Fall 1-31-2014

Vehicle re-routing strategies for congestion avoidance

Juan Pan
New Jersey Institute of Technology

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

VEHICLE RE-ROUTING STRATEGIES FOR CONGESTION AVOIDANCE

by
Juan (Susan) Pan

Traffic congestion causes driver frustration and costs billions of dollars annually in lost time and fuel consumption. This dissertation introduces a cost-effective and easily deployable vehicular re-routing system that reduces the effects of traffic congestion. The system collects real-time traffic data from vehicles and road-side sensors, and computes proactive, individually tailored re-routing guidance, which is pushed to vehicles when signs of congestion are observed on their routes. Subsequently, this dissertation proposes and evaluates two classes of re-routing strategies designed to be incorporated into this system, namely, Single Shortest Path strategies and Multiple Shortest Paths Strategies.

These strategies are firstly implemented in a centralized system, where a server receives traffic updates from cars, computes alternative routes, and pushes them as guidance to drivers. The extensive experimental results show that the proposed strategies are capable of reducing the travel time comparable to a state-of-the-art Dynamic Traffic Assignment (DTA) algorithm, while avoiding the issues that make DTA impractical, such as lack of scalability and robustness, and high computation time. Furthermore, the variety of proposed strategies allows the system to be tuned to different levels of trade-off between re-routing effectiveness and computational efficiency. Also, the proposed traffic guidance system is robust even if many drivers ignore the guidance, or if the system adoption rate is relatively low.

The centralized system suffers from two intrinsic problems: the central server has to perform intensive computation and communication with the vehicles in real-time, which can make such solutions infeasible for large regions with many vehicles; and driver

privacy is not protected since the drivers have to share their location as well as the origins and destinations of their trips with the server, which may prevent the adoption of such solutions. To address these problems, a hybrid vehicular re-routing system is presented in this dissertation. The system off-loads a large part of the re-routing computation at the vehicles, and thus, the re-routing process becomes practical in real-time. To make collaborative re-routing decisions, the vehicles exchange messages over vehicular ad hoc networks. The system is hybrid because it still uses a server to determine an accurate global view of the traffic. In addition, the user privacy is balanced with the re-routing effectiveness. The simulation results demonstrate that, compared with a centralized system, the proposed hybrid system increases the user privacy substantially, while the re-routing effectiveness is minimally impacted.

VEHICLE RE-ROUTING STRATEGIES FOR CONGESTION AVOIDANCE

by
Juan (Susan) Pan

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Science

Department of Computer Science

January 2014

APPROVAL PAGE

VEHICLE RE-ROUTING STRATEGIES FOR CONGESTION AVOIDANCE

Juan (Susan) Pan

Dr. Cristian M. Borcea, Dissertation Advisor — Date
Associate Professor, Computer Science, New Jersey Institute of Technology

---

Dr. Guiling Wang, Committee Member — Date
Associate Professor, Computer Science, New Jersey Institute of Technology

---

Dr. Vincent Oria, Committee Member — Date
Associate Professor, Computer Science, New Jersey Institute of Technology

---

Dr. Iulian Sandu Popa, Committee Member — Date
Assistant Professor, Computer Science, University of Versailles Saint-Quentin

---

Dr. Wu Chou, Committee Member — Date
VP, Chief IT Scientist, Huawei Research Lab

BIOGRAPHICAL SKETCH

Author: Juan (Susan) Pan

Degree: Doctor of Philosophy

Date: January 2014

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2014
- Master of Science in Computer Science,
  Tianjin University, Tianjin, China, 2006
- Bachelor of Science in Computer Science,
  Hebei University of Technology, Tianjin, China, 2003

Major: Computer Science

Presentations and Publications:

Juan (Susan) Pan, Iulian Sandu Popa and Cristian Borcea, "A Hybrid Vehicular Traffic Re-routing System for Congestion Avoidance," *IEEE Transactions on Vehicular Technology (to be submitted)*

Juan (Susan) Pan and Cristian Borcea, "Vehicular Sensor Networks," *Chapter in the Handbook of Sensor Networking: Advanced Technologies and Applications, CRC Press (to be submitted)*

Juan (Susan) Pan, Iulian Sandu Popa, Karine Zeitouni, and Cristian Borcea, "Proactive Vehicular Traffic Re-routing for Lower Travel Time," *IEEE Transactions on Vehicular Technology, Vol. 62, No. 8*, 2013

Daniel Boston, Steve Mardenfeld, Juan (Susan) Pan, Quentin Jones, Adriana Iamnitchi, and Cristian Borcea, "Leveraging Bluetooth Co-location Traces in Group Discovery Algorithms," *To appear in Elsevier Pervasive and Mobile Computing Journal , Special Section on Mobile Social Networks*, 2014.

Susan Juan Pan, Mohammad A. Khan, Iulian Sandu Popa, Karine Zeitouni, and Cristian Borcea, "Proactive Vehicle Re-routing Strategies for Congestion Avoidance," *in Proceedings of the 8th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS '12)*, May 2012.

Juan (Susan) Pan, Li Li, Wu Chou, "Real-Time Collaborative Video Watching on Mobile Devices with REST Services," *in Proceedings of the 3th FTRA International Conference on Mobile, Ubiquitous and Intelligent Computing (MUSIC 2012)*, June 2012.

Susan Juan Pan, Daniel Boston, and Cristian Borcea, "Analysis of Fusing Online and Co-presence Social Networks," *in Proceedings of the 2nd IEEE Workshop on Pervasive Collaboration and Social Networking (PerCol 2011)*, March 2011.

Steve Mardenfeld, Daniel Boston, Susan Juan Pan, Quentin Jones, Adriana Iamnitchi, and Cristian Borcea, "GDC: Group Discovery using Co-location Traces," *in Proceedings of the 2nd IEEE Symposium on Social Computing Applications (SCA-10)*, August 2010.

To my husband: David Paglia for all his love and support over the years. He has been my comfort through all the difficult moments.

ACKNOWLEDGMENT

I would truly like to express my most sincere appreciation to my Dissertation Advisor, Dr. Cristian Borcea who has continually challenged my abilities as a researcher over the years. Dr. Borcea has been a constant mentor to me, always dedicating the time to mold my weaknesses into strengths.

Special thanks are given to Dr. Iulian Sandu Popa from University of Versailles Saint-Quentin, France for his great efforts of collaborating with me to accomplish this work. I am grateful for the time he has dedicated to this dissertation and for all his contributions for improving my research.

I thank Dr. Vincent Oria for all his encouragement and assistance early on in this project and many opportunities he had given me to help me grow.

I would like to thank Dr. Guiling Wang for her generous advice. I am very grateful for all the time she took out of her schedule to listen to me and offer valuable advice.

I wish to give my sincere thanks to Dr. Wu Chou for supervising and leading me during my internship. It was an honor to work together with him. The lesson he taught me is essential in several aspects of my career. Finally, I would like to thank Drs. Sandu Popa, Oria, Wang, and Chou for being part of my dissertation committee.

I would like to acknowledge Dr. Li Li who was always patient with me and take the time to answer my numerous questions and brainstorm solutions to various issues that arose while working together with him. I really appreciate his help.

I would like to thank all the faculty and administration of the Computer Science department at NJIT as well as all my colleagues in the Networking Laboratory for their intellectual and moral support through the years.

In the end, I would like to acknowledge my husband, David Paglia; my parents, Zhixin Pan, Xiping Shen; and my parents in law, George Paglia and Lynn Paglia. I'm

blessed for having such a great family. Their unconditional love has always been the strength for me to confront various difficulties towards the path to the goal. Without them, nothing would have been possible.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

Traffic congestion has become an ever-increasing problem worldwide. Congestion reduces efficiency of transportation infrastructure and increases travel time, air pollution, and fuel consumption. In 2010, traffic congestion caused urban Americans to travel 4.8 billion hours more than necessary and to purchase an extra 1.9 billion gallons of fuel, for a congestion cost of $101 billion. It is predicted that by 2015, this cost will rise to $133 billion (i.e., more than $900 for every commuter). The amount of wasted fuel will jump to 2.5 billion gallons (i.e., enough to fill more than 275,000 gasoline tanker trucks) [74]. While congestion is largely thought of as a big city problem, delays are becoming increasingly common in small cities and some rural areas as well. Hence, finding effective solutions for congestion mitigation at reasonable costs is becoming a stringent problem. The thesis of my dissertation is that *more effective vehicle re-routing guidance can be proactively provided to individual drivers in timely manner, based on the collaborative knowledge collected from smart phones or systems embedded in vehicles, to alleviate the effects of congestion on the roads.*

The advances of the emerging sensing and computing technologies enables pervasive development of the Intelligent Transportation System (ITS). ITS aims to enhance the traveler experience by integrating technology and intelligence into the existing transportation infrastructure. Vehicle re-routing system (VRS) technology is a subset of ITS. In the past 30 years, various VRS technologies have been studied and developed around the world utilizing different schemes to achieve lower travel time for drivers. Currently static distributed road-side sensors (e.g., induction loops, video cameras) and vehicles acting as mobile sensors (i.e., using embedded vehicular systems or smart phones) can collect real-time data to monitor the traffic at fine

granularity. For example, the Mobile Millennium project [35] demonstrated that only a low percentage of drivers need to provide data to achieve an accurate traffic view. Others [28, 47] have presented an adaptive traffic light system based on wireless communication between vehicles and fixed controller nodes deployed in intersections.

Unlike a large amount of research which focuses on accurately collecting and calibrating the data (e.g., predicting travel times in specific areas at particular times of day or improving the traffic light cycles), this dissertation focuses on alleviating congestion by providing drivers with better alternative routes in real-time. Implicitly, fuel consumption and pollution will be reduced as well.

The rest of this chapter presents an overview of vehicle re-routing technology in Section 1.1, including the first generation in-car navigation systems, dynamic traffic assignment theory and vehicular network traffic guidance systems. Section 1.2 presents the problem statement addressed by the dissertation. Section 1.3 discusses the challenges of providing re-routing guidance for vehicles. The contributions of this dissertation are presented in Section 1.4, and contributors to this work are recognized in Section 1.5. Finally, Section 1.6 details the structure of this dissertation.

1.1 Vehicle Re-routing Systems

1.1.1 Current In-car Navigation Systems

The first generation in-car navigation systems and web services were limited from the start, as they only considered the road network, but not the traffic conditions, when computing the shortest route to a destination. This was due to the unavailability of traffic data. With the deployment of traffic surveillance infrastructure on more roads (e.g., loop detectors, video cameras), the global population has started to witness the implementation of web-based services/applications that present the drivers with the current view of the traffic and let them decide which route to follow. This is in line with Wardrop's first traffic equilibrium principle [85], which essentially states

that a user-optimum traffic equilibrium is found when drivers make their own selfish decisions (i.e., Nash equilibrium [43]). However, the usefulness of these applications is also limited: (i) they have accurate information mostly about highways and thus are not very useful for city traffic, and (ii) they cannot avoid congested areas and, at the same time, it is known that no true equilibrium can be found under congestion [16]. Non-recurring congestions, which represent over 50% of all congestions [17], are especially problematic, as drivers cannot use their experienced travel times to deal with them.

Recently, companies such as Google [1] and TomTom [2] have started to use infrastructure-based traffic information to compute traffic-aware shortest routes. These solutions are better than just showing the traffic conditions because they can guide drivers as function of other factors such as historic traffic and current weather. While this is a significant advancement, it still only applies to major roads. This problem may be overcome by collecting data from the drivers' smart phones [39, 61].

A more serious issue is that these solutions do not try to avoid congestions explicitly (i.e., they are reactive solutions) and provide the same guidance for all vehicles on the road at a certain moment as function of their destination (i.e., pull model in which drivers query for the shortest route to destination). Therefore, similar to route oscillations in computer networks, could lead to unstable global traffic behavior. When this happens, congestion is switched from one route to another, if significant numbers of drivers follow the guidance.

1.1.2 Dynamic Traffic Assignment

Dynamic Traffic Assignment (DTA) applies mathematical methodologies to model traffic dynamics throughout the road network. The goal of DTA is to compute, for each driver, an optimal route assignment such that no driver can improve his/her travel time by unilaterally shifting to alternative routes [85]. The key challenge is that

when the drivers are assigned certain routes, the travel time along the routes changes due to the increased volume and the limited capacity of the roads, especially when a road is over-saturated. This varied road travel time in turn affects other drivers' route choices. The currently perceived instantaneous travel time does not necessarily equate with the experienced/expected travel time in the future [16]. Therefore, in such highly dynamic conditions, a proper theoretical model of the road capacity and travel time plays a significant role. Currently, most of the popular DTA tools use simulation-based approaches to model the dynamics of the road travel time and compute drivers' route assignments iteratively. Specifically, given certain demand of (origin,destination) pairs, initial route assignments are communicated to each vehicle. Afterwards, in each consecutive step, the route assignments in the previous step are adjusted gradually until a certain convergence criterion is achieved where minimal adjustment is needed.

Although DTA provides analytical and theoretical guidelines for traffic assignment, there is still a significant gap between the theoretical or simulation results and potentially deployable solutions. One major issue is the computation overhead. Each DTA iteration requires a complex simulation process. To achieve moderate accuracy and convergence rate, many iterations are needed, which introduces huge computational overhead. Other issues include: convergence, sensitivity, realism of traffic dynamics, tractability for large scale road networks, capability of providing real-time guidance, behavior in the presence of congestion, ability to function when not all drivers are part of the system, and robustness to drivers who ignore the guidance [16].

1.1.3 Vehicular-based Cooperative Traffic Guidance Systems

Despite significant advances of in-car navigation system (e.g., Garmin, TomTom), web services for route computation (e.g., Google, Microsoft), and dynamic traffic

assignment [16, 51], the global population is still spending a tremendous amount of time in traffic jams.

The emerging advances in sensors and vehicle communication technology (e.g., Vehicular Ad Hoc Networks (VANETs)) provides a powerful platform for a wide range of distributed computing applications such as Vehicular-based Cooperative Traffic Guidance Systems.

