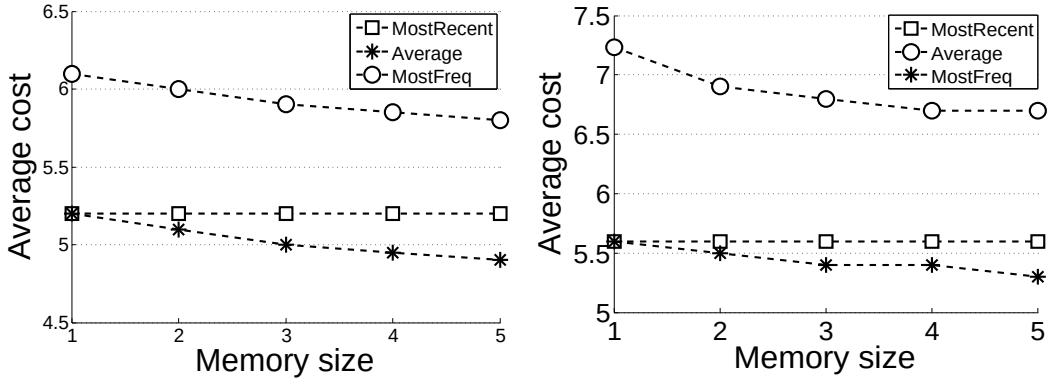
(a) *Roma* (b) *SanF*

Figure 4.25: The average delay vs. different number of copies.

the medium success rate and average delay. However, PeopleRank is very sensitive to the number of copies and the memory size. DTNFLOW and PROPHET is not sensitive to the number of copies and the memory size. However, PROPHET has very low success rate and high average delay. To sum up, considering memory size and limited number of copies are not a main concern in VDTN routing, AAR performs best in the four routing algorithms.

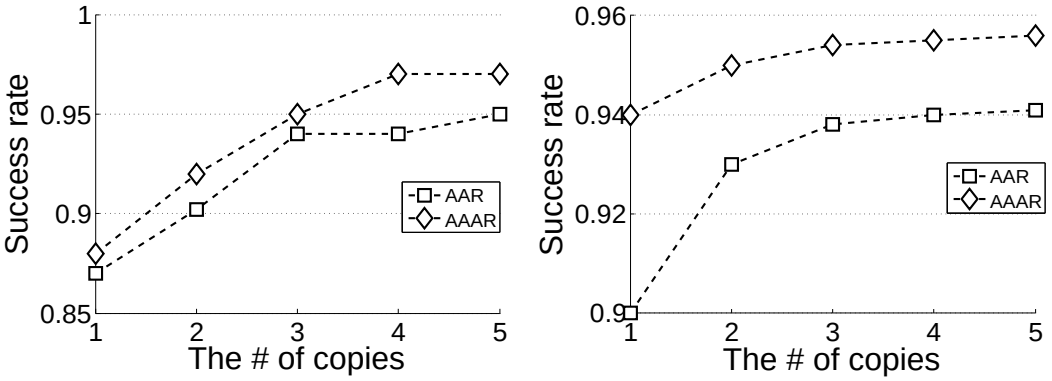
4.5.3 Performance of Advanced AAR (AAAR)

In this section, we evaluate the performance of advanced AAR and compare it with the basic AAR.



(a) *Roma* (b) *SanF*

Figure 4.26: The average cost vs. different number of copies.



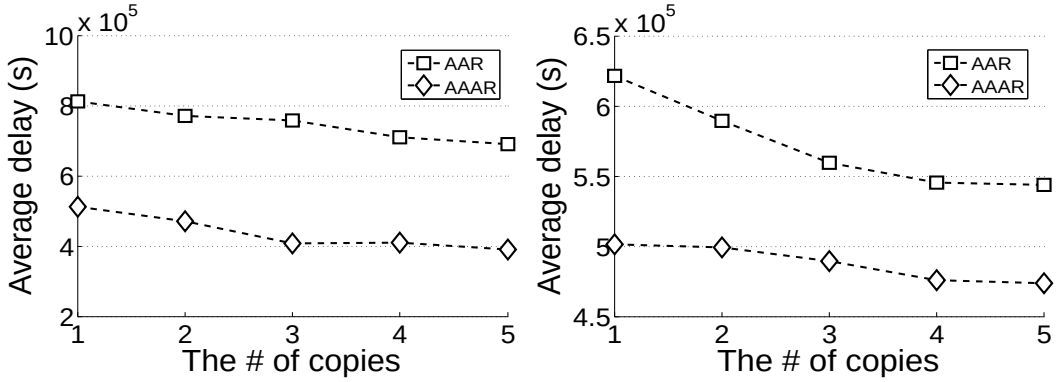
(a) *Roma* (b) *SanF*

Figure 4.27: The success rate vs. different number of copies.

4.5.3.1 Performance of different strategies in location visiting time prediction

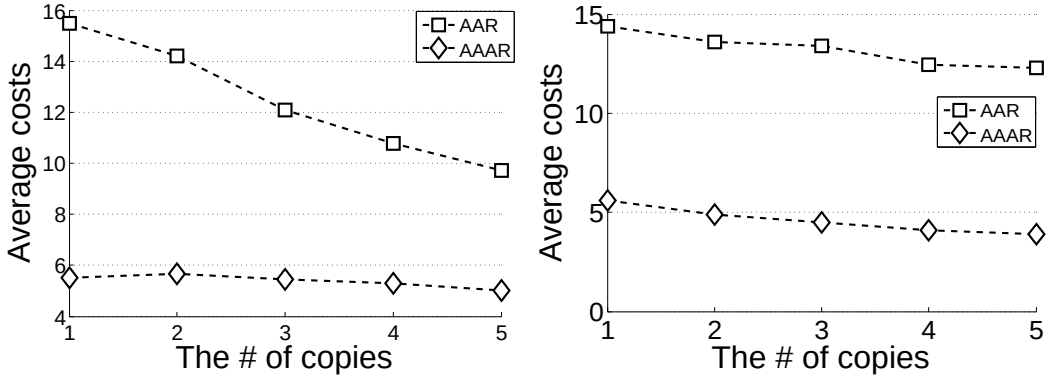
In Section 4.4.2, we designed three strategies for predicting the visiting time of each vehicle on each road side unit. In this section, we evaluate the performance of the three different strategies in packet routing. Here, we denote the strategy introduced in Section 4.4.2.1 as *MostRecent*, the strategy introduced in Section 4.4.2.2 as *Average* and the strategy introduced in Section 4.4.2.3 as *MostFreq*.

Figure 4.24 show the success rates with different memory sizes on each road side unit in the *Roma* and *SanF* traces, respectively, where we suppose that 1 unit memory (horizontal axis) can save 1 visiting time for each vehicle on the active vehicle visiting time table stored in road side units. Generally, the success rate follows *MostFreq*>*MostRecent*>*Average*, which indicates that the



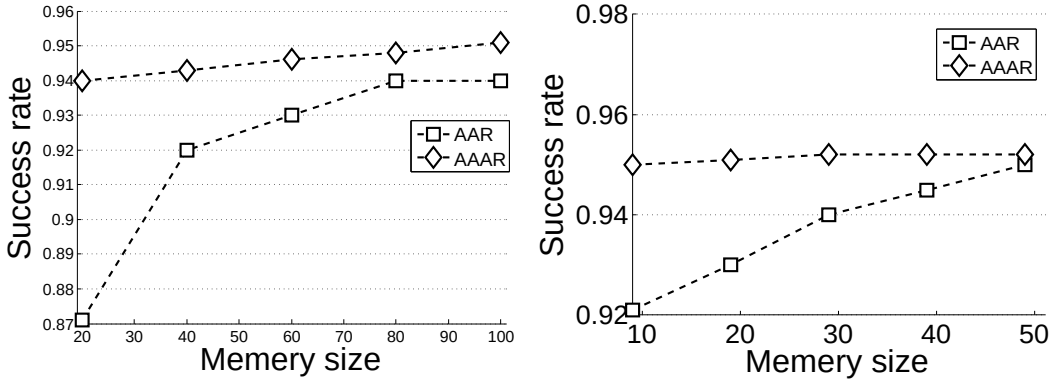
(a) *Roma* (b) *SanF*

Figure 4.28: The average delay vs. different number of copies.



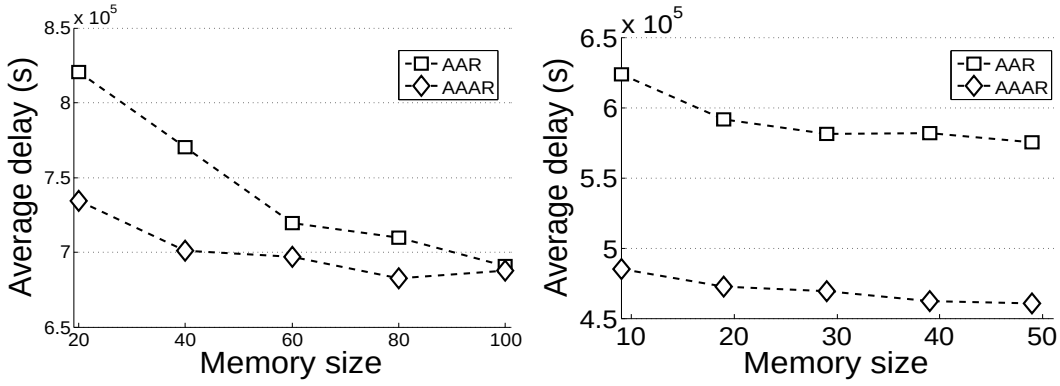
(a) *Roma* (b) *SanF*

Figure 4.29: The average cost vs. different number of copies.



(a) *Roma* (b) *SanF*

Figure 4.30: The success rate vs. different memory size.



(a) *Roma* (b) *SanF*

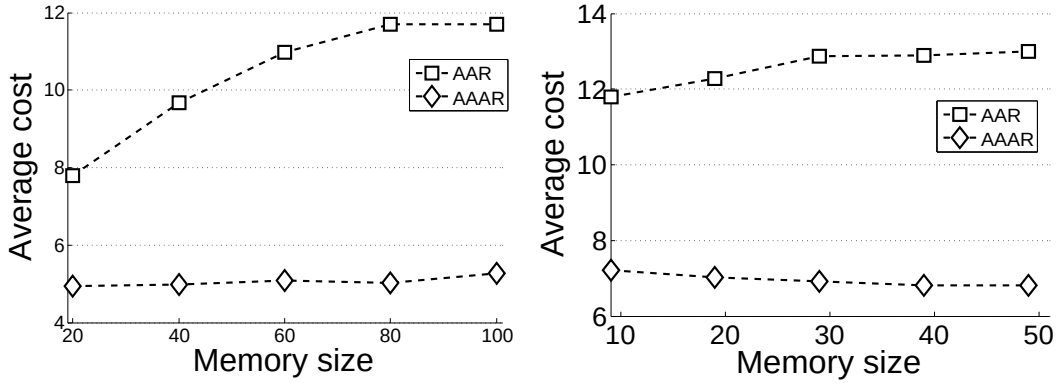
Figure 4.31: The average delay vs. different memory size.

average visiting time is not a good choice for predicting the future visiting time and the frequent appeared visiting time in the history tends to appear in the future. Also, when the memory size equals 1, *MostFreq* can only use the most recent visiting time to update the table and then *MostFreq* equals *MostRecent*. Therefore, as the memory size increases, the performance of *MostFreq* increases since there are more historical visiting times for predicting the future visiting time.

Figure 4.25 show the average delays with different memory sizes on each road side unit in the *Roma* and *SanF* traces, respectively. Generally, the average delay follows *Average*>*MostRecent*> *MostFreq*, which is consistent with the performance of success rate in Figure 4.24 due to the same reasons. The results confirm that *MostFreq* is the best choice for predicting the future visit time among the three strategies.

Figure 4.26 show the costs with different memory size on each road side unit in the *Roma* and *SanF* traces, respectively. Generally, the costs follows *Average*>*MostRecent*>*MostRecent*, which is consistent with the performance of success rate in Figure 4.24 and the average delays in Figure 4.25 since the inefficient routing leads to low success rate, long delay and more times of deliveries for routing.

From Figure 4.24, Figure 4.25 and Figure 4.26, we can conclude that strategy *MostFreq* has the highest success rate, shortest average delay and lowest cost, and both *MostFreq* and *MostRecent* have much higher success rate, lower average delays and costs than *Average*, which indicates that *Average* is the worst strategy for the prediction. The better routing performances can reflect better prediction of the strategies on future visiting time. Therefore, we can claim that *MostFreq* has the best prediction performance among the three strategies. Therefore, we adopt strategy *MostFreq* in



(a) *Roma* (b) *SanF*

Figure 4.32: The average cost vs. different memory size.

the following analysis.

4.5.3.2 Performance of AAAR comparing with basic AAR

In this section, we compare AAR and AAAR with different number of copies per packet and different memory sizes of each vehicle.

Figure 4.27, Figure 4.28 and Figure 4.29 show the success rates, average delays and costs with different number of copies per packet. Generally, the success rate follows AAAR>AAR, the average delay follows AAR>AAAR and the cost follows AAAR<<AAR. The improvement of performance success rate is not so significant since AAR already has a very good performance on success rate comparing to other algorithms. However, AAR generates relatively high average delay and much higher cost due to the frequent relays in scanning. In AAAR, since the packet copy stays on the road side unit which are most likely to be visited by the target vehicles at current time for the most time during the routing period or the copy is directly forwarded to the road side unit which is the predicted target's location, the number of relays in scanning is reduce and hence the performance on average delay and cost is significantly improved.

Figure 4.30, Figure 4.31 and Figure 4.32 show the success rates, average delays and costs with different memory sizes of each vehicle, where we suppose that 1 unit memory (horizontal axis) can save 1 packet. Generally, the sensitivities of different algorithms to the memory sizes follow AAR>AAAR. The performances success rate and average delay of both AAR and AAAR are improved with the increasing memory size since a larger memory size allows packets to scan sub-areas more frequently. The costs of both AAR and AAAR increase with the increasing memory

size since a larger memory size can lead to more times of relays.

To sum up, AAAR has a higher success rate and a lower average delay compared with AAR since we improved the uniformly scanning strategy by predicting the future visiting time on road side units of the target in routing. Also, AAAR has a much lower cost since once the packet copy arrives a road side unit that is predicted to be visited by the target vehicle at the current time period, the packet copy stays on the road side unit or the packet is directly forwarded to the road side unit which is the predicted target's location, which significantly decreases the relay cost.

Chapter 5

DIAL: A Distributed Adaptive-Learning Routing Method in VDTNs

5.1 Rationale

There are many works comparing the overall performances of different routing methods. However, there lacks a comprehensive analysis when it comes to the performances of different vehicle pairs with different features. Therefore, in this section, we measure the success rates of different routing methods on vehicle pairs with different features in *Roma* and *SanF* traces. For the routing methods, as introduced in Chapter 2, we choose AAR [73], Prophet [47] and PeopleRank [53] which represent location, contact and centrality based routing methods, respectively. For the features of vehicle pairs, we measure the contact distance, the geographic distance and the centrality of a vehicle pair which are defined as follows:

1. **Contact distance of a vehicle pair:** We first transfer the traces to *contact graph*s based on the contact durations. The nodes of the graphs are the taxies in the traces, the edges are the contacts between pairs of taxies. We naturally think that if two vehicles encounter

each other more often, they are in a closer relationship and only the contacts which have accumulative durations long enough can be considered as edges. To be more specific, we define an accumulative contact duration threshold (3000s in *Roma* and 5000s in *SanF*) and the contacts with accumulative durations larger than the threshold can be considered as edges. In the contact graphs, we calculate the contact distance of two vehicles as the shortest path between the two vehicles.

2. **Geographic distance of a vehicle pair:** AAR defines each vehicle with an active area. The vehicle frequently visits its corresponding active area. Here, we define the geographic distance of a vehicle pair as the shortest distance between the two vehicles' active areas.

3. **Centrality of a vehicle pair:** We define the centrality of a vehicle as the PageRank value of the vehicle. The centrality of a vehicle pair is the sum of the two vehicles' centralities.

5.1.1 Measurement

After we have determined the routing methods and features for the analysis. We run contact, centrality and location based methods on the traces simultaneously. Firstly, we randomly pick 1000 vehicle pairs (500 pairs in *Roma* and 500 apirs in *SanF*) and use contact, centrality and location based routing methods to deliver a packet between each vehicle pair simultaneously. Then we analyze the performance of different routing methods on different vehicle pairs. Figure 5.1 shows the percentage of vehicle pairs that each routing method performs the best. Generally, the performances of the success rate follow location > centrality > contact. However, when it comes to individual vehicle pairs, AAR performs best on 47% of the vehicle pairs. PeopleRank performs best on 42% of the vehicle pairs and Prophet performs best on 11% of the vehicle pairs. Therefore, we cannot conclude that one routing method is better than another routing method for every vehicle pair although the overall success rates are comparable. Actually, although location routing method which has the highest success rate can perform best on a lot of vehicle pairs, contact routing method which has lowest success rate can still perform better on some vehicle pairs.

Next, we try to figure out the reason why different routing methods have different routing performances on different vehicle pairs. Firstly, in order to find the reason why location based routing method performs better than centrality and contact based routing methods, we select top 50 vehicle pairs with the shortest delays when using contact routing method. Then, we compare their contact

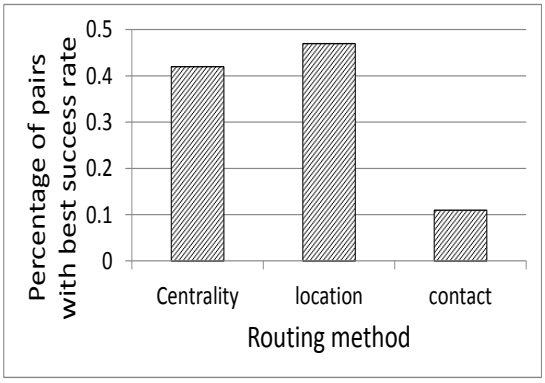


Figure 5.1: The diversity of different routing methods

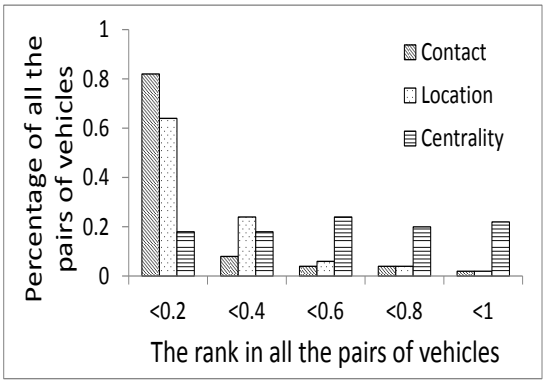


Figure 5.2: The top 50 vehicle pairs with the shortest delays of contact routing method

distances with geographic distances and centralities. Since contact distances, geographic distances and centralities cannot be compared directly, we transfer the values of different metrics to the ranks in all the vehicle pairs in the analysis. For example, for the contact distance, if the contact distance is 0.25, it means that the contact distance is longer than 25% of all the vehicle pairs; for the geographic distance, if the geographic distance is 0.25, it means that the geographic distance is longer than 25% of all the vehicle pairs; for the centrality, if the centrality is 0.25, it means that the centrality is smaller than 25% of all the vehicle pairs. Then, Figure 5.2 compares the contact distances with geographic distances and centralities of vehicle pairs. As shown in Figure 5.2, those vehicle pairs tend to have a relatively closer contact distances in all the vehicle pairs and at the same time have relative longer geographic distances and smaller centralities in all the vehicle pairs. This observation is reasonable obviously. For example as shown in Figure 5.5, suppose two vehicles A and B are in a same community named ECE community. However, the vehicles in ECE community are distributed in two different locations: ICAR of Greenville and Clemson. Vehicle A is in Greenville and vehicle B is in Clemson. In this scenario, we should put a particular emphasis on the contact based method if A wants to send B a packet. Suppose A has a location utility of 50 miles (the vehicle's frequently visited location is 50 miles away from Clemson), a centrality utility of 100 (the vehicle can meet 100 cars a day) and a contact utility of 0.09 (the vehicle meet B with probability 0.09). now vehicle A meets three vehicles: a vehicle with much higher location utility 10 miles (the vehicle's frequently visit location is 10 miles away from Clemson), a vehicle with a much higher centrality utility 1000 (the vehicle can meet 1000 cars a day) and a vehicle with a little higher contact utility 0.1 (the vehicle meet B with probability 0.1). If A chooses location method and the vehicle with Greenville