In recent years, most new vehicles come already equipped with GPS receivers and navigation systems. Car manufacturers such as Ford, GM, BMW and Toyota have already announced efforts to include significant computing power within their automotive design [8, 73, 83, 84]. After Google revealed its first self-driving car, a number of auto-makers began venturing into this research area. This trend is expected to continue and, in the near future, the number of vehicles equipped with computing technologies and wireless network interfaces will increase dramatically. These vehicles will be able to run network protocols that will exchange messages for safer and more fluid traffic on the roads.

Several protocols have been proposed for implementing vehicular communication. The most promising standard is IEEE 802.11p. The physical layer of 802.11p is a dedicated short-range 5.9 GHz for communication among vehicles and with roadside infrastructure [40].

So far, two classes of Vehicular-based Cooperative Traffic Guidance Systems have been studied: infrastructure-less solutions based on inter-vehicle communication and infrastructure-based solutions relying on the peer-to-peer paradigm.

Trafficview [62] and StreetSmart [20] are examples for the first class. They make use of wireless communication and GPS, which enable vehicles to collect and disseminate traffic information. This provides meaningful data to the drivers. Some other applications include intelligent traffic light adjustment [28] and bio-inspired organic system [69, 76, 80]. In the second category, Peertis [71, 72] discussed an efficient

and scalable architecture that incorporates a P2P overlay into a vehicular network based on cellular Internet communication to provide better service. Unlike the client server architecture discussed in the previous section, these solutions fully utilize VANETs communication power to provide distributed traffic services. Nevertheless, they simply use the collected data for naive sub-optimal shortest path computation which potentially switch congestion from one spot to another. Besides, VANETs have throughput and latency problems for larger scale networks and these problems become worse in highly congested areas. Due to wireless contention, the dropped packets lead to incomplete traffic view, thus to suboptimal decisions.

1.2 Problem Statement

This dissertation addresses the problem of providing efficient re-routing paths for drivers of cars on the roads with signs of congestion in centralized and hybrid (centralized server and VANETs) system. This dissertation argues that the time is ripe for building a proactive, intelligent, and real-time traffic guidance solution based on the dynamic situation in the road network. In this system, vehicles can be viewed as both mobile sensors (i.e., collect real-time traffic data) and actuators (i.e., change their path in response to newly received guidance). The system is cost-effective and easily deployable because it does not require road-side infrastructure; it can work using only smart phones carried by drivers [1]. Where road-side sensors are available, this system can take advantage of them to supplement the data provided by vehicles, thereby building an accurate representation of the global real-time traffic conditions. Periodically, the system evaluates the congestion levels in the road network. When signs of congestion are observed on certain road segments, the system computes proactive, individually-tailored re-routing guidance, which is pushed to vehicles that would pass through the congested segments. It is important to note that the drivers

[1] In the future, once vehicular embedded systems become widespread, they could be used instead of smart phones.

are not "forced" to follow alternative routes: the guidance may or may not be accepted by drivers.

The problem statement is **how to build a traffic re-routing system and re-routing algorithms that will be successful in a real-life deployment?** This system requires software that addresses the following questions: How to monitor the traffic accurately in the presence of low system penetration rate? How to protect drivers' privacy (i.e., location privacy) and, at the same time, achieve an optimal global view of the real-time traffic? How to predict congestion in real-time on different road types, based on data reported from smart phones and existing infrastructure? What algorithms should be used to compute and deliver effective re-routing guidance to individual drivers, before they encounter into congested areas? How to make the system robust and adaptable in the presence of drivers who ignore the provided guidance? How to make the system scalable to potentially millions of vehicles in terms of both computation and communication?

1.3 Contributions of Dissertation

This dissertation introduces a cost-effective and easily deployable vehicular traffic guidance system that reduces the effect of traffic congestions. Then, five re-routing strategies designed to be incorporated in this system are proposed and evaluated:

- Dynamic Shortest Path (DSP), which assigns to each vehicle the current shortest time path to destination;
- A* shortest path with Repulsion (AR*), which modifies the A* shortest path algorithm [32] by considering both the travel time and the paths of the other vehicles (as a repulsive force) in the computation of the shortest path.
- Random k Shortest Paths (RkSP), which computes k-shortest paths for each re-routed vehicle and randomly assigns the vehicle to one of them.
- Entropy Balanced k Shortest Paths (EBkSP), which computes k-shortest paths for each vehicle and assigns the vehicle to the path with the lowest popularity as defined by the path entropy.

- Flow Balanced k Shortest Paths (FBkSP), which computes k-shortest paths for each vehicle and assigns the vehicle to the path that minimizes the impact of traffic flow in a network region.

DSP and AR* are built on top of the classical shortest path algorithms that is adapted to the context of a real-time traffic guidance system. DSP is a basic strategy that updates dynamically the vehicles' routes when there are signs of congestion in the road network, to the current, travel time based shortest path. The simplicity of DSP suggests high computational efficiency. At the same time, DSP may have limited effectiveness in alleviating congestion in difficult circumstances (e.g., if the traffic is very dense) because it may lead to switching congestion from one spot to another. AR* tackles this shortcoming by taking into account the other vehicles' paths in the computation of a new vehicle route. However, the price to pay is increased complexity and therefore, increased computational cost.

To obtain a more flexible trade-off between the effectiveness and the efficiency of the re-routing, a second group of strategies are proposed, that are based on the k-shortest paths algorithm. The proposed is idea is to compute the set of k-best alternative paths for a re-routed vehicle and then assign each vehicle to the path that reduces the road network utilization with respect to the other vehicles' paths. These strategies are expected to be more effective than the simple DSP, since they take into account the other vehicle paths. Yet the KSP strategies should be more computationally efficient than AR*, since they limit the traffic flow optimization to the k alternative paths. Three policies are proposed to select the best path among the k-shortest paths as indicated above, i.e., random selection in RkSP, popularity based selection in EBkSP, and flow balanced selection in FBkSP.

The system was first implemented using a centralized architecture where a centralized server is responsible for both collecting traffic reports and re-routing path computation. The proposed system was extensively evaluated, integrating the five strategies through simulations over two medium-size urban road networks and

across several parameters including the re-routing period, the congestion threshold, the vehicle selection level, the vehicle priority, the number of alternative paths, the drivers' compliance rate, and the penetration rate. Moreover, a tool implementing a state-of-the-art DTA algorithm for traffic optimization was employed, to quantify both the improvement of the travel time provided by DTA and its computational cost. The results indicate that the five strategies significantly decrease the average travel time compared to "no-rerouting", thus lowering the average travel time at least by 2 times in most cases, and up to 5 times in certain cases. Compared to DTA, these strategies yield similar travel times at (much) lower computational costs. Additionally, these strategies are much more scalable with the number of vehicles than DTA. Among the proposed strategies, AR* has the lowest average travel time, but the highest computational cost. EBkSP and FBkSP can achieve comparable travel times as AR*, while demanding lower CPU times for the re-routing computations. DSP has by far the lowest computational cost, but it is also the least efficient at reducing the travel time. Finally, it is worth mentioning that these re-routing strategies are still effective in alleviating congestion even if many drivers ignore the guidance or if the system adoption rate is relatively low, which is important in facilitating the adoption of the system at a large scale.

The main focus of the first part of this dissertation are the routing algorithms in a centralized design. However, as for a practical deployable solution, scalability and privacy are essential. Since a purely centralized system internally suffers from scalability and privacy problems and a fully distributed system can not obtain a full picture of the road network traffic, a hybrid system is proposed. It is called "hybrid" since the system still requires a central server to obtain a global accurate traffic view. A privacy-aware reporting mechanism is designed to send traffic reports probabilistically as a function of the vehicle density on the roads. To measure privacy leakage, each traffic report is associated with a importance factor defined by the entropy of the road

segments. Once signs of congestion are detected by the cental server, only vehicles that reported recently and are close to congestion spots are notified. The traffic view is propagated in VANET and the computation is off-loaded to individual vehicle where distributed re-routing algorithms are executed collaboratively. Specifically, the EBkSP and AR* algorithms are extended to distributed versions, namely dEBkSP and dAR*. In order to achieve similar performance compared to the centralized design, four optimizations techniques are presented: prioritized broadcast, distance-based timer, network coding and k path compression.

Extensive simulation results demonstrate that the privacy leakage is reduced by 90% while only sacrificing 10% of the travel time. Additionally, re-routing frequency is reduced by 25%. Although dAR* exhibits lower travel time, analysis shows that dEBkSP has better scalability since the computation of k shortest paths is evenly distributed to vehicles.

1.4 Structure of Dissertation

The subsequent chapters of this thesis dissertation are structured as follows: Chapter 2 reviews related work. Chapter 3 describes the basic centralized system model and the challenges. Chapter 4 explains the two classes of re-routing strategies and the DTA algorithm used as baseline. Chapter 5 presents the experimental results and analysis for the centralized system. The hybrid system design is presented in Chapter 6. Chapter 7 shows the evaluation results of the hybrid system. The dissertation concludes in Chapter 8.

CHAPTER 2

RELATED WORK

This chapter presents background and related work literature in the domain of vehicle routing services in Section 2.1, efficient K shortest path generation in Section 2.2, dynamic traffic assignment model in Section 2.3, vehicular ad hoc networks in Section 2.4 and location privacy in Section 2.5. The chapter concludes in Section 2.6.

2.1 Existing Vehicle Routing Services

Projects such as Mobile Millennium [39, 90], CarTel [21], JamBayes [38], Nericell [61], and surface street estimation [31] use vehicle probe data collected from on-board GPS devices to reconstruct the state of traffic and estimate shortest travel time. The proposed research moves beyond this idea: instead of investigating the feasibility and accuracy of using mobile phones as traffic sensors, this dissertation focuses on using that information to recommend routes more intelligently, thus, achieving better efficiency in terms of avoiding congestion and reducing travel time.

Services such as INRIX [3] provide real-time traffic information at a certain temporal accuracy, which allows drivers to choose alternative routes if they are showing lower travel times. According to Wardrop's first traffic equilibrium principle [85], this could lead to a user-optimum traffic equilibrium. It is known, however, that no true equilibrium can be found under congestion [42]. Several initiatives have been take in the directions of predicting long-term recurrent and short term non-recurrent congestions [4]. However, the usefulness of these applications is also limited: (i) they have accurate information mostly about highways and thus are not very useful for city traffic, and (ii) they cannot avoid congestions and, at the same time, it is known that no true equilibrium can be found under congestion [16]. Non-recurring congestions,

which represent over 50% of all congestions [17], are especially problematic as drivers cannot use their experienced travel times to deal with them.

2.2 K Shortest Path Generation

A large body of existing route planning research focuses on fast generation of k-shortest paths [53, 75] in highly dynamic scenarios with frequent traffic information updates. In particular, [75] presents transit-node routing and highway-node routing to reduce the average query time and memory requirements. The work in [53] proposes two new classes of approximation techniques (e.g., K-AS-Aggressive, K-AS-Variance,Y-Moderate) that use pre-computation and avoidance of complete recalculations on every update to speed up the processing of continuous route planning queries. However, current instantaneous shortest paths are not necessarily equal to time-dependent shortest paths. These algorithms calculate shortest paths based only on the snapshot of current traffic conditions without considering the dynamic future conditions.

One of the essential properties of the travel time on the road network is the time-dependency. Computing shortest paths in a time varying spatial network is challenging since the edge (i.e., road segment) travel times changes dynamically. In this case, the computation not only considers the instantaneous travel time in one single snapshot of the traffic graph but also the relationship among the consecutive snapshots across time. George et at. [27] demonstrated a faster greedy time-dependent shortest path algorithm (SP-TAG) by using a Time Aggregated Graph (TAG) data structure instead of the time-expanded graph. SP-TAG saves storage and computation cost allowing the properties of edges and nodes to be modeled as a time series instead of replicating nodes and edges at each time unit. While algorithms such as SP-TAG provide insights into the dynamics of traffic network, two obstacles remain besides increased computational cost. Firstly, it is impractical to assume the system knows the exact travel time series of every single road segment given the traffic dynamics.

Secondly, these algorithms do not help with switching congestion from one spot to another if all the drivers are provided the same time-dependent shortest path.

2.3 Dynamic Traffic Assignment Model

An alternative to this work could be the research done on dynamic traffic assignment (DTA) which leads to either system-optimal or user-optimal route assignments. DTA research can be classified into two categories: analytical methods and simulation-based models. Analytical models such as [24, 57, 58] formulate DTA as either nonlinear programming problems, optimal control problems, or variational inequalities. Although they provide theoretical insights, the computational intractability prevents their deployment in real systems [66].

Simulation-based approaches [16, 25, 52, 81] have gained greater acceptability in recent years, in which the time-dependent user equilibrium is computed by iterative simulations. The simulations are used to model the theoretical insights that cannot be derived from analytical approaches. This process computes the assignment of traffic flows until the travel times of all drivers are stationary. Unfortunately, there are still a number of issues associated with these approaches that make their deployment difficult: tractability for large scale road networks given the computational burden associated with the simulator, capability of providing real-time guidance, effectiveness in the presence of congestion, and behavior of drivers who ignore the guidance. For example, they assume the set of Origin-Destination (OD) pairs and the traffic rate between every OD pair are known. This information is highly dynamic especially in city scenarios, leading to frequent iterations of computationally expensive algorithms even when not needed from a driver benefit point of view. Additionally, the OD set is large, and the DTA algorithms may not be able to compute the equilibrium fast enough to inform the vehicles about their new routes in time to avoid congestions. The proposed system, on the other hand, is designed to be effective and fast, although

not optimal, in deciding which vehicles should be re-routed when signs of congestion occur as well as computing alternative routes for these vehicles.

The complexity of DTA systems has led scientists to look for inspiration in Biology and Internet protocols. In [76], Wedde et al. developed a road traffic routing protocol, BeeJamA, based on honey bee behavior. Similarly, Tatomir et al. [80] proposed a route guidance system based on trail-laying ability of ants. Inspired by the well-known Internet routing protocols, prothmann et al. [69] proposed decentralized Organic Traffic Control. However, since they employ ad hoc networking, these approaches have only a partial view of the traffic conditions, which may lead to less accurate re-routing. Also, simply treating vehicles as packets which always listen to the guidance ignores the nature of human behavior. Furthermore, these systems react to real-time data without insight into future conditions, thus introducing greater vulnerability to switching congestion from one spot to another.