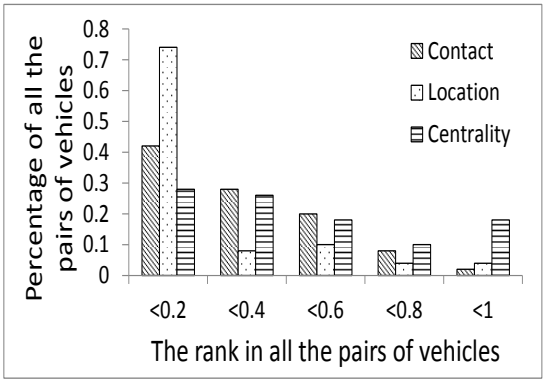


Figure 5.3: The top 50 vehicle pairs with the shortest delays of location routing method

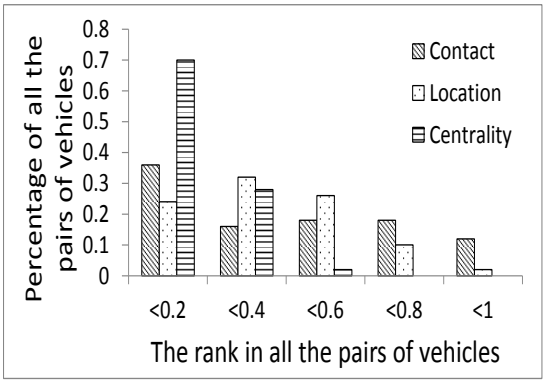


Figure 5.4: The top 50 vehicle pairs with the shortest delays of centrality routing method

with a location utility 10 mile, we may approach Greenville faster by current relay vehicle. However, it cannot be guaranteed that the packet can still easily find the next vehicle which is going to ICAR in Greenville since most of the vehicles in Greenville are not going to ICAR. Therefore, the location based method may cause a failure although we think we select a suitable vehicle with a little high location utility at one of the hops. Also, if A chooses vehicle which is very active as the relay vehicle, the relay vehicle may visit a lot of places but it is very likely that the relay vehicle won't visit Clemson at all since the map is very large. Therefore, the centrality based method may also cause a failure. On the contrary, A may just choose a vehicle with a little higher contact utility than itself. But since A and B are in the same community, it is guaranteed that there must be better choices one after another and it is with high probability that contact based method can successfully deliver the packet at last. Therefore, in this scenario, maybe it's a better choice to choose the contact method and a relay vehicle with a little higher contact utility than location and centrality methods and the relay vehicles with much higher location and centrality utilities.

Similarly, we select top 50 vehicle pairs with the shortest delays when using location routing method. Then, we compare their geographic distances with contact distances and centralities. Figure 5.3 compares the contact distances with geographic distances and centralities of vehicle pairs. As shown in Figure 5.3, those vehicle pairs tend to have a relatively closer geographic distances in all the vehicle pairs and at the same time have relative longer contact distances and smaller centralities in all the vehicle pairs. From Figure 5.2 and Figure 5.3, we can also see that vehicle pairs which have relative high contact(geographic) distances tend to have relative high geographic(contact) distances too. The reason may be that vehicles which have closer frequently visited locations tend to meet

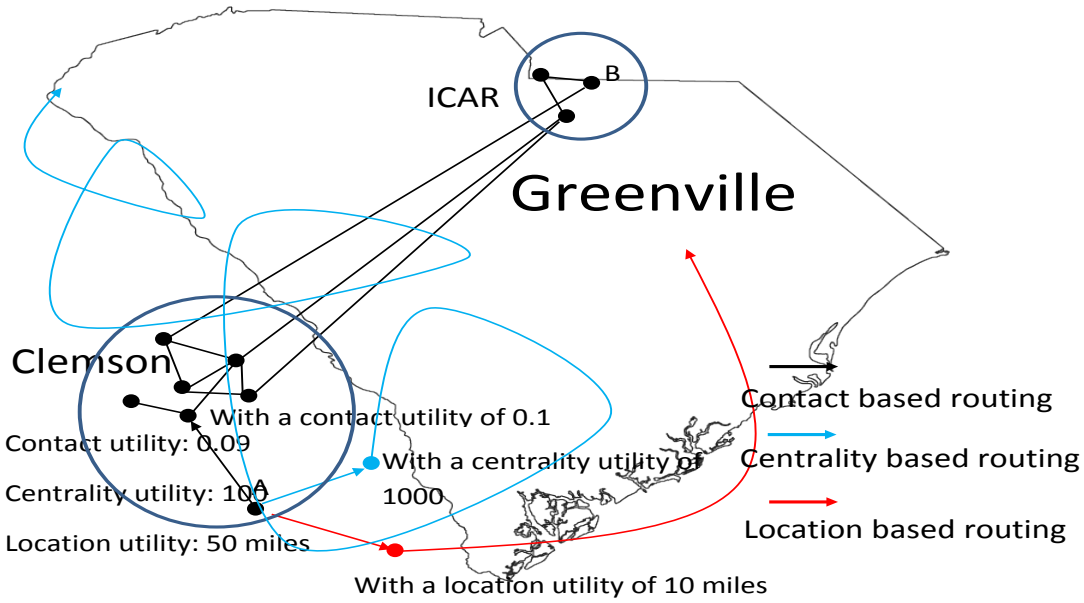


Figure 5.5: An example which shows the correlation between routing performances of different methods and the features of the vehicle pairs

each other with higher possibilities. However, as shown in Figure 5.2 and Figure 5.3, a relative high contact(geographic) distances cannot guarantee relative high geographic(contact) distances since there are many other factors which influence vehicles' contact distances. Finally, we select top 50 vehicle pairs with the shortest delays of centrality routing method. Then, we compare their geographic distances with contact distances and centralities in Figure 5.4. As shown in Figure 5.4, those vehicle pairs tend to have relatively higher centralities in all the vehicle pairs and at the same time have relative longer contact and geographic distances in all the vehicle pairs.

5.1.2 Analysis

Based on the above analysis, we find that there are some correlations between the features (contact distance, geographic distance and centrality) of vehicle pairs with the performances of different routing methods on them. However, as we can see from Figure 5.2, Figure 5.3 and Figure 5.4, such correlations are not always very clear. For example as shown in Figure 5.3, there are still many vehicle pairs in which location routing method performs well but at the same time, with relatively long geographic distances. We further analyze the reason and find that geographic distance is not the only factor that influences the performance of location based routing method. For example as shown in Figure 5.6, suppose that two vehicles A and B are in two communities which named GSP

airport and ALT airport, respectively. Also, vehicles A and B are with a long geographic distance of 100 miles. In this scenario, we should put a particular emphasis on the location based method if A wants to send B a packet even though A and B are with a long geographic distance. Suppose A has a location utility of 100 miles (the vehicle A's frequently visited location is 100 miles away from vehicle B's frequently visited location), a centrality utility of 100 (the vehicle can meet 100 cars a day) and a contact utility of 0.1 (the vehicle meets B with probability 0.1). Suppose vehicle A meets three vehicles: a vehicle with a little higher location utility 50 miles (the vehicle A's frequently visited location is 50 miles away from vehicle B's frequently visited location), a vehicle with a much higher centrality utility 1000 (the vehicle can meet 1000 cars a day) and a vehicle with a much higher contact utility 0.5 (the vehicle meet B with probability 0.5). If A chooses contact method and the vehicle with much higher contact utility 0.5, we may approach B on the contact graph faster by current relay vehicle. However, it cannot be guaranteed that the packet can still easily find the next vehicle with higher contact utility since we cannot make sure that the packet can reach B's community. Therefore, the contact based method may cause a failure although we think we select a suitable vehicle with a little high contact utility in one of the hops. Also, if A chooses vehicle which is very active as the relay vehicle, the relay vehicle may visit a lot of places but it is very likely that the relay vehicle won't visit Clemson at all since the map is very large. Therefore, the centrality based method may also cause a failure. On the contrary, A may just choose a vehicle with a little higher location utility (50 miles) than itself even if 50 miles seems still far away from B. However, we can see from Figure 5.5 that both of their frequently visited locations are on the road 85 although two locations are far from each other. Road 85 is with heavy traffic and unlimited relay vehicles. Therefore, we can make sure there will be more suitable location based relay vehicle in the next hops. Therefore, we would like to choose location based method and a vehicle with only a little higher location utility.

5.1.3 Challenge and Solution

Based on the above analysis, we find that vehicle pairs usually have their unique situation which may be very complex and it is necessary to design a unique routing method for each vehicle pair in VDTNs in order to take advantages of different routing methods simultaneously. However, when it comes to real implementation, it is challenging to find the best routing method for each vehicle pair for the following reason:

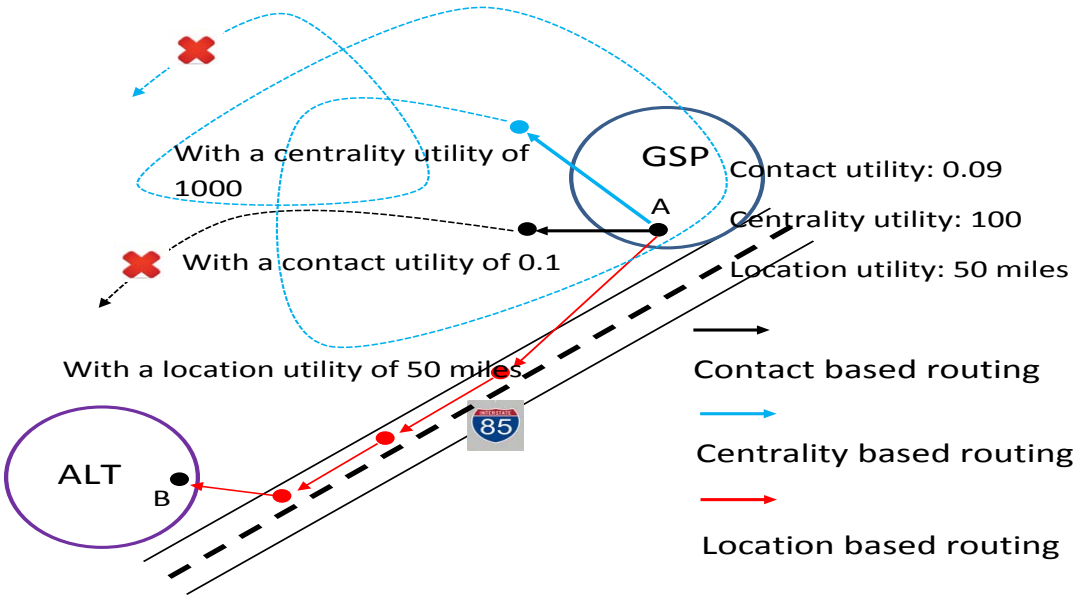


Figure 5.6: An example which shows some random factors in the routing

1. There are many factors which can influence the performances of different routing methods. It's impossible for us to design corresponding strategy for each of them. Furthermore, even if we can list as many as factors and design corresponding strategy for each of them, it still will not be the best way since the situation for each vehicle pair is a combination of different factors.

In order to solve the problem, first of all, we give a threshold to the corresponding utility of each routing method. In addition to continually sending packets to relay vehicles with higher the same kind of utility, the first relay vehicle (so as the following relay vehicles) must have a utility larger than the corresponding threshold. A higher threshold means the corresponding routing method is less suitable for the vehicle pair and we won't choose it unless the current encountered vehicle has a relatively high routing utility (i.e., has very high chance to deliver the packet). A lower threshold means the corresponding routing method is more suitable for the vehicle pair. Therefore, although the current encountered vehicle may not have very high chance to deliver the packet, there must be more chances in the future.

Then, we ignore the detailed factors in the micro-scope and only focus on the routing success rates of different routing methods. We believe that a higher success rate can reflect the underestimate of the corresponding method in the system and therefore, we can decrease the threshold of the

method. On the contrary, we can also increase the threshold of the method. In order to calculate, we let the target vehicle records the numbers of successful delivered copies delivered by different routing methods and let the source vehicle records the numbers of copies delivered by different routing methods. Once the target vehicle receives the copies, it sent a packet to the source vehicle with the information of successful delivered copies delivered by different routing methods.

In this way, we can jump over the difficulty of analyzing different factors from a micro-scope and at the same time, each vehicle pair can learn its own optimized thresholds continually.

5.2 System Design Overview

Before introducing the detailed design of DIAL, firstly, we give an overview of DIAL. DIAL consists of two components: the information fusion based routing method and the adaptive learning framework. As we introduced in Chapter 2, in order to improve the routing performance, current routing methods take advantages of the information shared by centralized infrastructures, which is deviated from the initial goal of building DTNs. By taking advantages of the human beings communication feature we mentioned above, the information fusion based routing method enables DIAL to improve the routing performance by sharing and fusing multiple information without centralized infrastructures. Firstly, two vehicles adopt centrality based method to achieve the first communication. In the first communication, two vehicles store the frequently visited locations and frequently encountered vehicles of each other. Then, once two vehicles need to communicate again, instead of adopting centrality information alone again, they communicate with each other based on the more detailed information of target vehicle which includes the frequently visited locations and frequently encountered vehicle. At the same time, the information of the frequently visited locations and frequently encountered vehicles of target vehicle is updated from time to time during the communication. Therefore, the information fusion based routing method can share multiple information of vehicles in the network and choose different routing method to deliver packets based on the different information. At the same time, in order to balance the numbers of copies of a packet sent by different routing methods and optimize the routing performance, we set each routing method with a threshold of utility based on the overall routing efficiency of the method. In each routing method, the relay vehicles not only need to have a higher utility of the corresponding method than the previous relay vehicles, but also need to have a higher utility than the corresponding threshold of the method.

In information fusion based routing method, the thresholds for different methods are static. However, from observation (i), the performances of different methods can be different on different vehicle pairs. Therefore, we design an adaptive-learning framework which further enables DIAL to design personalized routing strategies for different vehicle pairs without centralized infrastructures. Similar as the information fusion based routing method, by taking advantages of the human beings communication feature, we can calculate the routing success rates of different routing methods which use different information. Then based on the feedback of the success rates, we can analyze the performances of different routing methods and adjust the routing strategies. For example, vehicle B frequently receives the packets sent from Vehicle A. These packets can be delivered by contact based method, centrality based method or location based method. Once vehicle B sends packets to vehicle A, vehicle B sends the numbers of packets successfully delivered by different methods from vehicle A last time at the same time. Then vehicle A can calculate the success rates based on the numbers of packets successfully delivered by different methods and adjust thresholds for different routing methods accordingly to give more preference to the method that can lead to the highest success rate. The routing strategy can be self-adaptive in the adaptive-learning framework as shown in Figure 1.6. Therefore, we can provide different vehicle pairs with different routing strategies and at the same time, the routing strategies can be continually improved according to the feedbacks of the routing performances.

In the following part of this section, we introduce the detailed information fusion based routing method and adaptive-learning framework, respectively.

5.3 Information Fusion based Routing Method

In the information fusion based routing method, we first introduce the initial routing method of delivering a packet from vehicle A to vehicle B when vehicle A and vehicle B never communicated before. Then, we introduce a data structure on vehicle B named *address book* which stores the frequently visited locations and frequently encountered vehicles of vehicles which send packets to vehicle B frequently. Finally, we describe the process of routing process from B to A based on the personalized information of vehicle A stored in the address book of B.

5.3.1 Initial Routing Method

We adopt PeopleRank, which is a centrality based routing method, as the initial routing method of DIAL since PeopleRank can be implemented without centralized infrastructures. Although there are many advanced routing methods beyond PeopleRank, PeopleRank is our ideal option since most advanced routing methods adopt centralized information in order to improve the routing performance. The basic routing process of PeopleRank is as follows.

1. Firstly, we consider vehicles are socially related to each other. Such social relationships can be based on explicit friendships on personal communication. Then, we adopt PageRank algorithm for calculating the centrality of different vehicles. However, in [53], the PageRank value is called PeopleRank value.

2. Consequently, the PeopleRank value is given by

$$PeR(N_i) = (1-d) + d \sum_{N_j \in F(N_i)} \frac{PeR(N_j)}{|F(N_j)|} \tag{5.1}$$

   where N_1, N_2,...,N_n are vehicles, $F(N_i)$ is the set of neighbors that links to N_i, and d is damping factor which is defined as the probability, at any encounter, that the social relation between the nodes helps to improve the rank of these nodes. This means that, the higher the value of d, the more the algorithm accounts for social relation between the vehicles. As a result, the damping factor is a very useful in controlling the weight given to the social relations for the forwarding decision.

3. The PeopleRank value is updated every time when two vehicles encounters.

4. PeopleRank routing is a routing by continually selecting vehicles with higher PeopleRank values.

5.3.2 Building the Address Book

After we have chosen the initial routing method, in order to distributedly share the personalized information of vehicles, in DIAL system, each vehicle maintains an address book by itself. For example as shown in Figure 5.7, the address book stores Location Tables (LTables) and Contact Tables (CTables) of vehicles with ID 1, 2, 3, 4 and so on. A LTable records the frequently visited

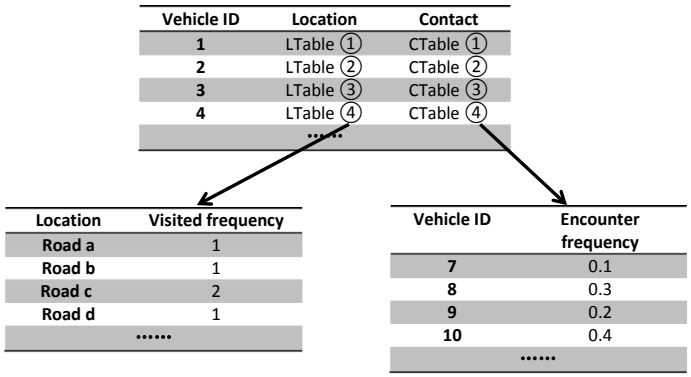


Figure 5.7: An example of address book

road ID and visited frequency of each road of the corresponding vehicle. A CTable records the frequently encountered vehicles and the encounter frequency of the vehicles of the corresponding vehicle. All these information can be used in improving the initial routing method introduced above in the information fusion based routing method. The calculation methods of visited frequency in LTable and encounter frequency in CTable are introduced as follows:

1. The encounter frequency of vehicles is measured by the method in [47]. Specifically, the contact utility is calculated every time when once two vehicles encounter by:

$$C(v_i, v_j) = C_{old}(v_i, v_j) + (1 - C_{old}(v_i, v_j)) \times C_{init}(v_i, v_j) \tag{5.2}$$

   where $C(v_i, v_j)$ is the updated encounter frequency utility; $C_{old}(v_i, v_j)$ is the old encounter frequency utility and $C_{init}(v_i, v_j)$ is the initial value of contact utility of all the vehicle pairs, which is set to a value selected from $(0, 1)$. This definition ensures that the two vehicles with a high encounter frequency have a larger encounter frequency utility.

2. The visited frequency of locations is measured by our previous method in [73]. The basic idea is as follows:

   (a) Firstly, we divide road map to small road sections which can be denoted by road intersections.

   (b) Then each vehicle keeps recording the number of visiting times on each road sections.

5.3.3 Maintaining the Address Book

After a vehicle A has calculated its own address information, vehicle B build and maintain the address information of vehicle A as follows:

1. Once vehicle A delivers a packet to vehicle B, vehicle A delivers its address information to vehicle B with the packet.

2. Once vehicle B receive a packet and address information from vehicle A, vehicle B checks its address book. If vehicle A is in the address book, go to Step (3); otherwise, go to Step (4).

3. Vehicle B updates the address information of vehicle A by the new address information and change the corresponding updated time to the current time stamp.

4. Vehicle B deletes the oldest address information from address book and add the address information of vehicle A and set the corresponding updated time to the current time stamp.

Besides building and maintaining its address book by the information provided by source vehicles, vehicle B can also maintain its address book by other vehicles as follows:

1. Once vehicle B encounters another vehicle C, vehicle B checks whether there are useful address information of its frequently contact vehicles. If there is useful address information, go to Step (2).

2. Vehicle B checks the corresponding updated time of the address information. If it's later than the address information stored in the address book, update the address information.

5.3.4 Selecting Relay Nodes based on Address Information

After vehicle A has built the address information of vehicle B in its address book, we choose contact, centrality and location based routing methods simultaneously to deliver packets from A to B, which guarantees that the best routing method is considered. However, in a combination of different routing methods, if we meet several vehicles all with higher contact utility first, then all the copies of the packet will be delivered by the same method only. Then, if the source vehicle meets a relay vehicle with very high centrality utility or location utility, the source vehicle will lose the chance to deliver the packet by those vehicles since each packet can only have limited copies in order to avoid the congestion. For example as shown in Figure 5.8, a larger size of the circle presents a

larger opportunity source vehicle meet to successfully deliver the packet. Suppose each packet can have 5 copies for routing and we continually send the copies to vehicles with any kind of utilities which are higher than current relay vehicle as shown as the trajectory of the black car in the figure. Then, all the copies are ran out by relay vehicles with a little higher contact utilities (the relatively small black circles). Although the vehicle meet other vehicles with much higher location utility and centrality utility (the relatively big blue and red circles) later, it loses the chance to select them as relay vehicles. In order to conquer this problem, we set each method with a utility threshold. In addition to always forward packets to vehicles with higher utilities, the relay vehicles in each routing method should also have larger corresponding utility than its threshold.