There has been several other literatures that aim to provides near-optimal route to drivers but better scalability compared to DTA. The first related work is the Ph.D Thesis [49] from MIT. The basic idea is divided into two steps: Calculated the possible first k shortest paths from source to destination, and then determine which path each vehicle should take by minimizing a Lyapunov-style cost function. Meanwhile, the work [93] uses dynamic programming keep tracking traffic information in the network. Every time a car comes an intersection, it calculates the first-k shortest path candidates and proportionally chooses a candidate by using probability calculated from boltzmann distribution.

2.4 Vehicular Ad hoc Networks

There has been significant effort and progress on Vehicular Ad hoc Networks (VANETs) technology in recent years. VANETs are based on vehicle-to-vehicle and vehicle-to-infrastructure wireless communication. IEEE 802.11p is an approved amendment to

the IEEE 802.11 standard to add wireless access in vehicular environments (WAVE). It defines enhancements to 802.11 required to support Intelligent Transportation Systems (ITS) applications. Essentially, the VANETs research can be classified into three categories: routing, communication optimization, applications. These three categories tackle the VANETs key issues from the bottom to the top.

Obviously, routing is the fundamental point providing the basis for communication of the platform. For example, one of the major works is presented by [64] where RBVT-R and RBVT-P protocols are proposed. Both protocols leverage real-time vehicular traffic information to create road-based paths consisting of successions of road intersections that have, with high probability, network connectivity among them. Simulation results shows 40% increase in delivery ratio and 85% decrease in average delay.

The second category involves applying optimization techniques to minimize the communication overhead. The work in [79] described an effective safety alert broadcast algorithm for VANETs. Similarly, the technique in [50] presented a domain specific data aggregation scheme and a genetic algorithm to minimize the overall bandwidth requirements and placement of the roadside units in the initial deployment. Since VANETs are used for collaboration and information sharing, one of the key challenges is to avoid "broadcast storm". The work in [63] firstly analyzed this issue in MANETs and proposed a probabilistic, counter-based and distance-based schema to reduce redundant messages. As an extension, [89] investigated this problem in VANETs. Article [46] described how to utilize 2-hop neighborhood information more effectively to reduce redundant transmissions.

The traffic efficiency can be further improved by using network coding [22] and data compression. In [41], Katti et al. lay out a basic mechanism to efficiently forward packets by mixing packets from different sources using XOR. Meanwhile, Lee et al. [82] proposed a network coding-based file swarming protocol targeting VANETs

to allow shorter file downloading time. As for data compression, one example is MIT CarSpeak project [45], which proposed a loss-resilient compression octree structure to reduce the packet size and prevent packet loss.

Potential applications in VANETs are safety and non-safety related. Examples of safety related applications are in paper [13] which presents an overview of highway cooperative collision avoidance. The other services includes automatic road traffic alert dissemination, dynamic route planning, service for parking availability, audio and video file sharing between moving vehicles, and context-aware advertisement [62, 70, 72, 94].

The proposed hybrid system in this dissertation shares some similarities and differences with the above works. The similarity is that the hybrid design allows each vehicle to exchange its trip information across the VANETs. The difference is that a centralized server acts as a coordinator to collect the accurate traffic view based on the updates from each vehicle. Once congestion is detected, the latest traffic view is pushed to the selected vehicles and optimal individually tailored paths are distributively computed by each vehicle. The advantage is that real computation is off-loaded to VANETs to reduce computational cost which improves the scalability of the system. Furthermore, the hybrid system incorporates a privacy enhancing module, which dramatically reduces the location privacy leakage because each vehicle only submits location reports probabilistically in highly congested areas. In other words, the centralized server (through 3/4G network) and VANETs work together to fully utilize both internet access and VANETs ad-hoc communication.

2.5 Location Privacy Protection

There is always a trade-off between privacy and information sharing or disclosure. On the one side, the amount of the information gathered directly affects the effectiveness of the system. On the other hand, information disclosure violates the people's privacy (e.g., location, trajectory). The questions is how to measure the privacy and

minimize the privacy leakage. As for the measurement, the work in [60] analyzes the information leaks in the lookup mechanisms of structured peer-to-peer (P2P) anonymous communication systems and how these leaks can be used to compromise anonymity. Meanwhile, paper [14] defined Self Exposure Risk Index (SERI) and External Exposure Risk Index (XERI) to assess the privacy leakage.

As for location privacy, a large body of work focuses on spatial cloaking [29] to provide k-anonymity, which guarantees a user to be indistinguishable from at least k-1 others. The work in [26] argues that both spatial and temporal dimensions should be considered in the algorithm to achieve better k-anonymity, where a framework is designed to enables each mobile client to specify the minimum level of anonymity that it desires and the maximum temporal and spatial tolerances that it is willing to accept. Various techniques can be used to achieve advanced k-anonymity, for example, [92] computes location entropy while [55] uses the prefix of the location hash value. To prevent the location tracking, [77] achieves k-anonymity by injecting k-1 fake location traces. Fundamentally, k-anonymity reduces the quality of the user's localization, which is not applicable for continuous location based services such as real-time vehicle re-routing. The technique in [59] demonstrates path confusion approach which uses mobility prediction to create a web of intersecting paths, preventing un-trusted location based services from tracking users while providing highly accurate real-time location updates. An uncertainty-aware path cloaking algorithm is proposed in [36] for preserving privacy in GPS traces that can guarantee a level of privacy even for users driving in low-density areas. All the above mechanisms require a trusted centralized entity such as a proxy server for location reporting.

This dissertation argues that the proposed hybrid system greatly improves the driver's location privacy. The server only needs to acquire data from high density roads to produce a roughly accurate traffic view. The vehicles utilize VANETs to estimate road locate density and only upload to the central server probabilistically when

necessary. Afterwards, once again, through VANETs, the traffic view is propagate and optimal re-routing route computed distributively. On one hand, the probabilistic update scheme on high density road dramatically reduce the location leakage. On the other hand, risk of location tracking is distributed on the VANETs.

2.6 Chapter Summary

This chapter discussed the existing studies related to intelligent vehicular re-routing systems. The vehicular routing services are initially presented followed by fast k path generation and dynamic traffic assignment. Finally, VANET communication optimizations and location privacy systems were also reviewed.

CHAPTER 3

SYSTEM OVERVIEW AND CHALLENGES

This chapter firstly provides general overview of the basic centralized design of the proposed smart vehicle re-routing system, traffic data representation and estimation in Section 3.1.1, congestion prediction in Section 3.1.2, selection of vehicles to be re-routed in Section 3.1.3, ranking the selected vehicles in Section 5.2.7 and alternative route computation and assignment in Section 3.1.5. The potential challenges (e.g., privacy, scalability, robustness) are discussed in Section 3.2. Finally, the summary of the chapter is presented Section 3.3.

3.1 Basic Centralized Design

The objective of this dissertation is to implement and evaluate a real-time, cost-effective, and easily deployable vehicular traffic guidance system that reduces the effect of traffic congestions and lowers the trip times for all drivers. Implicitly, fuel consumption and pollution will be reduced as well. To achieve this goal, a system was developed consisting of smart phone-based vehicular networks and a back end server infrastructure for traffic monitoring and coordination. Smart phones were chosen as the vehicular platform because they are already carried by drivers in many vehicles, are powerful (have several communication interfaces, GPS, accelerometer, powerful CPU, plenty of storage, etc.), and are easily programmable. Once they become widespread, vehicular computing systems could be considered instead of smart phones. Figure 3.1 presents a comparison between these existing solutions and the system. This dissertation proposes to leverage the smart phone ubiquity to design and quickly deploy a cheap and effective traffic re-routing system. Such a system will benefit all of us through faster routes, less money spent on gas, and lower pollution.

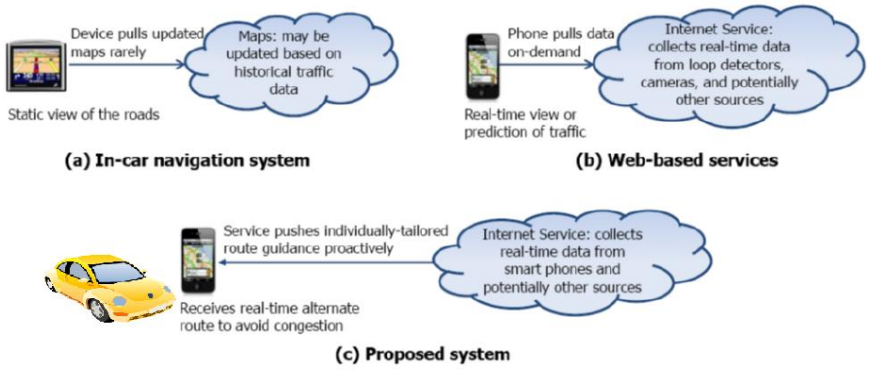


Figure 3.1 The system overview.

Particularly, the system model is composed of: (1) a centralized traffic monitoring and re-routing service (which can physically be distributed across several servers), and (2) a vehicle software stack for periodic traffic data reporting (position, speed, direction) and showing alternative routes to drivers. Vehicles run this software either on a smart phone or an embedded vehicular system. Vehicles are equipped with GPS receivers and can communicate with the service over the Internet when needed. When starting a trip, each vehicle informs the service of its current position and destination; the service sends back a route computed according to its strategy. It is assumed that the service knows the road network as well as the capacity and legal speed limits on all roads.

Logically, the traffic guidance system operates in four phases executed periodically: (1) data collection and representation; (2) traffic congestion prediction; (3) vehicle selection for re-routing; and (4) alternative route assignment for each such vehicle and pushing the guidance to the vehicles. Since data collection has been studied extensively in the literature, This issue is not addressed and it is assume that the centralized service receives traffic data from vehicles and road-side sensors where available. Each of the other phases are discussed in detail in this section and Chapter 4.

3.1.1 Traffic Data Representation and Estimation

The road network is represented as a directed, weighted graph, where nodes correspond to intersections, edges to road segments, and weights to estimated travel times. The weights are updated periodically as new traffic data becomes available. Several methods can be employed to estimate the travel time over a road segment. For instance, using vehicle probe data collected from on-board GPS devices to reconstruct the state of traffic is a well-studied topic [51, 90]. Greenshield's model [10] is used to estimate the travel time since it is used extensively in dynamic traffic assignment models by transportation researchers. The model considers that there is a linear relationship between the estimated road speed V_i and the traffic density K_i (vehicles per meter) on road segment i, as in Equation 3.1:

$$V_i = V_f(1 - \frac{K_i}{K_{jam}}) \qquad T_i = L_i/V_i \tag{3.1}$$

where K_{jam} and V_f are the traffic jam density and the free flow speed for road segment i, while T_i and L_i are the estimated travel time and length for the same segment. The free flow speed V_f is defined as the average speed at which a motorist would travel if there were no congestion or other adverse conditions. To simplify this implementation, it is considered that the free flow speed is the road speed limit. Basically, K_i/K_{jam} is the ratio between the $current\_number\_of\_vehicles$ and the N_{max}. N_{max} is the max number of vehicle allowed on the road. The $current\_number\_of\_vehicles$ is obtained from the traffic data collected by the service, whereas $N_{max} = length\_of\_road/(avg\_vehicle\_length + min\_gap)$.

3.1.2 Congestion Prediction

Periodically, the service checks the road network to detect signs of congestion. A road segment is considered to exhibit congestion signs when $K_i/K_{jam} > \delta$, where $\delta \in [0, 1]$ is a predefined threshold value. Choosing the right value for δ is particularly important

for the service performance. If it is too low, the service could trigger unnecessary re-routing; this may lead to an increase in the drivers' travel times. If it is too high, the re-routing process could be triggered too late and congestion will not be avoided. The evaluation in Chapter 5 confirms these hypotheses.

3.1.3 Selection of Vehicles to be Re-routed

When a certain road segment presents signs of congestion, the service looks for nearby vehicles to re-route. Specifically, vehicles are selected from incoming segments (i.e., segments which bring traffic into the congested one). To decide how far from congestion to look for candidates for re-routing, the service uses a parameter L (level). This parameter denotes the furthest distance (in number of segments) a candidate vehicle can be away from the congested segment. In practice, L could be computed as function of the severity of congestion; for example, the "level of service" (LOS) can be used to define in the Highway Capacity Manual [54]. L's value has to be large enough to mitigate congestion. If L is too high, however, more vehicles than necessary will be selected for re-routing, which can have undesired consequences (e.g., creating congestion in another spot). Since the focus of this dissertation is on the re-routing algorithms and the analysis of their performance, L was considered tuning parameter that is varied during these experiments.

The service performs a breadth first search (BFS) on the inverted network graph (i.e., the road network graph is directed), starting from the congested segments with maximum depth L and considers all these cars as candidates for re-routing. The process is illustrated in Figure 3.2. Assuming L=2 and signs of congestion are detected on the segment Sc, the system recursively selects the vehicles situated on the incoming segments of the congested segment in two steps. First, the vehicles located on segments S1 are included in the candidate set, followed by the vehicles situated on segments S2.

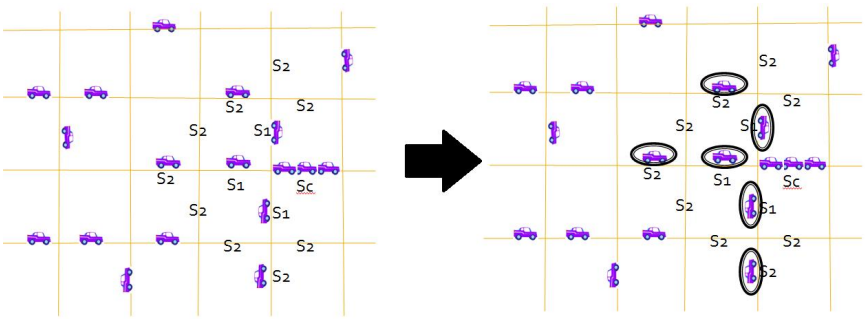


Figure 3.2 The vehicle selection process.