To be more specific, firstly, we set three thresholds: centrality threshold, contact threshold and location threshold as follows:

1. **Contact threshold (Thr_{con}):** We define that the contact utility of the relay vehicle must be large than Thr_{con} for source vehicle to deliver a copy of a packet to the relay vehicle for contact based routing.

2. **Location threshold (Thr_{loc}):** We define that the frequently visited location between relay vehicle and target vehicle must be smaller than $\frac{1}{Thr_{loc}}$ for source vehicle to deliver a copy of a packet to the relay vehicle for location based routing.

3. **Centrality threshold (Thr_{cen}):** We define that the centrality utility of the relay vehicle must be large than Thr_{cen} for source vehicle to deliver a copy of a packet to the relay vehicle for centrality based routing.

As shown in Figure 5.8, in DIAL, we set thresholds to different methods which can guarantee that the copies can be left for those vehicles with very high utilities met later which are denoted as the bigger blue and red circles as shown as the trajectory of the red car in the figure. Therefore, we can fuse the different kinds of information and improve the routing efficiency.

Once we set the thresholds for different routing methods, vehicle B can adopt different routing method to deliver packets to A simultaneously based on the following steps:

1. Once a packet is generated by vehicle B, vehicle B checks its address book. If the target vehicle A is in the address book, go to Step (3); otherwise, go to Step (2).

2. The current relay vehicle adopts the initial routing method to select next relay vehicle.

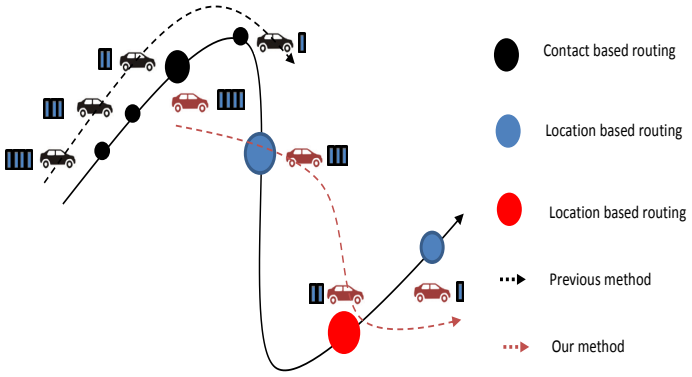


Figure 5.8: Routing with combination of different methods

3. The current relay vehicle adds the location and contact information of vehicle A to the packet and then selects a vehicle as relay vehicle which has larger centrality utility than centrality threshold, larger contact utility than contact threshold or larger location utility than location threshold.

4. If a relay vehicle has more than one utility which is higher than the threshold, we pick the method which is used with smaller number of times in order to balance the load caused by different methods.

5.4 Adaptive-learning Framework

In adaptive-learning framework, we first introduce a data structure named *routing strategy table* stored on each vehicle. The routing strategy table stores the thresholds of different utilities which are corresponding to different routing methods to different target vehicles. Then, we introduce the detailed method for maintaining the thresholds in the routing strategy table. Finally, we describe the whole DIAL routing process.

5.4.1 Building and Maintaining Routing Strategy Table

In the previous section, we set the centrality threshold, contact threshold and location threshold to constant values. Obviously, such a strategy is not ideal since optimal thresholds can be different from one vehicle pair to another vehicle pair. For example as shown in Figure 5.9, vehicle A is inactive and 0.1 is a high enough contact threshold for delivering packets to vehicle A, while

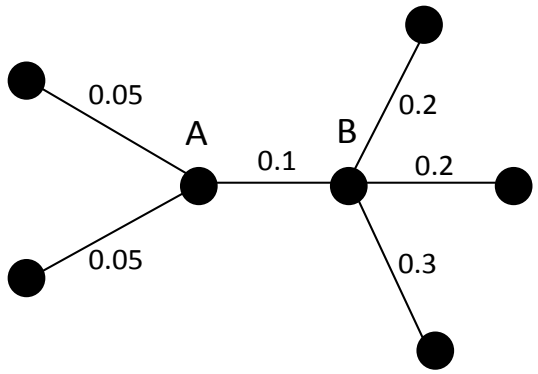


Figure 5.9: An example of the routing process

Vehicle ID	Thresholds		
	Contact	Centrality	Location
B	0.1	1.7	5
C	0.4	2.5	7
D	0.2	1.6	3
	.....		

Figure 5.10: An example of strategy table of vehicle A

vehicle B is active and maybe 0.2 can be a good enough contact threshold for delivering packets to vehicle B. Another is that even for the same target vehicle, the optimal threshold can be different due to the source vehicles are different. Suppose the target vehicle is B. If the source vehicle is A, then the contact threshold may be small since A is inactive. However, if the source vehicle is B, then the contact threshold may be bigger since B meets a lot of vehicles and has more chances to meet a more frequent encountered vehicle. The same thing can happen to centrality threshold and location threshold. In order to set different thresholds for different vehicle pairs, in the adaptive-learning framework, a vehicle A maintains a routing strategy table as shown in Figure 5.10. The routing strategy table stores the thresholds for vehicle A to deliver packets to vehicles with IDs B, C, D and so on.

However, as we mentioned in Section 5.1, it is difficult to predict the routing performances of different routing methods on different vehicle pairs. To solve this problem, in the adaptive-learning framework, instead of applying features of vehicle pairs to predict the performances of different routing methods on different vehicle pairs, we consider the routing process and all the features of vehicle pairs as a black box. By taking advantages of the human beings communication feature that most interactions are generated by pairs of people who interacted often previously, the adaptive-learning framework tests the routing performances of different routing methods on different vehicle pairs. Then based on the feedback of the tests, the adaptive-learning framework can adjust the thresholds of utilities for different routing methods.

To be more specific, the source vehicle A remembers the number of copies sent by different routing methods to target vehicle B. At the same time, target vehicle B remembers the number of copies successfully delivered by different routing method from source vehicle A and send it back

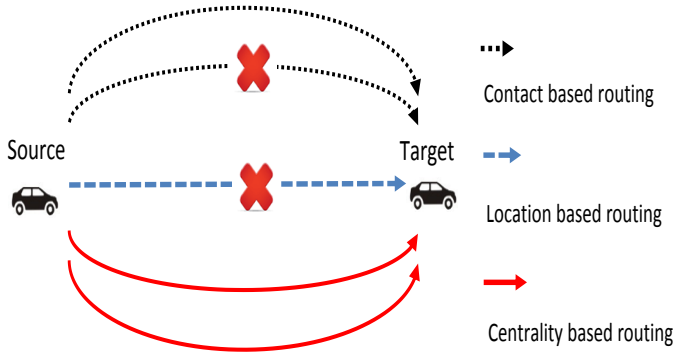


Figure 5.11: An example of strategy table of vehicle A

to source vehicle A with other packets. Then, vehicle A can calculate the success rates of different routing methods on itself. If the success rate of a routing method is higher than others, it means that the threshold of that utility is too low and a lot of copies delivered to vehicles with higher utility than the threshold are wasted. Therefore, we increase the threshold. Otherwise, if the success rate of a routing method is lower than others, it means that the threshold of that utility is too low. Therefore, we decrease the threshold. Here, we define the success rate of a specific method as:

$$CU_{con/loc/cen} = \frac{|S_{con/loc/cen}|}{|SS_{con/loc/cen}|} \tag{5.3}$$

where $CU_{con/loc/cen}$ is the success rate of contact, location and centrality based routing methods, respectively; $|S_{con/loc/cen}|$ is the set of copies sent by a specific method and $|SS_{con/loc/cen}|$ is the set of copies successfully delivered by a specific method. For example as shown in Figure 5.11, there are 6 copies which are tried to be delivered, in which 2 of them are tried to be delivered by contact based routing, 1 of them are tried to be delivered by location based routing and 2 of them are tried to be delivered by centrality based routing, respectively. Finally, 1 of them is successfully delivered by contact based routing and 2 of them are successfully delivered by centrality based routing, respectively. Therefore, we have $CU_{con} = \frac{1}{2} = 0.5$, $CU_{loc} = \frac{0}{1} = 0$ and $CU_{cen} = \frac{2}{2} = 1$.

Based on the calculated success rates, we adjust the thresholds of the three different methods as follows:

$$Thr_{con}^{new} = Thr_{con}^{old} + (CU_{con} - M) \times \Delta_{con} \tag{5.4}$$

$$Thr_{loc}^{new} = Thr_{loc}^{old} + (CU_{loc} - M) \times \Delta_{loc} \tag{5.5}$$

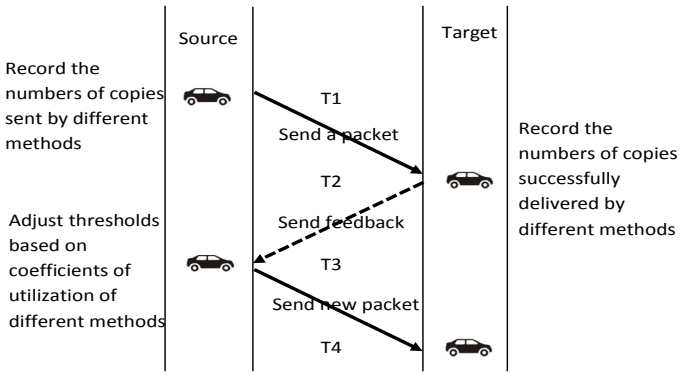


Figure 5.12: An example of strategy table of vehicle A

$$Thr_{cen}^{new} = Thr_{cen}^{old} + (CU_{cen} - M) \times \Delta_{cen} \tag{5.6}$$

where Thr_{con}^{new}, Thr_{loc}^{new} and Thr_{cen}^{new} are the new thresholds of contact, location and centrality based routing methods, respectively. Thr_{con}^{old}, Thr_{loc}^{old} and Thr_{cen}^{old} are the old thresholds of contact, location and centrality based routing methods, respectively. Δ_{con}, Δ_{loc} and Δ_{cen} are the new increments of thresholds of contact, location and centrality based methods, respectively. M is the median of the three thresholds.

To sum up, as shown in Figure 5.12 the routing strategy table is built and maintained by the following steps:

1. Initially, vehicle A gives the thresholds to all the vehicles as constants.
2. Once vehicle A is sending a packet to vehicle B, vehicle records the number of copies sent out by different methods, respectively, as shown in time $T1$ of Figure 5.12.
3. Once vehicle B receives the copies of the packet sent by A, vehicle B records the numbers of copies successfully delivered to itself by different methods, respectively, as shown in time $T2$ of Figure 5.12.
4. Once vehicle A receives the feedback sent by vehicle B, vehicle A adjusts the thresholds of different methods by Formula 5.4, Formula 5.5 and Formula 5.6, respectively, as shown in time $T3$ of Figure 5.12.

5.4.2 Detailed DIAL Routing Process

Based on the above description, now we give a detailed DIAL routing process as follows:

1. Once a packet is generated by vehicle B, vehicle B checks its address book. If the target vehicle A is in the address book, go to Step (3); otherwise, go to Step (2).

2. The current relay vehicle adopts the initial routing method to select next relay vehicle. Then go to Step (5).

3. The current relay vehicle checks strategy table. If there is a routing strategy to target vehicle A, update the thresholds with the values in strategy table; otherwise use the initial values of the thresholds. Then go to Step (4)

4. The current relay vehicle adds the location and contact information of vehicle A to the packet and then selects a vehicle as relay vehicle which has larger centrality utility than centrality threshold, larger contact utility than contact threshold or is going to visit the locations vehicle A visits more frequent than location threshold.

5. The current relay vehicle checks its address book. If the target vehicle A is in the address book, go to Step (3); otherwise, go to Step (2).

5.5 Performance Evaluation

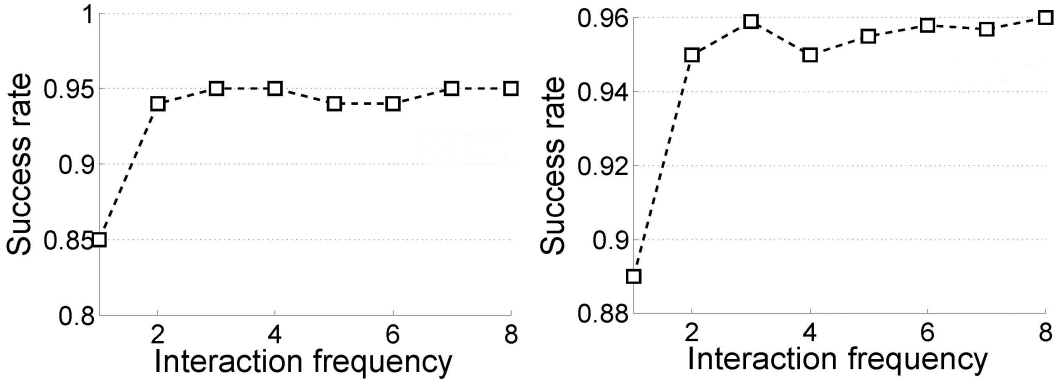
In this section, we evaluate the performance of DIAL and compare it with other methods. We conduct the trace-driven experiments on both the *Roma* and *SanF* traces. In order to evaluate on continually interacting between vehicles on a long term, we recursively set the states of vehicles to the beginning of the trace data and replay the trace data once the trace data run out since the durations of the trace data are not long enough. Based on the above experiment environment, we use the following metrics to evaluate the routing performance:

1. *Success rate:* The percentage of packets that successfully arrive at their target vehicles.

2. *Average delay:* The average time per packet for successfully delivered packets to reach their target vehicles.

Our evaluation is divided to two aspects:

1. From a micro-scope, we measure the routing performance of DIAL with different interaction frequencies since DIAL is designed based on the fact that most interactions are generated by pairs of people who interacted often previously.



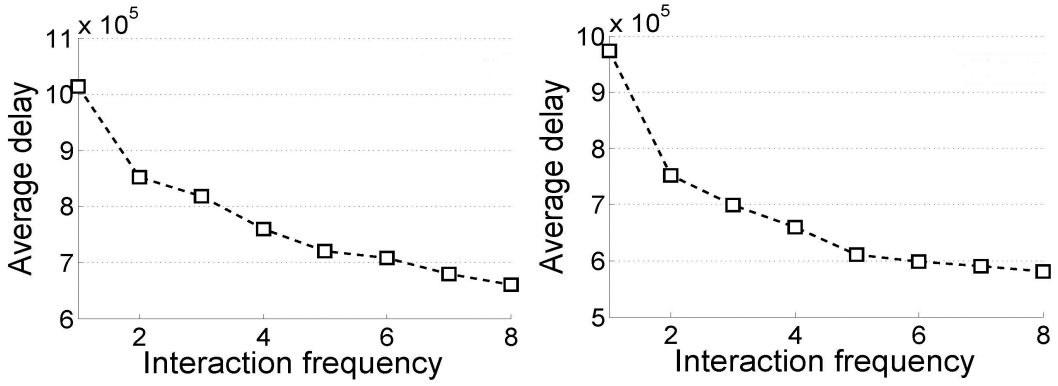
(a) *Roma* (b) *SanF*

Figure 5.13: The success rate vs. the interaction frequency

2. From a macro-scope, we compare the performance of DIAL with AAR [73], PeopleRank [53] and PROPHET [47] methods. AAR represents location based routing method. PeopleRank represents centrality based routing method. PROPHET represents contact based routing method. The details of the methods are introduced in Chapter 2.

5.5.1 Performance Comparison with Different Parameters

It is obvious that DIAL is influenced by the interaction frequency. Therefore, firstly, we analyze the influence of interact frequency on the performance of DIAL. In order to test the change of the routing performance when vehicle pairs continually interact with each other. We randomly select 100 vehicle pairs and continually generate packets between each of them. Suppose that a vehicle pair interact with each other one time is one time of the interactions. Figure 5.13 shows the change of success rates and average delays of DIAL with the increasing of the times of the interactions between each vehicle pair. As shown in Figure 5.13, the success rate is significantly improved at the first several interactions. The reason is that, in the first time of interactions, the successfully delivered packets send the useful information from source vehicles to target vehicles. Then, the target vehicles send packets back to their source vehicles by taking advantages of the information came from the first interaction. Therefore, some of the vehicle pairs which initially cannot be delivered from target vehicle to source vehicle can successfully deliver the packet in the second time of interactions. Therefore, the success rate is significantly improved in the second time of interactions. By taking advantages of the information brought by first few interactions, the routings on most vehicle pairs which the routings are only successful in one side have been improved.



(a) *Roma* (b) *SanF*

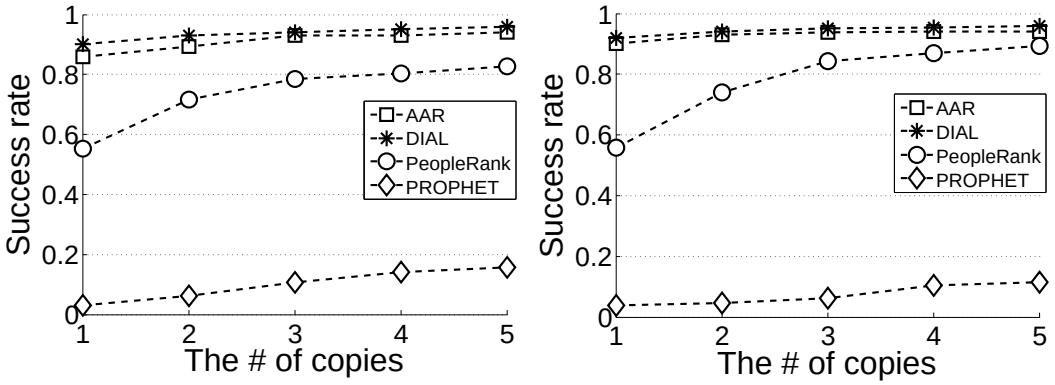
Figure 5.14: The average delay vs. the interaction frequency

Therefore, in the next following interactions, the success rate is not changing too much since routings on those vehicle pairs which the routings are failed in both sides cannot take advantages of DIAL.

On the contrary, the average delay is continually decreasing with the increasing of the times of the interactions for a long time as shown in Figure 5.14. The reason is that, in the first few times of interactions, by taking advantages of the location and contact information brought, the success rate and average delay can be improved obviously. In the new following interactions, although the routings on the vehicle pairs which the routings were failed in both sides cannot be improved, the vehicle pairs which are successfully interacted can still optimize their routing strategies based on our adaptive learning framework. At the same time, for contact, location and centrality based routing methods, the routing method with higher success rate usually leads to a shorter average delay. Therefore, the average delay can be continually improved.

5.5.2 Performance Comparison with Other Methods

Then, we compare DIAL with other routing methods. In the evaluation of previous methods, the packets are randomly generated to evaluate the performance of their methods. However, the interactions between people follows a special pattern. Therefore, in order to improve the accuracy of the evaluation, instead of randomly generating the packets, we reproduce a series of packets which follow the special pattern in [51] and most interactions are generated by vehicle pairs who interacted often previously.