The union of all the vehicles affected by all the congestion segments was selected in whole network within the same level, then do vehicle ranking (Section 5.2.7), path computations and the assignments (Chapter 4).

3.1.4 Ranking the Selected Vehicles

The selected vehicles need to be ranked and assigned to alternative paths according to their rank for all strategies, except DSP. In this way, the performance of the strategies improve. The impact a congested road segment has on a vehicle's travel time is different depending on the remaining distance to the vehicle's destination. Intuitively, the drivers that are close to their arrival point may have a different perception of the congestion than the drivers that are far away from their destinations. This system uses an urgency function to rank the vehicles that are selected for re-routing. Hence, the vehicles with higher urgency are re-routed first and get relatively better routes.

Definition 1. *Given a set of vehicles* $V = (v_1, v_2, v_3, ..., v_m)$ *to be re-routed, two urgency functions were defined to compute the re-routing priority of a vehicle in* V*:*

- *Relative Congestion Impact:* RCI*=(RemTT-*$RFFTT$*)/*$RFFTT$
- *Absolute Congestion Impact:* ACI*=RemTT-*$RFFTT$

where RemTT is the remaining travel time, and $RFFTT$ *is the remaining free flow travel time for the vehicle.*

RCI measures the congestion impact on a vehicle relative to its remaining travel time, whereas ACI emphasizes the absolute increase in the travel time. Therefore, RCI gives a higher priority to vehicles that are close to their destinations, while ACI ranks first the vehicles that are further from their destinations. In Section 5.2, these two urgency functions are evaluated under different re-routing strategies.

3.1.5 Alternative Route Computation and Assignment

There are two main requirements for the re-routing algorithm: (1) compute an alternative routes for each driver that improve both single driver's trip time and global network efficacy (2) push the guidance to drivers fast to allow them enough time to switch on the new route. Essentially, a best effort algorithm is required, which finds good enough alternatives with real-time constraint.

Accordingly, two types of solutions were developed. The first solution is "Single Shortest Path Strategies", the second one is "Multiple Shortest Paths Strategies". The former computes one single path for each vehicle collaboratively according to the other vehicles's paths. The later computes k loopless shortest paths [48, 56, 67] according to current travel time and pick the optimal one for each vehicle. This is because one of the main goals of the system is to prevent moving congestion from one path to another. If all vehicles are re-routed on their current shortest paths, the algorithm might end-up doing exactly that. For this reason, vehicles are provided with alternative paths that lower their currently predicted travel time, but these paths do not have to be the shortest. The specification of the re-routing algorithms is presented in Chapter 4.

3.2 Challenges

3.2.1 Privacy

In order to accurately estimate the traffic conditions and provide individual re-routing guidance, the system must be aware with relatively high accuracy of the current position and destination/route of each vehicle. This requirement may lead to major privacy concerns and could preclude the adoption of the system. Therefore, it is necessary to explore ways to reduce drivers' privacy exposure risks.

The first solution is to send the traffic reports anonymously. In this way, drivers prevent others from linking their location to their identity. However, it is well known that identity can still be inferred from location traces by linking them together or with external sources [34]. Hence, a goal of the system is to balance between data collection accuracy and driver's privacy. Spatial and temporal cloaking are the main techniques for k-anonymity, which guarantees a user to be indistinguishable from at least k-1 others [26]. However, k-anonymity sacrifices the location accuracy, thus is not applicable for continuous location based services in this case. Naturally, minimizing the report frequency for each driver fits better since the location must be precise whereas a small sample is sufficient. Indeed, according to the Mobile Millennium Project [33], reports from 2-3% of drivers (or 2-3% of the time) are enough to obtain an accurate view of the traffic. Virtual Trip Lines have been proposed precisely in this context [35]. Besides, high density also provide natural obstacle for an un-trusted identity to identify or distinguish vehicles on the road. Accordingly, in Chapter 6, a distributed privacy by utilizing VANETs is proposed. Specifically, each vehicle on VANETs periodically detect road density and only send traffic report to the server based on certain probability when road density is high enough. This way, the server only receives limited amount of vehicle location data but still roughly accurate traffic view. Once congestion is detected, VANETs helps further propagate the traffic map

and re-routing path is computed locally on each vehicle. Chapter 7, this hybrid privacy protection module is evaluated through extensive simulations.

3.2.2 Robustness

This dissertation aims to create a robust system that works well in the presence of less than 100% penetration rate and acceptance rate. The main goal of the system is to provide benefits to drivers who have the system and accept the guidance even for low penetration and acceptance rates. A question that needs to be answered is: what are the minimum rates that allow the system to work properly? As by-product, because the system avoids congestions, the trip times are in fact lowered for all traffic participants. However, the individual benefits may vary significantly.

A salient feature of this system is its ability to adapt to drivers behavior. Smart phones can learn how drivers react to guidance and pass this knowledge to the re-routing algorithm. For example, the algorithm may give preference for re-routing to drivers who have a high acceptance rate.

3.2.3 Accurate Real-time Traffic View

The system has to adapt the number and frequency of reports submitted by smart phones to balance the need for an accurate global view of the traffic with the needs for privacy, scalability, and low communication overhead. These reports, in potential conjunction with historic data, are used to predict travel time and congestion on each road segment.

The average speed/travel time on a road segment can be determined accurately even if a low number of vehicles report. However, the question is: how to estimate the volume of traffic if only some vehicles report? A simple solution is to take advantage of one-hop ad hoc WiFi communications between phones (i.e., broadcast "hello" messages) to locally estimate the traffic density of the segment as function of

the number of messages heard per unit of time and transmission range. Then, the system may aggregate such reports from different vehicles on the same segment to achieve good volume estimations.

At this stage, only speed, volume, and road capacity are considered in predictions. As for future work, additional information such as road characteristics, traffic light delays, or weather conditions has to be incorporated in an effective deployable system.

3.2.4 Effective Real-time Guidance

There are two main requirements for the re-routing algorithm: (1) provide alternative routes with lower trip times (or do not interfere), and (2) push the guidance to drivers fast to allow them enough time to switch on the new route. Essentially, a best effort algorithm is required, which finds good enough alternatives in the allocated time constraint (i.e., time to reach the re-routing intersection).

This algorithm provides good alternative paths by considering all vehicles' route choices. In another words, more effective and efficient paths are computed for each vehicle based on collaborative knowledge to avoid the re-occurrence of another congestion. It provides better scalability than DTA since it avoid iterative and complex simulations during the computation process. Moreover, to meet real-time constraint, a distributed hybrid scheme is presented in Chapter 7 to offload time consuming path computation to VANETs. Both centralized and hybrid architecture is investigated through simulations extensively with varies parameters.

Another factor that impacts the driver benefits is the selection of vehicles for re-routing. The system needs a good utility function to select the best candidate vehicles. The system should re-route vehicles proportional to its distance to congestion. Moreover, it is necessary to evaluate the impact on the vehicle if it switches the pathes. Therefore, the system firstly picked the vehicles within certain segments from congested spot. Afterwards, a ranking function based on the remaining travel time is

applied. The more the remaining travel time increases, the more impact the congestion created for this vehicle, thus the more urgent this vehicle should be re-routed.

3.2.5 Implementation, Evaluation and Deployment

A prototype system can quickly be deployed by building a centralized service and offering free iPhone and Android applications. The system is designed to work even with few users, but the benefits become visible once enough users are in the system to achieve a good view of the real-time traffic. To help bootstrap the system, traffic data from infrastructure sources could be incorporated as well. As the system grows, there will be a need to add more servers; care must be taken in this case to avoid bottlenecks when the graph algorithms run over a potentially large distributed system. A seamless partitioning of the network such as in [87] would help in this situation.

For a successful system, it is imperative to have good guidance from the beginning to help with early adoption. Therefore, accurate simulations using real-life data must be performed to compare the performance of different re-routing algorithms under different sets of parameters. In this dissertation, the SUMO [11] simulator is used with a real city map in which the traffic is generated according to the traffic flow of the people living in the city.

A final question about deployment is who will provide and manage the servers for this system in centralized or hybrid architecture. This system is expected to be of interest to private companies, similar to the deployment of current navigation systems; these companies can either sell the phone applications or offer them for free and make money through location-based advertising.

3.2.6 Communication Overhead and Scalability

Thus far, a centralized version of the system has been presented. Naturally, one might ask: does this system scale for a large city network and a large number of

vehicles, especially when considering the real-time constraints for pushing the guidance to drivers? Many optimizations can be done in terms of data collection to reduce the amount of data and implicitly of communication. Another one is to maintain "normal traffic" parameters per segment at each phone and submit reports only when significant variation is observed. Depending on how much data is maintained on the phones, appropriate data management techniques and optimizations need to be considered as well.

Despite all these optimization, it may still be difficult to scale a centralized system. In consequence, decentralized and hybrid architectures are explored to improve system scalability User privacy can also be benefited from those architectures.

The Ad hoc architecture exchange information using multi-hop ad hoc communication among vehicles. It enables the best privacy protection and can quickly detect signs of congestion in small regions. Yet, its decision algorithms must be localized (e.g., consensus between vehicles in a certain region) which may result in sub-optimal re-routing decisions.

The Peer-to-peer design [71, 72] could achieve the same privacy benefits as the ad hoc architecture and, at the same time, acquire a global view of the traffic. Vehicles communicate with each other over the Internet and form peer-to-peer (P2P) networks for data collection and sharing. However, if designed naively, the system may not scale due to potentially large routing latencies in the P2P network.

Therefore, it is possible that a hybrid architecture, using both vehicle-to-vehicle and vehicle-to-server communication, has the potential to reach the best balance among the competing challenges mentioned before. There are several types of hybrid architectures that could be developed, but they share a common idea: split the computation between the server and the phones to improve system scalability and protect users' privacy.

Individual vehicle decisions: In this architecture, the centralized server collects anonymous traffic reports from vehicles, maintains the real-time traffic view (i.e., the graph), and distributes the graph to all vehicles or only to vehicles that demand it. Privacy is improved because the server does not need to know the identity and destination of vehicles. The vehicles have to make their own re-routing decisions under this architecture, similar to the ad hoc architecture. The advantage is that they have a global view of the traffic in this case. To avoid many vehicles going over the same shortest path, and thus provoking congestion, the application on the phones could use probabilistic methods to pick alternative routes (e.g., pick out of the k best routes with certain probabilities) to achieve a degree of load balancing.

Collaborative vehicle decisions: Compared with the previous one, this architecture adds ad hoc communication between vehicles. Collaborative ad hoc algorithms are used to coordinate the choice of alternative routes among vehicles located in the same region. Thus, more effective re-routing is achieved at a cost of a slightly more complex system design.

In this dissertation, the collaborative vehicle decisions are made through knowledge sharing by utilizing VANETs. Each vehicle equipped with 3/4G Internet connectivity submits traffic updates to the central server. Once congestion is detected, the traffic map is pushed to the selected vehicles. Each vehicle starts to disseminate their trip data through VANETs. Once enough knowledge is collected, each vehicle runs the re-routing algorithm locally. To further prevent location privacy leakage, each vehicle only needs to report traffic data in high density roads based on a probabilistic formula. Chapter 6 will describe the details of this hybrid system architecture.

3.3 Chapter Summary

The chapter has outlined the basic centralized framework for a dynamic re-routing system, discussed several alternative system architectures (centralized, decentralized,

and hybrid), and identified system-centric (e.g., scalability, effective real-time guidance) and human-centric (e.g., privacy, robustness to driver choices) challenges that have to be overcome to make this vision reality.

CHAPTER 4

CENTRALIZED RE-ROUTING STRATEGIES

Recent research has proved that real-time traffic flow data and road travel time can be determined based on data reported by vehicles or road-side sensors [38, 61, 90]. The question is how to utilize this knowledge in an intelligent fashion to avoid congestion and reduce the drivers' travel times. This chapter presents five re-routing strategies, categorized in two classes. The first class presented in Section 4.1 includes two re-routing strategies that compute a single, alternative new path for each re-routed vehicle. The strategies are based on the well-known Dijkstra algorithm and on the A^* algorithm with a modified heuristic, respectively. Section 4.2 presents the second class consisting of three re-routing strategies that compute multiple, alternative new paths for each of the re-routed vehicles. Then, different heuristics are used to choose the best alternative path to be assigned to a vehicle. Based on the five proposed strategies, this thesis describe the main re-routing process executed by the traffic guidance system in Section 4.3. Finally, Section 4.4 presents a Dynamic Traffic Assignment strategy [25] that is used as a baseline to measure the effectiveness and efficiency of the proposed strategies. The chapter concludes in Section 4.5.

4.1 Single Shortest Path Strategies

4.1.1 Dynamic Shortest Path (DSP)

DSP is a classical re-routing strategy that assigns the selected vehicles to the path with lowest travel time. However, different from the existing systems, the system described in this dissertation system takes a proactive approach. Specifically, each time a road segment presents signs of congestion, the service obtains the set of cars whose paths intersect this road segment and computes for each car a new shortest path based on the current travel time in the road network. Therefore, the path of

each car can be periodically updated on an event-driven basis. The advantage of this strategy lays in its simplicity and consequently reasonable computational cost, i.e., $O(E + Vlog(V))$ [23], where E is the number of road segments and V is the number of intersections of the road network. This strategy is expected to provide good results when the number of re-routed vehicles is low, since in this case the risk of switching congestion from one spot to another is low. Hence, locally redirecting the traffic when congestion happens should be sufficient in this case. On the other hand, when the traffic density is higher, there is an increased risk of switching the congestion from one road to another. Moreover, the re-routing frequency for a driver is likely to increase in this case, which can be annoying to drivers.