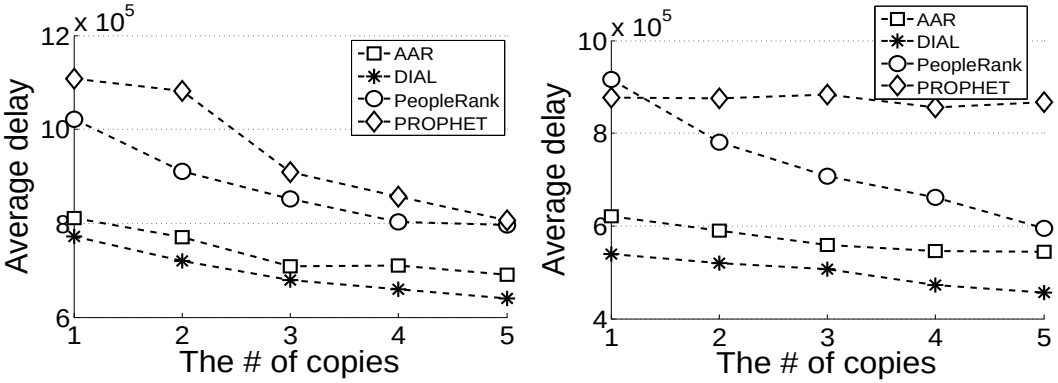
(a) *Roma* (b) *SanF*

Figure 5.15: The success rate vs. different number of copies

5.5.2.1 Performance comparison with different numbers of copies

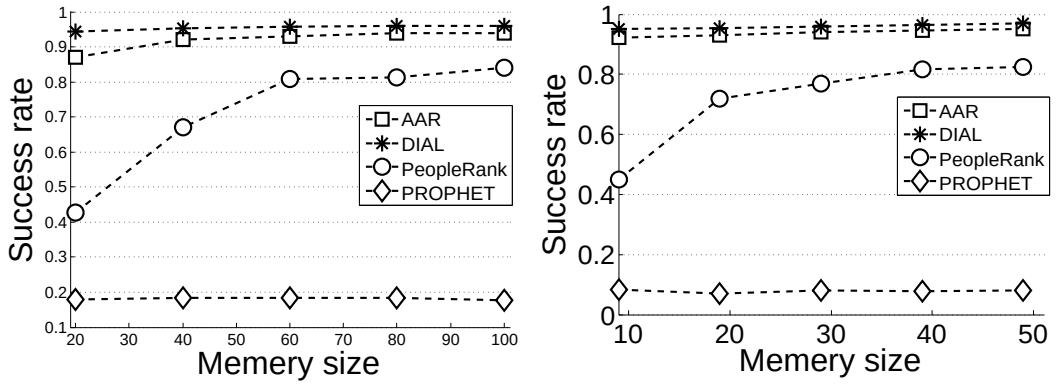
First, we compare the success rates and average delays with different numbers of copies of each packet. Figure 5.15 shows the success rates with different numbers of copies per packet on *Roma* and *SanF* traces, respectively. Generally, the performances follow DIAL > AAR > PeopleRank > PROPHET. The performance of AAR is better than PeopleRank since AAR uses the global information of each vehicle's frequently visited locations. DIAL performs slightly better than AAR since we take advantages of the human beings communication feature and learn the best routing strategy for each vehicle pair. PROPHET performs the worst, since it is difficult to encounter a vehicle that has a high probability to encounter the destination vehicles in the VDTNs. Although DIAL has similar performance as AAR, AAR uses the global information which is not easy to be implemented in reality. However DIAL achieve the similar success rate in a totally distribute way which is easy to be implemented in reality.

Figure 5.16 shows the average delays with different numbers of copies per packet on *Roma* and *SanF* traces, respectively. Generally, the average delays follow PROPHET > AAR > PeopleRank > DIAL. The delay of PROPHET is the largest, since the relay vehicles need to wait a long time to encounter a vehicle that has a high probability to encounter the destination vehicles in the VDTNs. The delay of DIAL is the smallest, since we continually optimize the routing strategy for each vehicle pair during the routings. Based on the above evaluation, we find that the success rate and average delay of DIAL are both improved. Although the improvement of success rate is not significant, DIAL method does not rely on global information which makes it easier to be implemented in reality. Also, the average delay of DIAL is much shorter than other methods.



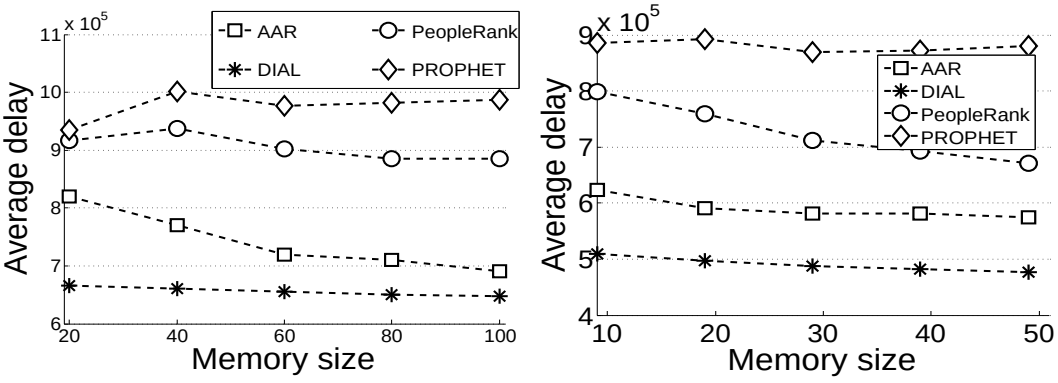
(a) *Roma* (b) *SanF*

Figure 5.16: The average delay vs. different number of copies



(a) *Roma* (b) *SanF*

Figure 5.17: The success rate vs. different memory sizes



(a) *Roma* (b) *SanF*

Figure 5.18: The average delay vs. different memory sizes

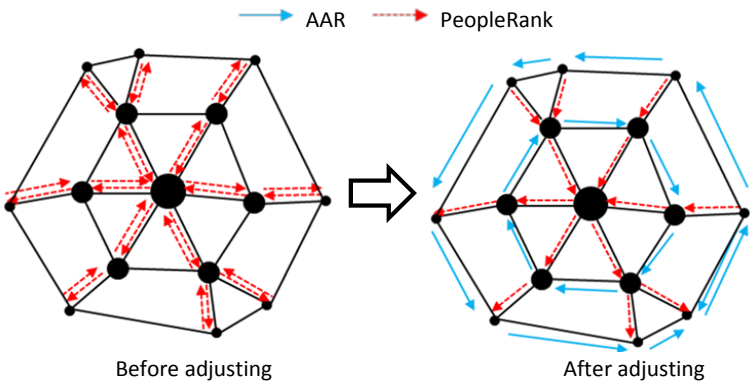


Figure 5.19: An example of load balance function of DIAL

5.5.2.2 Performance comparison with different memory sizes

Then, we compare the success rates and average delays with different memory sizes of the vehicles and we suppose 1 unit of memory can store 1 packet. Figure 5.17 and Figure 5.18 shows the success rates and average delays with different memory sizes, respectively. Generally, the sensitivities of different methods to the memory sizes follow PeopleRank>AAR>DIAL>PROPHET. The performance of PeopleRank is very sensitive to the memory size, since all the packets tend to be forwarded to few vehicles with very high PeopleRank values and the limited memory size can significantly influence the routing process negatively. PROPHET is insensitive to the memory size, since the packets only tend to find those specific vehicles with high probability to encounter the target vehicles, which guarantees load balance. However, PROPHET generates low success rate due to the same reason. AAR delivers a packet relies on many relay vehicles from one road intersection to another road intersection and if vehicles' memory is limited, some packets may lose the chance for delivery. Therefore, AAR is still relatively sensitive. DIAL method is relatively insensitive comparing with AAR and PeopleRank since we adopt the adaptive-learning framework which will adjust the routing strategies from time to time. For example as shown in Figure 5.19, At the beginning time, PeopleRank sends all the packets to the central vehicle A with the biggest PeopleRank value. Since the memory of the central node is limited, the success rate of PeopleRank is decreased when more and more packets are coming. While at the same time, The success rate of AAR is not decreased since a lot of normal vehicles which don't have large PeopleRank value have enough memories for delivery. Based on the mechanism of adaptive-learning framework, the centrality threshold will be increased and location utility will be decreased. Therefore, in the next round, less copies of packets

will be sent to those busy vehicles with high PeopleRank value and more copies will be sent to those idle vehicles. As a result, the overall performance will be improved by a load balance function of adaptive-learning framework.

Chapter 6

Conclusions and Future Work

In this dissertation, we proposed three routing methods for VDTNs. Firstly, we measured the social network features of important nodes, community structure and fractal structure feature of the community of two VNET traces. Then, by fully utilizing the social network features in VNETs, we proposed SPread, an efficient multi-copy routing algorithm. SPread carefully assigns different copies of each packet to different communities which are close to the destination community. Then, each copy can search the destination community through different weak ties, which can enhance the efficiency of current multi-copy routing algorithms. For the routing of each copy, current routing algorithms either fail to exploit each node's reachability information to different nodes (centrality based methods) or simply use single-hop reachability information, e.g., similary and probability, (community based methods). In order to overcome the above drawbacks, inspired by personalized PageRank algorithm, we designed new algorithms for calculating multi-hop reachability of vehicles to different communities and vehicles dynamically. Therefore, the routing efficiency of each copy can also be enhanced. The trace-driven simulation demonstrates that SPread can significantly improve the multi-copy routing efficiency and has a highest success rate and lowest average delay in comparison with other algorithms.

Further, by taking advantage of the unique features of VNETs, we proposed Active Area based Routing method (AAR). Instead of pursuing the target vehicle on the entire VDTN area, AAR spreads copies of a packet to the active sub-areas of the target vehicle where it visits frequently and restricts each copy in its responsible sub-area to search the target vehicle based on contact frequency. The trace-driven simulation demonstrates that AAR has a highest success rate and lowest average

delay in comparison with other algorithms. In our future work, we will discuss the possibility of routing in VDTNs without the help of road side units.

Finally, by taking advantages of the human beings communication feature that most interactions are generated by pairs of people who interacted often previously, we proposed DIAL, an efficient VDTNs routing method. DIAL has two components: information fusion based routing method and adaptive-learning framework. The information fusion based routing method enables DIAL to improve the routing performance by sharing and fusing multiple information without centralized infrastructures. Furthermore, based on the information shared by information fusion based routing method, the adaptive-learning framework enables DIAL to design a personalized routing strategies for different vehicle pairs without centralized infrastructures. Therefore, DIAL can not only share and fuse multiple information of each vehicle without centralized infrastructures, but also dynamically each vehicle pair with personalized routing strategy dynamically. The trace-driven simulation demonstrates that DIAL can slight improve the VDTNs routing success rate comparing with previous routing method which is based on centralized information. At the same time, DIAL significantly improve the VDTNs routing average delay and enables the routing system to dynamically balance the loads among vehicles.

In the future, we will explore the possibility of routing in VDTNs with the help of road side units. Without a centralized infrastructure, the routing performance cannot be guaranteed. Thus, we will also try to explore the possibility to build a hybrid network among vehicles incorporating an infrastructure, which can improve the performance and at the same time decrease the cost by only using a centralized infrastructure.

Bibliography

[1] Graphi. http://www.thebigmoney.com/.

[2] Reka Albert, Hawoong Jeong, and Albert-Laszlo Barabasi. The internet's achilles' heel: Error and attack tolerance of complex networks. *Nature*, 406:200–0, 2000.

[3] Yuichi Asahiro, Refael Hassin, and Kazuo Iwama. Complexity of finding dense subgraphs. *Discrete Applied Mathematics*, 2002.

[4] Bahman Bahmani, Ravi Kumar, and Sergei Vassilvitskii. Densest Subgraph in Streaming and MapReduce. *PVLDB*, 2012.

[5] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 1999.

[6] Albert-Laszlo Barabasi and Zoltan N. Oltvai. Network biology: Understanding the cells functional organization. *Nat Rev Genet*, 5:101–113, 2004.

[7] G. Bianconi and A L. Barabasi. Bose-einstein condensation in complex networks. *Phys. Rev. Lett.*, 86:5632–5635, 2001.