4.1.2 A* Shortest Path with Repulsion (AR*)

The DSP strategy only takes into account the current view of the traffic when performing re-routing, without considering the impact the re-routing will have on the future traffic. To address this limitation, AR* is proposed, which modifies the A* search algorithm to include the prior re-routing decisions into the computation of the current shortest path. A* [32] uses a best-first search and a heuristic function to determine in which order to visit the network nodes (road intersections in this case). Given a node x, a heuristic function $F(x)$ is computed as the sum $G(x) + H(x)$. $G(x)$ is the path-cost from the start node to x, which corresponds to the travel time in this case, while $H(x)$ is a heuristic estimation of the remaining travel time from x to the destination node. In addition, $H(x)$ has to be "admissible" (i.e., it must not overestimate the remaining travel time to the destination) to produce the shortest path between the source and the destination. Therefore, $H(x)$ is computed as the Euclidean distance divided by the maximum speed in the road network.

Algorithm 1 A Star Shortest Path with Repulsion Re-routing

```
1: procedure AstarRepulsion(start,end)
2: P[start]=empty {the reverse pointer of the path, which is used to re-construct the path}
3: closedset = set() {The set of nodes already evaluated}
4: openset = set()
5: openset.add(start) {The set of tentative nodes to be evaluated, initially containing the
   start node}
6: Gscore[start] = 0.0 {Travel time cost from start along best known path}
7: Hscore[start] = Euclidean(start, end)/maxspeed
8: Rscore[start] = 0.0
9: Fscore[start] = 1.0
10: while openset is not empty do
11:    sumF=SumFscore(openset)
12:    sumR=SumRscore(openset)
13:    for all node in openset do
14:       Fscore[node]=(1-β)*Fscore[node]/SumF + β*Rscore[node]/SumR
15:    end for
16:    current=getleastFscore(openset)
17:    if current==end then
18:       return (Fscore[current],P)
19:    end if
20:    openset.remove(current)
21:    closedset.add(current) {add current to closedset}
22:    for all edge in current.outEdges do
23:       node=edge.endnode
24:       if node in closedset then
25:          continue
26:       end if
27:       tentative_g_score=Gscore[current] + edge.actualtime
28:       tentative_r_score=Rscore[current] + edge.weight_footprints
29:       if node not in openset then
30:          openset.add(node)
31:          Hscore[node]=Euclidean(node, end)/maxspeed
32:          tentative_is_better=True
33:       else
34:          if tentative_g_score<Gscore[node] then
35:             tentative_is_better=True
36:          else
37:             tentative_is_better=False
38:          end if
39:       end if
40:       if tentative_is_better == True then
41:          P[node]=edge
42:          Gscore[node]=tentative_g_score
43:          Rscore[node]=tentative_r_score
44:          Fscore[node]=Gscore[node]+Hscore[node]
45:       end if
46:    end for
47: end while
48: return (0.0,{})
49: end procedure
```

Future congestion occurs if many drivers take the same road segment within the same future time window. [1] As this model assumes that the drivers share their route information with the service, it is possible to estimate the future footprint of each driver in the road network.

Definition 2. *A weighted footprint counter, fc_i, of a road segment i is defined as follows: $fc_i = n_i \times \omega_i$, where n_i is the total number of vehicles that are assigned to paths that include segment i, and ω_i is a weight associated with i. $\omega_i = \frac{len_{avg}}{len_i \times lane_i} \times \frac{Vf_{avg}}{Vf_i}$, where len_{avg} is the average road segment length in the network, Vf_{avg} is the average free flow speed of the network, len_i is the length of i, Vf_i is the free flow speed of i, and $lane_i$ is number of lanes of i.*

In the formula, n_i represents the discretized future traffic flow on road i. The weights were incorporated into the formula in order to count for different road characteristics. For example, suppose there are two road segment r_i, r_j. Although n_i=n_j, the segments should not be treated equally since r_i has higher capacity (more lanes or longer length), thus the possibility of causing congestion is lower. In other words, the impact of the traffic flow n_i on road r_i is lower than n_j on r_j even though n_i=n_j.

In AR^*, the heuristic function $F(x)$ is modified to include the other vehicles sharing the same path as a repulsive force. Specifically, the repulsive score $R(x)$ of a node x is defined as the sum of the weighted footprint counters (cf. Definition 2) from the starting node to the node x. Thus, the path-cost function becomes $F(x) = (1-\beta) \times (G(x) + H(x)) + \beta \times R(x)$, where $G(x)$ and $H(x)$ are computed same as in the original algorithm. $G(x) + H(x)$ measures the travel time factor, while $R(x)$ reflects the impact of other vehicle traces on the examined path. Since the travel time and the repulsive force use different metrics, the values are normalized their values are computed. $F(x)$ as a linear combination of the two factors. Parameter β presents

[1] The time window size equals the period used by the system to evaluate congestion.

the weight of the repulsion. If β is too high, the resulting path will be diverted too far away from the optimal. Similarly, if β is too low, it will be the same as the naive shortest path(DSP) strategy. Therefore, the beta value was varied from 0.01 towards 0.5 and it was observed that the interval [0.05,0.2] results in better performance in Section 5.2.8.

The complete algorithm is presented as pseudo code in Algorithm 1. Starting from the initial node, the algorithm maintains a queue of nodes to be traversed, denoted as the open set (lines 3-5). At each iteration, the node with the lowest Fscore value is removed from the queue (lines 16-19), the values of its neighbors are updated accordingly (lines 27-28), and these neighbors are added to the queue (line 30). The algorithm continues until the end node has been reached or until the queue is empty. The normalization of the travel time and the repulsive force factors is done at line 14. A path is returned at line 18 if found, otherwise an empty path is returned in line 48.

Figure 4.1 illustrates a simple example of how AR^* is used in re-routing. Given the assumption that vehicles v_1, v_2, v_3 having the same origin and destination, i.e., from ab to ij, need to be re-routed and that $urgency(v_1) > urgency(v_2) > urgency(v_3)$. At the beginning, since no vehicle has been assigned any path, AR^* performs normal A^* search and assigns the shortest path ab, bc, cd, di, ij to vehicle v_1. When computing the shortest path for vehicle v_2, AR^* will find ab, bg, gh, hi, ij. Although v_2 has the same destination as v_1, the path found by AR^* is different since it considers the footprints produced by $v1$ as a repulsion. Hence, AR^* avoids the already assigned paths as much as possible, while still keeping the new path as short as possible. Finally, the procedure is repeated for vehicle v_3 and the path ab, bc, cd, di, ij is obtained for the same reasons.

Notice that AR^* has to be employed by the re-routing system in an iterative manner. Namely, after the selected vehicles to be re-routed have been ranked based on their urgency, the system calculates sequentially each vehicle's route starting from

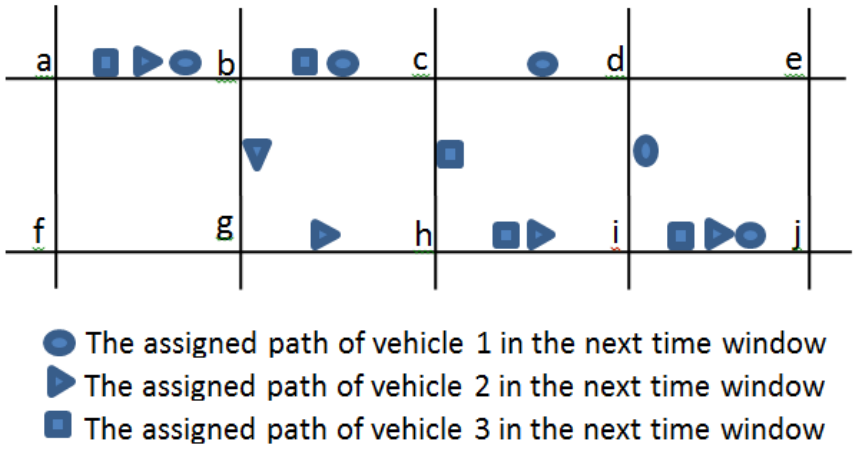


Figure 4.1 AR^* re-routing example. All road segments have same weight and $\beta = 0.5$.

the most urgent one. Therefore, in the case of AR^*, the computation time increases linearly with the number of re-routed vehicles. On the other hand, as explained in the next sections, the rest of the proposed re-routing methods optimize this phase by grouping the vehicles to be re-routed based on their origin-destination, which leads to lower computational complexity.

4.2 Multiple Shortest Paths Strategies

The two strategies proposed above compute a single path for each re-routed vehicle. However, the two methods have opposite behaviors. Firstly, DSP sacrifices effectiveness (since it does not consider the impact of re-routing on the future traffic) to optimize the computational cost (by grouping the re-routed vehicles on their origin-destination). Secondly, AR* trades efficiency (since it computes an alternative path for each vehicle) for effectiveness (by taking into account the future traffic configuration). In this section, a new class of re-routing strategies designed to obtain the best trade-off between efficiency and effectiveness are introduced. For these strategies, the re-routing process is divided into two steps. First, k-shortest paths are computed for each selected vehicle based on the travel time in the road network, where k is a predefined parameter. Compared to DSP this approach involves a higher computation time, but it still

permits to group vehicles on their origin-destination. Therefore, the computation time is expected to be lower than AR*. Second, the vehicles are assigned to one of their k-shortest paths in the order of their ranking. Among the k-shortest paths, the algorithm selects the path the least employed by other vehicle traces. Hence, this strategy is expected to have an effectiveness similar to AR*. Three heuristics for the selection of the best path among the k-shortest paths are proposed.

4.2.1 Random k Shortest Paths (RkSP)

RkSP assigns each selected vehicle to one of the k paths randomly. The goal is to avoid switching congestion from one spot to another by balancing the re-routed traffic among several paths. Compared to DSP, the price to pay is a higher computational complexity, $O(kV(E + Vlog(V)))$ [48], which increases linearly with k. Although a larger k will allow better traffic balancing, it also increases the difference in the travel time among the k paths. Therefore, to prevent an excessive increase of the travel time for some drivers, RkSP limits the maximum allowed relative difference between the fastest and the slowest path to 20%.

4.2.2 Entropy Balanced k Shortest Paths (EBkSP)

While RkSP addresses the main potential shortcoming of DSP (i.e., moving congestion to another spot), it has its own deficiencies. First, it increases the computational time, which matters because the alternative paths must be computed and pushed to vehicles before they pass the re-routing intersection. Second, it assigns paths randomly to vehicles, which is far from optimal both from a driver point of view and from the global traffic point of view. To address this second shortcoming of RkSP, the EBkSP strategy is proposed. The idea is to perform a more intelligent path selection by considering the impact that each selection has on the future density of the affected road segments. The more intelligent path selection comes at the cost of a slightly increased complexity.

However, this optimization is expected to improve the traffic from a global point of view. In addition, as in AR*, EBkSP ranks the cars to be re-routed based on an urgency function that quantifies the degree to which the congested road affects the driver travel time. Thus, the more affected vehicles will have priority and be re-routed first.

The entropy idea comes from Shannon information theory [78]. Several works [18, 92] have successfully applied it to compute the popularity of a visited area among all the users. To avoid creating new congestions through re-routing, a "popularity" measure is associated with road segments in EBkSP. Entropy is used to define the popularity of a path as follows.

Definition 3. *Let* $(p_1, ..., p_k)$ *be the set of paths computed for the vehicle which will be assigned next. Let* $(r_1, ..., r_n)$ *be the union of all segments of* $(p_1, ..., p_k)$*, and let* $(fc_1, ..., fc_n)$ *be the set of weighted footprint counters associated with these segments. The popularity of* p_j *is defined as* $Pop(p_j) = e^{E(p_j)}$*.* $E(p_j)$ *is the weighted entropy of* p_j *and is computed as* $E(p_j) = -\sum_{i=1}^{n} \frac{fc_i}{N} \ln \frac{fc_i}{N}$*,* $N = \sum_{i=1}^{n} n_i$*.*

The value of $E(p_j)$ measures the probability that a number of vehicles will be on the path p_j in a time window. According to the above definition, the value of $Pop(p_j)$ is $0 \leq Pop(p_j) \leq m$, where m is the number of vehicles. $Pop(p_j)$ has the maximum value m when every previously assigned vehicle traverses entirely p_j (i.e., they take the same path). $Pop(p_j)$ has the minimum value when no one takes the path p_j. Intuitively: the higher the popularity of a path, the higher the probability that more drivers will take this path.

After vehicle selection and ranking, the central server assigns each vehicle to the least popular path among its k-shortest paths in order to avoid potential future congestions. Specifically, the first vehicle is assigned the current best path without considering others. Then, the road network footprints are updated based on the new path. When assigning the second vehicle, the popularity score of its k-shortest paths

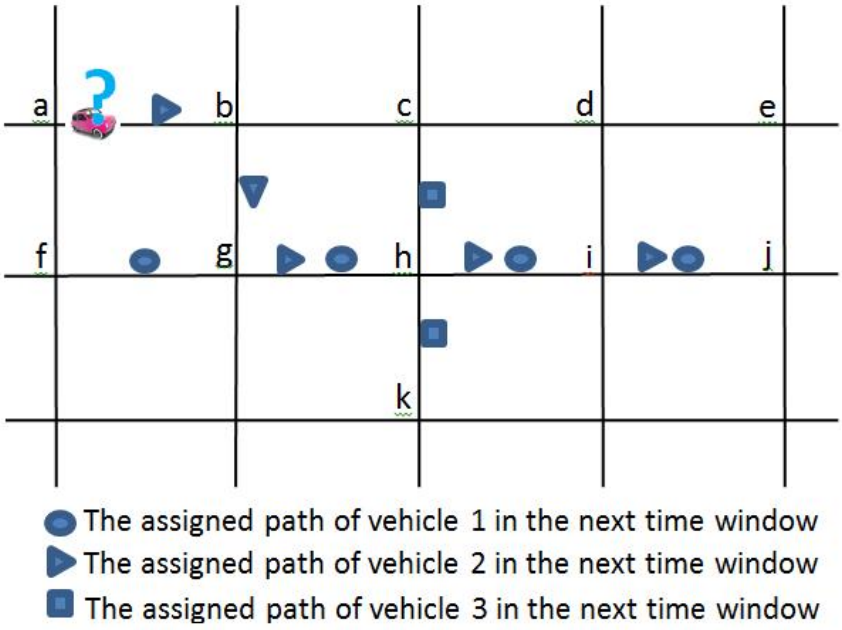


Figure 4.2 A EBkSP re-routing example. All segments have same weight.

are calculated and the least popular path will be chosen. The process is then repeated for the rest of the re-routed vehicles.