[8] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of community hierarchies in large networks. *CoRR*, abs/0803.0476, 2008.

[9] United States Census Bureau. Annual estimates of the population of metropolitan and micropolitan statistical areas. *2011 Population Estimates*, 2012.

[10] John Burgess, Brian Gallagher, David Jensen, and Brian Neil Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *Proc. of INFOCOM*. IEEE, 2006.

[11] Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. *APPROX*, 2000.

[12] Jie Chen and Yousef Saad. Dense subgraph extraction with application to community detection. *IEEE Trans. Knowl. Data Eng.*, 2012.

[13] Jingchun Chen and Bo Yuan. Detecting functional modules in the yeast protein-protein interaction network. *Bioinformatics*, 2006.

[14] Kang Chen and Haiying Shen. Dtn-flow: Inter-landmark data flow for high-throughput routing in dtns. In *Proc. of IPDPS*, pages 726–737. IEEE, 2013.

[15] Boris V. Cherkassky and Andrew V. Goldberg. On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, 1997.

[16] Fan Chung and Linyuan Lu. Annals of combinatorics. *Annals of combinatorics*, 6:125–145, 2002.

[17] Fan Chung and Linyuan Lu. The average distances in random graphs with given expected degrees. *PNAS*, 99:15879–15882, 2002.

[18] Elizabeth M. Daly and Mads Haahr. Social network analysis for routing in disconnected delay-tolerant manets. In *Proc. of MobiHoc*, pages 32–40. ACM, 2007.

[19] S. N. Dorogovtsev, J. F. Mendes, and A. N. Samukhin. Structure of growing networks with preferential linking. *Phys. Rev. Lett.*, 85:4633–4636.

[20] J. Duch and A. Arenas. Community detection in complex networks using extremal optimization. *Physical Review E*, 2005.

[21] Ahmed Elwhishi and Pin-Han Ho. Sarp - a novel multi-copy routing protocol for intermittently connected mobile networks. In *Proc. of GLOBECOM*, pages 1–7. IEEE, 2009.

[22] P. Erdos and A. Renyi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci.*, 7:17, 1960.

[23] Uriel Feige, Guy Kortsarz, and David Peleg. The dense k-subgraph problem. *Algorithmica*, 2001.

[24] David Gibson, Ravi Kumar, and Andrew Tomkins. Pregel: a system for large-scale graph processing. In *Proc. of VLDB*, 2005.

[25] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *PNAS*, 99(12):7821–7826, 2002.

[26] A. V. Goldberg. Finding a maximum subgraph. *Technical Report*, 1984.

[27] M.S. Granovetter. The strength of weak ties. *American Journal of Sociology*, (78):1360–1380, 1973.

[28] R. Guimer, L. Danon, A. Daz-Guilera, F. Giralt, and A. Arenas. Self-similar community structure in a network of human interactions. *Physical Review E*, 2003.

[29] Alexander Gutfraind, Lauren Ancel Meyers, and Ilya Safro. Multiscale network generation. *CoRR*, abs/1207.4266, 2012.

[30] Yang Hanxin, Wang Binghong, and liu Jianguo. Step-by-step random walk network with power-law clique-degree distribution. *CHIN.PHYS.LETT.*, 25.7, 2008.

[31] Taher H. Haveliwala. Topic-sensitive pagerank. In *Proc. of WWW*, pages 517–526. ACM, 2002.

[32] Bian Hong and Yu Haizheng. An Efficient Control Method of Multi-copy Routing in DTN. In *Proc. of NSWCTC*, pages 153–156. IEEE, 2010.

[33] Theus Hossmann, Thrasyvoulos Spyropoulos, and Franck Legendre. Know Thy Neighbor: Towards Optimal Mapping of Contacts to Social Graphs for DTN Routing. In *Proc. of INFOCOM*, pages 866–874. IEEE, 2010.

[34] Pan Hui, Jon Crowcroft, and Eiko Yoneki. Bubble rap: Social-based forwarding in delay-tolerant networks. *IEEE Trans. Mob. Comput.*, 10(11):1576–1589, 2011.

[35] H. Jeong, S.P. Mason, A.L. Barabasi, and Z.N. Oltvai. Lethality and centrality in protein networks. *Nature*, 2001.

[36] David R. Karger and Clifford Stein. A new approach to the minimum cut problem. *J. ACM*, 1996.

[37] P. L. Krapivsky and S. Redner. Organization of growing random networks. In *Phys. Rev. E*, volume 63,066123, 2001.

[38] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Trawling the web for emerging cyber-communities. In *Proc. of WWW*, 1999.

[39] Andrea Lancichinetti, Mikko Kivelä, Jari Saramäki, and Santo Fortunato. Characterizing the community structure of complex networks. *CoRR*, abs/1005.4376, 2010.

[40] Ilias Leontiadis and Cecilia Mascolo. Geopps: Geographical opportunistic routing for vehicular networks. In *Proc. of WOWMOM*, pages 1–6. IEEE, 2007.

[41] Jure Leskovec, Deepayan Chakrabarti, Jon M. Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker graphs: An approach to modeling networks. *Journal of Machine Learning Research*, 11:985–1042, 2010.

[42] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proc. of KDD*, 2005.

[43] Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *TKDD*, 2007.

[44] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6:29–123, 2009.

[45] Feng Li and Jie Wu 0001. Mops: Providing content-based service in disruption-tolerant networks. In *Proc. of ICDCS*, pages 526–533. IEEE Computer Society, 2009.

[46] X Li and G. Chen. A local world evolving network model. *Physics Letters A*, 328:274–286, 2003.

[47] Anders Lindgren, Avri Doria, and Olov Scheln. Probabilistic routing in intermittently connected networks. *Mobile Computing and Communications Review*, 7:19–20, 2003.

[48] J gila Bitsch Link, Daniel Schmitz, and Klaus Wehrle. Geodtn: Geographic routing in disruption tolerant networks. In *Proc. of GLOBECOM*, pages 1–5. IEEE, 2011.

[49] Bracciale Lorenzo, Bonola Marco, Loreti Pierpaolo, Bianchi Giuseppe, Amici Raul, and Rabuffi Antonello. CRAWDAD data set roma/taxi (v. 2014-07-17). Downloaded from http://crawdad.org/roma/taxi/, July 2014.

[50] S. Thomas McCormick, M. Rammohan Rao, and Giovanni Rinaldi. Easy and difficult objective functions for max cut. *Mathematical Programming*, 2003.

[51] Andrew G. Miklas, Kiran K. Gollu, Kelvin K.W. Chan, Stefan Saroiu, Krishna P. Gummadi, and Eyal de Lara. Exploiting Social Interactions in Mobile Systems. Lecture Notes in Computer Science, pages 409–428. Springer, 2007.

[52] S. Milgram. The small world problem. *Psychology Today*, 61:60–67, 1967.

[53] Abderrahmen Mtibaa, Martin May, Christophe Diot, and Mostafa H. Ammar. Peoplerank: Social opportunistic forwarding. In *Proc. of INFOCOM*, pages 111–115. IEEE, 2010.

[54] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, 2004.

[55] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 2004.

[56] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, 2004.

[57] Gnce Keziban Orman, Vincent Labatut, and Hocine Cherifi. An empirical study of the relation between community structure and transitivity. *CoRR*, abs/1207.3234, 2012.

[58] Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. CRAWDAD data set epfl/mobility (v. 2009-02-24). Downloaded from http://crawdad.org/epfl/mobility/, February 2009.

[59] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabási. Hierarchical organization of modularity in metabolic networks. *Science*, 297:1551–1555, 2002.

[60] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A. L. Barabási. Hierarchical organization of modularity in metabolic networks. *Science*, 297:1551–5, 2002.

[61] Hannu Reittu and Ilkka Norros. On the power-law random graph model of massive data networks. *Perform. Eval.*, 55(1-2):3–23, 2004.

[62] Barna Saha, Allison Hoch, Samir Khuller, Louiqa Raschid, and Xiao-Ning Zhang. Dense subgraphs with restrictions and applications to gene annotation graphs. In *Proc. of RECOMB*, 2010.

[63] C. Seshadhri, Tamara G. Kolda, and Ali Pinar. Community structure and scale-free collections of erdös-réyi graphs. *CoRR*, abs/1112.3644, 2011.

[64] Chao Song, Ming Liu, Yonggang Wen, Jiannong Cao, and Guihai Chen. Buffer and switch: An efficient road-to-road routing scheme for vanets. In *Proc. of MSN*, pages 310–317. IEEE, 2011.

[65] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. Efficient routing in intermittently connected mobile networks: the multiple-copy case. *IEEE/ACM Trans. Netw.*, 16(1):77–90, 2008.

[66] Mechthild Stoer and Frank Wagner. A simple min cut algorithm. *ESA*, 1994.

[67] Andrew Symington and Niki Trigoni. Encounter based sensor tracking. In *Proc. of MobiHoc*, pages 15–24. ACM, 2012.

[68] Md. Yusuf Sarwar Uddin, Hossein Ahmadi, Tarek F. Abdelzaher, and Robin Kravets. A low-energy, multi-copy inter-contact routing protocol for disaster response networks. In *Proc. of SECON*, pages 1–9. IEEE, 2009.

[69] Li Wan, Bin Wu, Nan Du, Qi Ye, and Ping Chen. A new algorithm for enumerating all maximal cliques in complex network. In *ADMA*, 2006.

[70] D.J. Watts and S.H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, (393):440–442, 1998.

[71] D.J. Watts and S.H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, pages 440–442, 1998.

[72] D.J. Watts and S.H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 1998.

[73] Bo Wu, Haiying Shen, and Kang Chen. Exploiting active sub-areas for multi-copy routing in vdtns. In *Proc. of ICCCN*, pages 1–10. IEEE, 2015.

[74] Yuchen Wu, Yanmin Zhu, and Bo Li 0001. Trajectory improves data delivery in vehicular networks. In *Proc. of Infocom*, pages 2183–2191. IEEE, 2011.

[75] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *CoRR*, 2012.

[76] Eiko Yoneki, Pan Hui, Shu Yan Chan, and Jon Crowcroft. A socio-aware overlay for publish/-subscribe communication in delay tolerant networks. In *Proc. of MSWiM*, pages 225–234. ACM, 2007.

[77] Hongzi Zhu, Shan Chang, Minglu Li, Kshirasagar Naik, and Sherman X. Shen. Exploiting temporal dependency for opportunistic forwarding in urban vehicular networks. In *Proc. of INFOCOM*. IEEE, 2011.