Figure 4.2 illustrates an example of EBkSP re-routing. It is assumed that vehicles (v_1, v_2, v_3) have been assigned to their paths before v_4, and each road has the same weight (i.e., $\omega_i = 1$). The footprints of (v_1, v_2, v_3) in the next time window are (fg, gh, hi, ij), (ab, bg, gh, hi, ij), and (ch, hk), respectively. For v_4, which travels from ab to ij, there are three alternative paths with similar travel times: $p_1(ab, bg, gh, hi, ij)$, $p_2(ab, bc, ch, hi, ij)$, and $p_3(ab, bc, cd, di, ij)$. The union of their segments is the set $(ab, bg, gh, hi, ij, bc, ch, cd, di)$, and their weighted footprint counters are (1,1,2,2,2,0,1,0,0). Consequently, N=11, $E_V(p_1)$=2.29, $E_V(p_2)$=1.67 and $E_V(p_3)$=0.53. Hence, v_4 will be assigned to p_3 because it is the least popular.

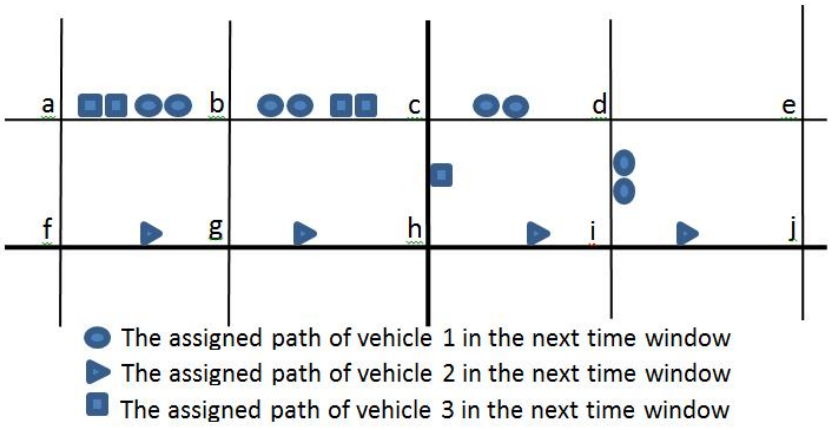
Algorithm 2 Flow Balanced k Shortest Path Re-routing

```
1: procedure LocalOptAssign(allkPaths, sortedVehicles)
   {generate initial solution}
2: for all vehicle in sortedVehicles do
3:    {origin, dest}=getVehicleOD(vehicle)
4:    newpath = pickPath_leastfootprints(allkPaths, origin, dest)
5:    reduction=getReduction()
6:    vehicle.selectedpath=newpath
7:    updateFootprint(vehicle)
8: end for
   {locally optimize the initial solution}
9: iter=0
10: repeat
11:    for all vehicle in sortedVehicles do
12:       {origin, dest}=getVehicleOD(vehicle)
13:       newpath=pickpath_random(allkpath,orgin,dest)
14:       newreduction=getReduction(newpath,vehicle.selectedpath)
15:       if newReduction<reduction then
16:          vehicle.selectedpath=newpath
17:          updateFootprint(vehicle)
18:          reduction=newReduction
19:       end if
20:    end for
21:    iter=iter+1
22: until iter>MaxIteration{MaxIteration is a constant, set as 10 here.}
23: end procedure
```

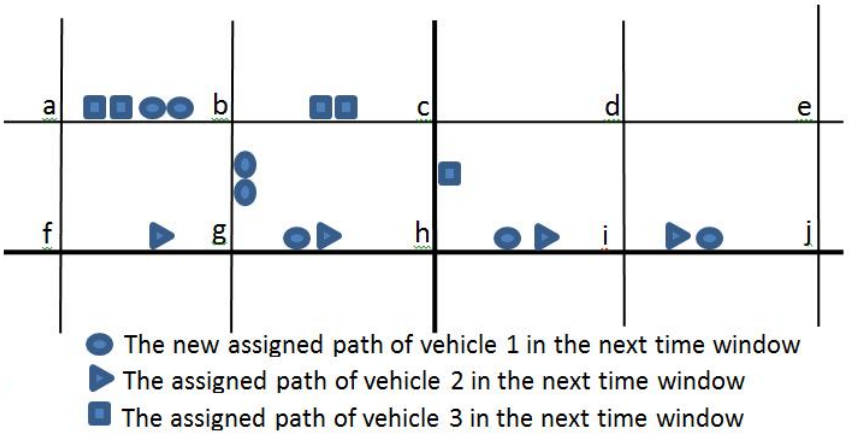
4.2.3 Flow Balanced k Shortest Paths (FBkSP)

RkSP and EBkSP distribute the traffic load of the re-routed vehicles by randomly choosing between alternative paths or by balancing the system entropy among multiple paths. Since the key idea is load balancing, an alternative approach designed to directly balance the traffic load, i.e., the weighted footprint counters, through local search optimization [37]. The goal of the local "search" is to find the path assignment in which the sum of the weighted footprint counters is minimal, i.e., to minimize $\sum_{s_i \epsilon S} fc_{s_i}$ in a network region, where S is the set of all region segments. As described in Definition 2, weighted footprint counter fc_i indicates the impact of the traffic flow on road segment r_i (i.e., the possibility of generating future congestion on r_i). Therefore, the summation of the weighted footprints counters of all the road segments measures the risk of congestion of the whole network. In another words, as a weighted footprint

counter indicates the future flow magnitude, minimizing the sum of the weighted footprint counters means having balanced flows on all paths, and thus, reducing the risk of producing congestion.



(a) The old assignment



(b) The new assignment

Figure 4.3 A FBkSP example.
$\omega_{fg} = \omega_{gh} = \omega_{hi} = \omega_{ij} = \omega_{ch} = 1$,
$\omega_{ab} = \omega_{bc} = \omega_{cd} = \omega_{de} = \omega_{af} = \omega_{bg} = \omega_{di} = \omega_{ej} = 2$.

Figure 4.3 illustrates how the path assignment affects the total number of weighted footprint counters. Assume that initially the vehicles (v_1, v_2, v_3) are assigned to the paths (ab, bc, cd, di, ij), (fg, gh, hi, ij) and (ab, bc, ch), respectively, and that the road segments have different weights (cf. Figure 4.3). Then, the sum of the weighted footprint counters in this network region is 18 (cf. Figure 4.3(a)). However, if v_1

switches to the path (ab, bg, gh, hi, ij), the sum of the weighted footprint counters is reduced to 16 as shown in Figure 4.3(b). Therefore, the system will select the latter assignment.

To implement the optimization of the total number of footprints in a road network region, a random search strategy (cf. Algorithm 2) is implemented. The system generates first a good path assignment solution for all selected vehicles by assigning to each vehicle the path with the current least number of footprints (lines 2-8 in Algorithm 2). This initial assignment does not necessarily guarantee the minimum sum of footprint counters of the considered network region, i.e., the union of all segments of the k shortest paths of the re-routed vehicle. Therefore, the system randomly modifies the initial assignment in order to improve it (lines 14-16). If the new assignment reduces the total number of weighted footprint counters in the network region, the new assignment is accepted (lines 18-19). Otherwise, the assignment is rejected. This process runs iteratively until the limit number of iterations is attained (line 26).

Disjointness of the k paths. The k-shortest paths (kSP) algorithm used in this dissertation computes a set of k shortest-time paths that are loopless but potentially overlapping. Using k disjoint shortest paths (or paths with a low degree of similarity) does not necessarily improve the re-routing performance of these algorithms. In order to compute k disjoint shortest paths, a typical algorithm computes first m shortest paths (m>k), and then selects the k disjoint paths from the set of m paths. Once the computation cost for determining the m paths is paid, EBkSP and FBkSP will perform better over m paths than over a subset of k paths because the total number of road segments that can be used for load balancing is larger.For example, vehicle A needs to be rerouted from origin O to destination D. In order to calculate 3 disjoint paths, paths set M are calculated, M={[O,A,B,C,E,F,G,H,D], [O,A,B,C,E1,F1,G,H,D], [O,A,B,C,E2,F1,G1,H,D], [O,A1,B1,C,E,F,G,H,D], [O,A1,B1,C,E1,F1,G,H,D]}. By

removing the similar path, the 3-disjoint paths computed are set N. In this case, N={[O,A,B,C,E,F,G,H,D], [O,A,B,C,E2,F1,G1,H,D], [O,A1,B1,C,E1,F1,G,H,D]}. The effectiveness of load balancing globally across all vehicles on path set N must be less than or equal to the effectiveness on set M since $N \subseteq M$. Besides, computing disjoint paths set N requires the same computation power as path set M, therefore, we can see, in the sense, that pruning similar path can only reduce load balancing performance and increase computation time. The experimental results presented in Section 5.2.6 confirm that increasing the k value improves significantly the effectiveness of the re-routing.

4.3 Re-routing Process

In this section, the global re-routing process is presented, which was the basis for the traffic guidance system described in this thesis. The process is presented in Algorithm 3. The system periodically looks for signs of congestion in the road network (line 4). If signs of congestion are detected, then the system selects the vehicles situated near to the congested road segments (cf. Section 3.1.3) and ranks them based on the urgency function (cf. Section 5.2.7). Finally, alternative routes are computed for the selected vehicles by using one of the five proposed re-routing strategies. It is worth noticing that except AR*, all the other re-routing strategies optimize the alternative path search by grouping the vehicles on their origin-destination (line 10). This can lead to a significant reduction of the computational cost as showed in Section 5.2.

Algorithm 3 The Main Process

```
1: procedure main
2: while true do
3:    updateEdgeWeights()
4:    congestedRoads=detectCongestion(edgeWeights)
5:    if #congestedRoads>0 then
6:       for all road in congestionRoads do
7:          selectedVehicles=selectedVehicles ⋃ selectVehicles(road)
8:       end for
9:       sortedVehicles=sortByUrgency(selectedVehicles)
10:      allpaths=Emtpy
11:      if not AR* then
12:         odPairs=updateODPairs(selectedVehicles)
13:         if DSP then
14:            allPaths=Dijkstra(odPairs)
15:         else
16:            allPaths=compuate_all_kShortestPaths(odPairs)
17:         end if
18:         doReroute(allPaths, sortedVehicles)
19:      else
20:         for all vehicle in sortedVehicles do
21:            {origin, dest}=getVehicleOD(vehicle)
22:            newPath=AstarRepulsion(origin,dest)
23:            if newPath is not empty then
24:               setRoute(vehicle, newPath)
25:            end if
26:         end for
27:      end if
28:   end if
29:   wait(period) {The process executes periodically.}
30: end while
31: end procedure

32: procedure doReroute(allPaths, sortedVehicles)
33: if FBkSP then
34:    LocalOptAssign(allPaths, sortedVehicles)
35: else
36:    for all vehicle in sortedVehicles do
37:       {origin, dest}=getVehicleOD(vehicle)
38:       if DSP then
39:          newPath = allPaths[origin][dest][0]
40:       end if
41:       if RkSP then
42:          newPath = pickPath_random(allPaths[origin][dest])
43:       end if
44:       if EBkSP then
45:          newPath = pickPath_leastPopular(allPaths[origin][dest])
46:          updateFootprint(vehicle, newPath)
47:       end if
48:       setRoute(vehicle, newPath)
49:    end for
50: end if
51: end procedure
```

4.4 Dynamic Traffic Assignment

The work on DTA algorithms is essential for the problem considered in this dissertation, i.e., improving the individual travel time through traffic re-routing and guidance. Nevertheless, as explained in Chapter 2, DTA is not yet the most viable solution for real-time traffic guidance, mainly because of the DTA's very high computational complexity coupled with the high dynamics of the traffic and the imperfections in traffic knowledge. In spite of this, DTA can offer valuable information as, for example, the level of improvement in the travel time that can be achieved in an ideal situation (i.e., where computational cost is not an issue and the traffic information is perfect). Therefore, DTA is employed to obtain a lower bound on the optimization of the travel time for comparison with the results produced by the proposed strategies.

The DTA model used in this dissertation tries to achieve stochastic user equilibrium (SUE) through an iterative simulation process and mathematical modeling (see Chapter 2). Given the traffic demand, it chooses some initial routes assuming zero traffic. Then, it calculates the network load and the travel times by simulation and updates the route choices of the drivers. This process is repeated until the travel times are stationary or a maximum number of iterations is reached. The simulation-based DTA tool employed in this dissertation was proposed in [5, 25]. At least three parameters have to be given as input: a road network, a set of trips, and the maximum number of iterations. The higher the number of iterations is, the higher is the probability to achieve a SUE traffic state. In these experiments, the maximum number of iterations is defined to 50, since that was the value specified in [12]. The DTA algorithm, as defined in [25], is summarized next:

Step 1: Initialize the route of each driver by the optimal route in the empty network.

Step 2: Calculate the time dependent costs of the road segments by simulation.

Step 3: Recalculate the optimal routes of a certain portion p of the drivers using the time dependent costs from step 2.

Step 4: If routes have changed in step 3, go to step 2.

Note that the DTA algorithm involves not only shortest path graph computations but also simulations. The purpose of the simulation is to help DTA acquire a relative accurate estimation of the travel times given the assignment of the previous iteration. Then, the estimated travel times are used to adjust the assignment in the next iteration.

However, this inevitably leads to increased computational burden. In comparison, the approach in this dissertation approach proposes alternative routes to drivers during their entire journey based on the dynamic conditions in the road network, and most of the computation is spent on shortest path graph algorithms. Therefore, this approach is expected to be more efficient than DTA.

4.5 Chapter Summary

This chapter presented two classes of vehicle routing algorithms, namely, Single Shortest Path strategy (SSPS) and Multiple Shortest Paths Strategies (MSPS). These algorithms leveraged real-time vehicular traffic information to compute individually tailored optimal paths with high probability of lowering travel time and avoiding congestion in the future. The SSPS strategy focused on integrating other vehicle's paths to compute the current vehicle's path. Meanwhile, MSPS emphasized the load balancing efficiency to avoid potential congestion in the future. This chapter also reviewed a state of art DTA model that will be compared with the proposed algorithms.

CHAPTER 5

EVALUATION OF CENTRALIZED SYSTEM

This chapter presents the evaluation through simulations of the five vehicle re-routing strategies presented in Chapter 4 in the basic centralized design. The main objective of this simulation-based evaluation is to study the performance of the five re-routing strategies under various scenarios. Specifically, to address the following questions:

- Which strategy leads to the most benefits for drivers in terms of travel time and number of re-routings?
- What is the trade-off between strategy effectiveness and their efficiency in terms of computation time? How do the proposed strategies compare to a DTA-based approach in terms of effectiveness and efficiency?
- Which strategies scale better with the number of cars?
- How do parameters (number of alternative paths, car selection level, etc.) influence the performance?
- How robust is the system under various compliance rates (i.e., percentage of drivers who follow the guidance) and penetration rates (i.e., percentage of vehicles which have this software)?

The experimental settings are firstly introduced in Section 5.1. Afterwards, the results are presented and analyzed in Section 5.2. The chapter concludes in Section 5.3.

5.1 Experimental Settings

Both SUMO 15.0 [11] and TraCI [86] were employed for these simulations. SUMO is an open source, highly portable, microscopic road traffic simulation package designed to handle large road networks. TraCI is a library providing extensive commands to control the behavior of the simulation including vehicle state, road configuration, and traffic lights. The re-routing strategies algorithms were implemented using TraCI. Essentially, when SUMO is called with the option to use TraCI, SUMO starts up,

Table 5.1 Statistics of the Two Road Networks

	Brooklyn	**Newark**
Network area	$75.85km^2$	$24.82km^2$
Total number of road segments	551	578
Total length of road segments	$155.55km$	$111.41km$
Total number of intersections	192	195

loads the scenario, and then waits for a command. Thus, variables in the simulation can be changed (e.g., new paths assigned to certain vehicles). Then, a new command can be sent with how many seconds to run the simulation before stopping and waiting for another command.

Two urban road maps were downloaded from OpenStreetMap [31] in osm format. One is a section of Brooklyn, NY and the other is in Newark, NJ. The Netconvert tool in SUMO was used to convert the maps into a SUMO usable format, and the Trafficmodeler tool [65] to generate vehicle trips. Netconvert removes the pedestrian, railroad, and bus routes, and sets up a static traffic light at each intersection to make the simulations more realistic (as the maps do not have STOP signs). All roads have the same speed limit (13.9 m/s); some roads have one lane in each direction, while others have just one lane based on the specification in the OpenStreetMap osm file. The statistics of the two networks are shown in Table 5.1. By default, the shortest travel time paths are automatically calculated and assigned to each vehicle at the beginning of simulation based on the speed limit. Figure 5.1 illustrates the simulation process. Figures 5.2 (a) (b) show the traffic flow in both networks. Trafficmodeler was used to generate a total of 1000 cars in the Brooklyn network from the left area to the right area in an interval of 1000 seconds. The origins and the destinations are randomly picked from the left area and the right area, respectively. In the Newark network, 908 cars were generated having the origins picked randomly from the peripheral road segments and the destinations on the road segments inside the hot spot circle.

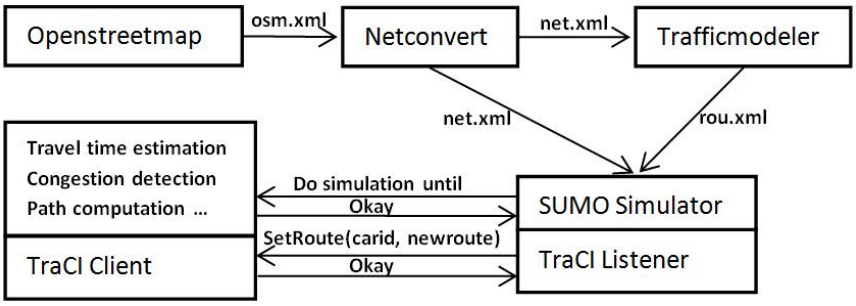


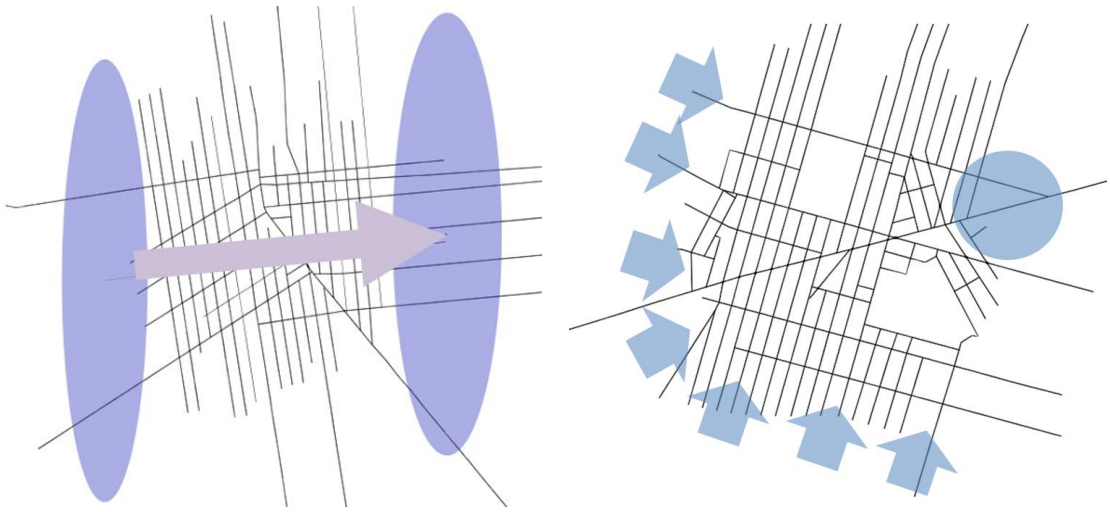
Figure 5.1 The simulation process.

In the simulations, the default settings in SUMO 15.0 were used for vehicle length=5m, the minimal gap=2.5m, the car following model (Krauss [44]), and the driver's imperfection=0.5. For each scenario, the results are averaged over 20 runs. Initially, an ideal scenario is assumed, in which all drivers have the system and accept the route guidance. These assumptions are relaxed in the last part of the evaluation. Table 5.2 defines the parameters used in this evaluation.

A DTA-based re-routing strategy (cf. Section 4.4) was also implemented by using a DTA tool provided with the SUMO generator. Contrary to the proposed approach, implemented in this dissertation, the DTA strategy computes the routes leading to user equilibrium for all the vehicles in one shot, before any vehicle starts its journey. Thus, the DTA tool produces a file containing the routes of all the simulated vehicles, which is supplied to the SUMO simulator. Based on this route file, SUMO

Table 5.2 Parameters in Centralized Re-routing Algorithms

period	The frequency of triggering the re-routing; by default period=450s
threshold δ	Congestion threshold; if $K_i/K_{jam} > \delta$, the road segment is considered congested; by default $\delta = 0.7$
urgency	Urgency policy: RCI or ACI
level L	Network depth to select vehicles for re-routing starting from the congested segment and using BFS on the inverted network graph
# paths k	The max number of alternative paths for each vehicle; by default $k = 4$
repulsion weight β	The weight of repulsion in AR^*; by default $\beta = 0.05$



(a) Brooklyn (b) Newark

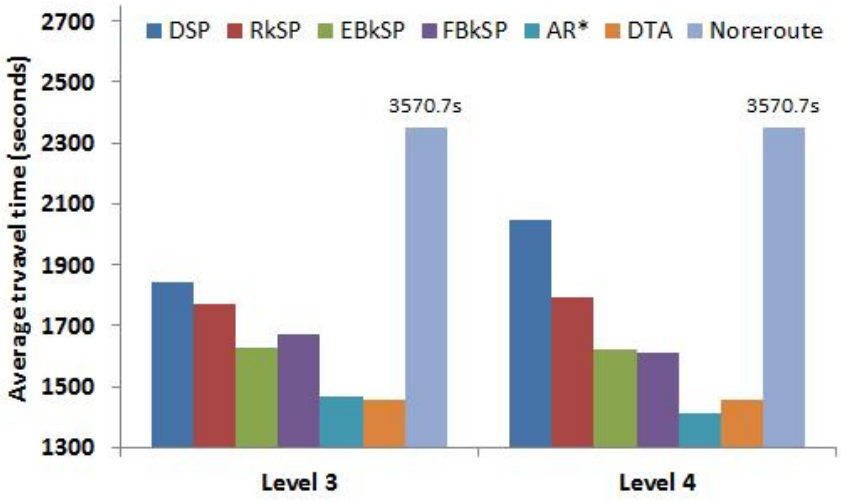
Figure 5.2 Traffic flow in the road networks.

generates a single, continuous simulation, i.e., without any other route changes as in the case of the proposed strategies. Hence, the CPU time measured in the next section indicates, in the case of DTA, the time required to produce the route file, whereas in the case of the proposed strategies, the cumulated time (i.e., over the whole simulation) required to compute alternative paths for the re-routed vehicles.

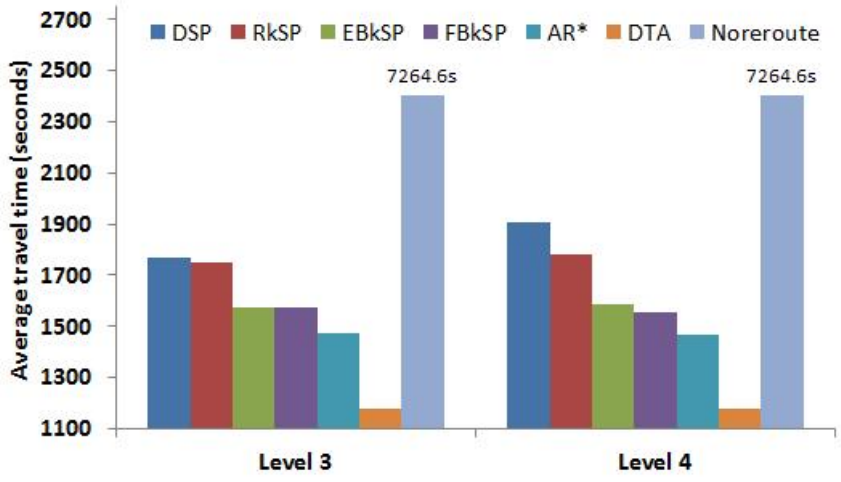
5.2 Results and Analysis

5.2.1 Average Travel Time

Figure 5.3 presents the average travel time obtained with the five strategies and with DTA on both networks. The "no-reroute" bars indicate the travel time in the absence of any re-routing. The results show that all the proposed strategies improve the travel time significantly. In most cases, the proposed strategies obtain travel times at least two times lower than no-rerouting. For instance, with a selection level of 3, compared to "no-reroute", EBkSP reduces the travel time by 2.2 times and 4.5 times on Brooklyn and Newark, respectively. As expected, DTA has the best average travel time since it can achieve user equilibrium. Based solely on the obtained



(a) Brooklyn



(b) Newark

Figure 5.3 Average travel time (L=(3,4), k=4, urgency=ACI, period=450s, δ=0.7, β=0.05).

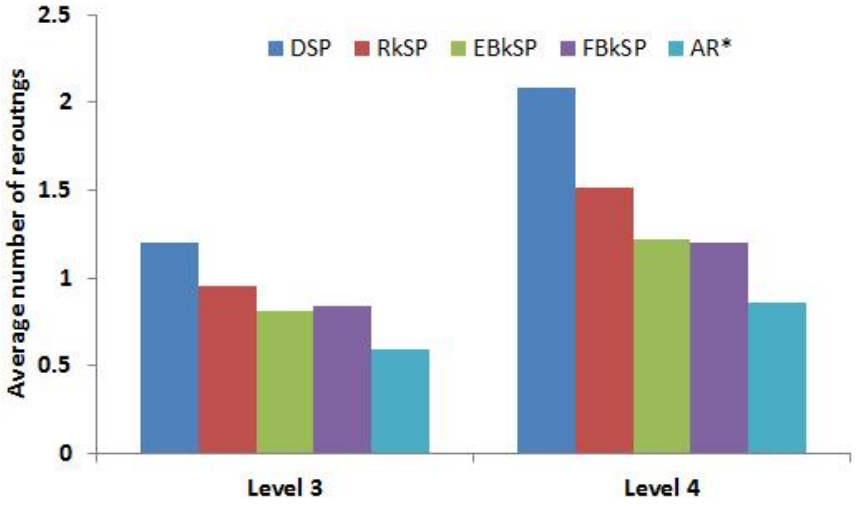
average travel time, the five strategies are ranked as follows: DTA>AR* >(EBkSP, FBkSP)>RkSP>DSP>no-rerouting. The results confirm the hypotheses laid out in Chapter 4 with statistical significance of 95% confidence interval. DSP can improve the travel time, since it re-routes dynamically the vehicles by considering the traffic conditions. However, in some cases, e.g., if many vehicles have similar current positions and destinations, respectively, new congestions can be created by the re-routing process. RkSP avoids this shortcoming since it balances the traffic flow over several paths. Nevertheless, a randomly picked path is not necessarily the best one. EBkSP and FBkSP offer even better performance by carefully selecting the path for each re-routed vehicle. Finally, AR* has the best performance among the proposed strategies as it considers all the other vehicles in the road network in the computation of a new route.

The experiments also demonstrated that setting the depth level to 3 or 4 is best for selecting a relatively optimal number of vehicles for re-routing (the two values lead to similar performance for Brooklyn, while level 3 is better for Newark). Lower level values do not select enough cars, whereas higher values increase the number of re-routings (see Figure 5.4). Therefore, the level parameter is set to 3 in the remaining experiments.

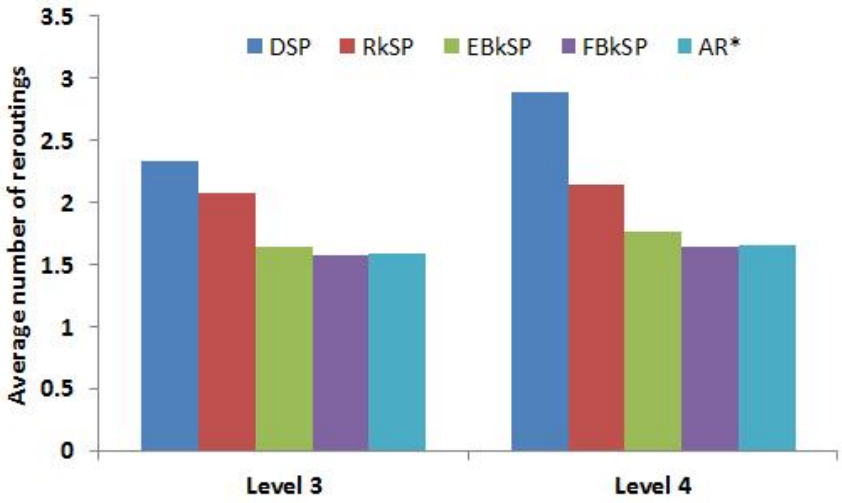
5.2.2 Average Number of Re-routings

It is important that the re-routings frequency for a given vehicle during a trip stay low. From the driver point of view, changing the path to the destination too often can be distracting and annoying. From the system point of view, having a low number of re-routings means decreasing the computational burden because the re-routing process is costly. Figure 5.4 compares the number of re-routings across the five proposed strategies. The statistical analysis shows that AR* <(EBkSP, FBkSP)<RkSP<DSP in terms of average rerouting number with 95% confidence interval [1]. For example,

[1] A t-test was performed for each pair of the strategies



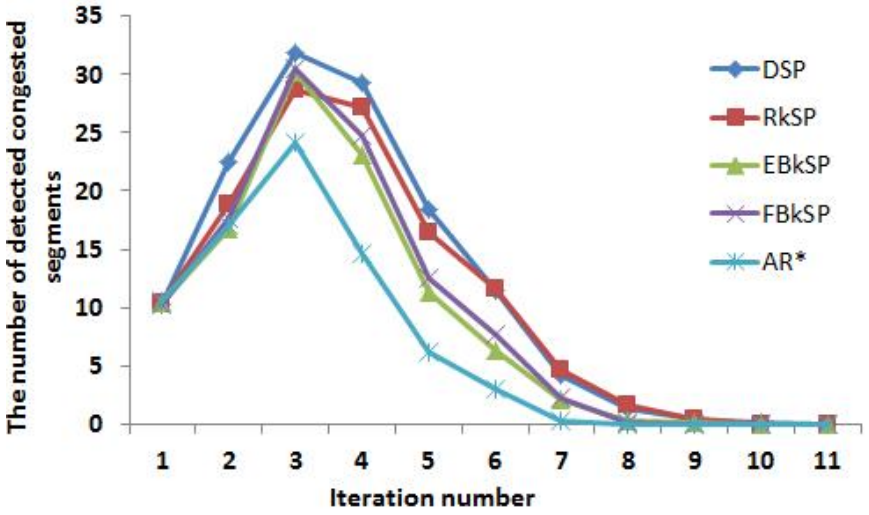
(a) Brooklyn



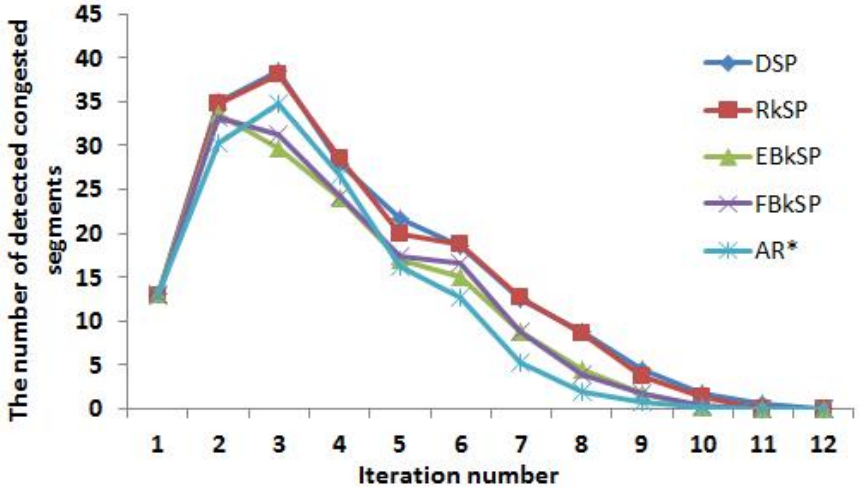
(b) Newark

Figure 5.4 Average number of re-routings (L=(3,4), k=4, urgency=ACI, period=450s, δ=0.7, β=0.05).

compared to DSP, AR* reduces the average number of re-routings by up to 2.0 and 1.5 times, while compared to RkSP, AR* is better by 1.6 and 1.3 times on Brooklyn and Newark, respectively. The reason is that by considering future path information in the re-routing decision, EBkSP, FBkSP and AR* can not only mitigate the current congestion, but also avoid creating new congestions; hence, the lower necessity for recurrent re-routing.



(a) Brooklyn



(b) Newark

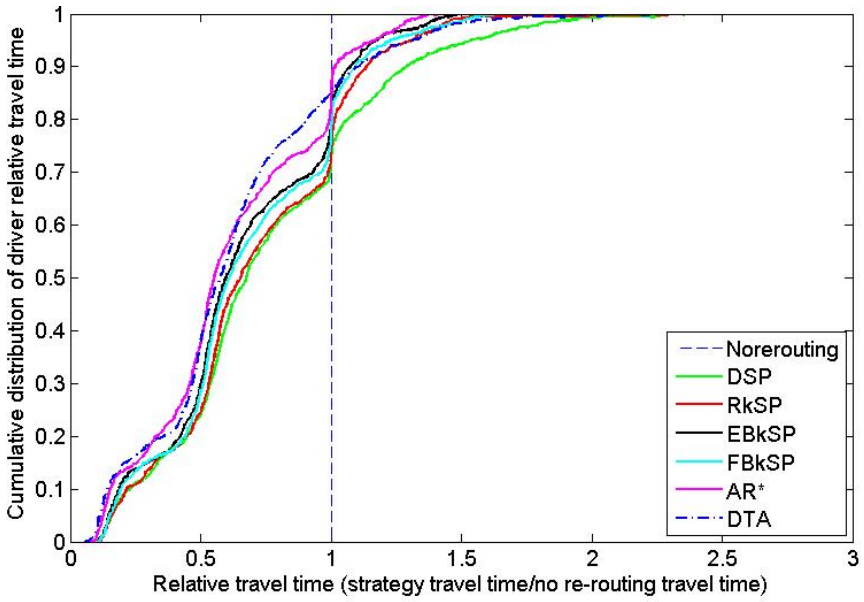
Figure 5.5 Number of congested road segments on the Brooklyn network over time/iterations. (L=3, k=4, urgency=ACI, period=450s, δ=0.7, β=0.05).

To confirm this analysis, the number of congested segments was also measured in each iteration for Brooklyn. Figure 5.5 shows the results. As traffic is generated during the first 1000 seconds (i.e., iterations 1-3), the number of congested roads increases for all strategies. Then, the number of congested roads decreases for two reasons. First, no more traffic is generated, and this effect is observed in the "no-rerouting" curv. Second, and more importantly for these strategies, re-routing helps to dramatically reduce this number. As expected, EBkSP and FBkSP have comparable results and reduce the number of congested roads faster than DSP and RkSP. It was also apparent that although AR* did not have the best performance at the beginning of the simulation, it was capable to alleviate congestion much faster than the other methods afterwards.

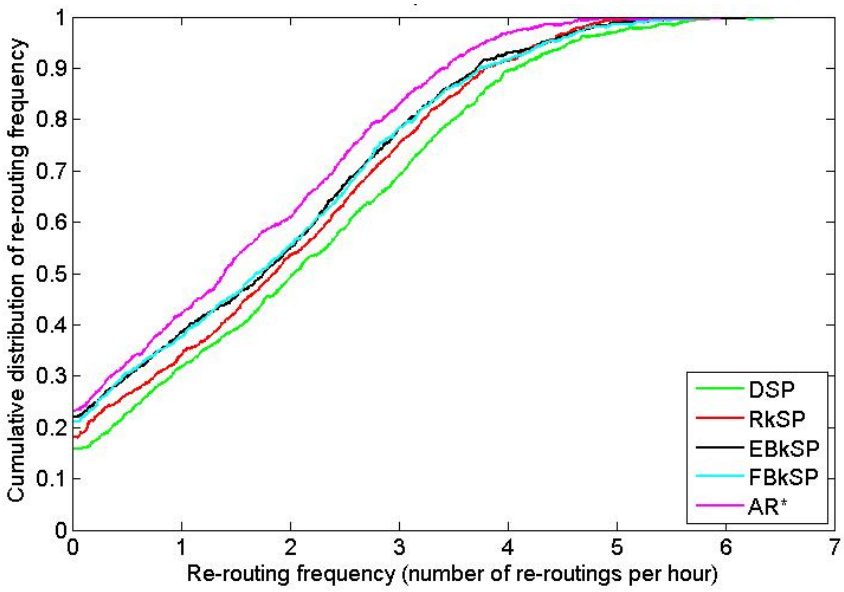
5.2.3 Distribution of Travel Time and Re-routing Frequency

The average travel time and the average number of re-routings measure the performance of the system from a global point of view. Here, the performance from a driver point of view (e.g., how many drivers end up with a shorter travel time?) is investigated. Two new metrics are also introduced: (1) the relative travel time (RelT) is defined as the ratio of the travel time with re-routing and the travel time with no re-routing; thus, RelT measures the travel time gains or loses for individual drivers; (2) the re-routing frequency (RRF) is defined as the number of re-routings per hour experienced by a driver; thus, RRF measures the driver distraction due to re-routing.

Figure 5.6 presents the cumulative distribution of RelT and RRF for each re-routing strategy for the Brooklyn network. The values are averages (per driver) computed across 20 runs of simulations. Similar results were obtained for the Newark network, which was omitted. AR* has the best results for both RelT and RRF, followed closely by EBkSP and FBkSP. The system manages to improve the travel time for a large majority of drivers. Similarly, a large majority of drivers experience no



(a) Relative travel time CDF



(b) Re-routing frequency CDF

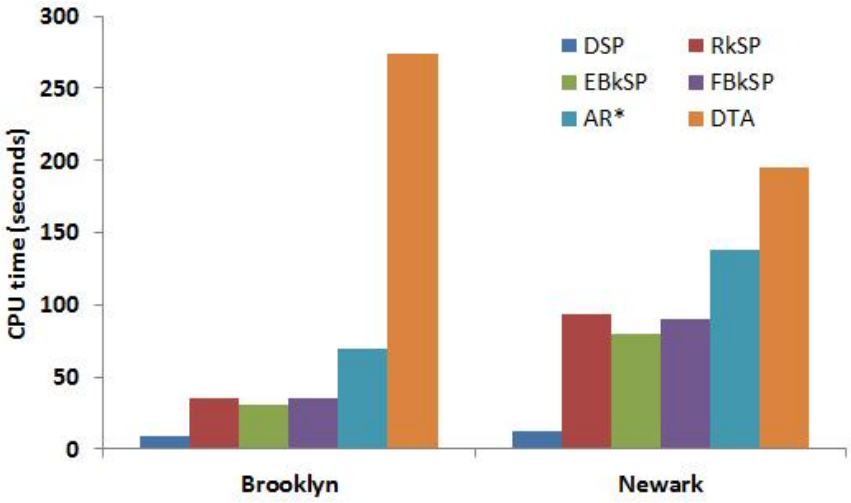
Figure 5.6 CDF of relative travel time and re-routing frequency per hour on Brooklyn network.(L=3, k=4, urgency=ACI, period=450s, δ=0.7, β=0.05).

more than 3 re-routings per hour, which is assumed to be acceptable in city scenarios with heavy traffic.

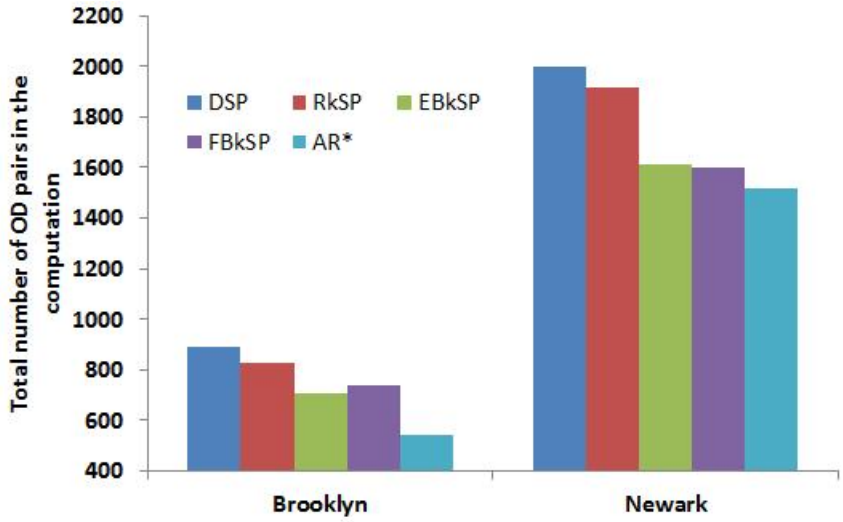
However, there is a relatively small percentage of drivers (i.e., ranging from 10% for AR* to 25% for DSP), that end up with increased travel time after re-routing. The observed increase is limited to less than 50% for most of these drivers. Note that this phenomenon is equally present in DTA, where around 15% of the drivers have increased travel time. The main reason for these results is that the proposed re-routing strategies have not been designed to achieve user-optimal equilibrium, and thus cannot guarantee the best travel time for each user. More surprisingly, even DTA which was designed to achieve user equilibrium cannot do it; reasonable conjecture is that this is due to the difficulty to find an equilibrium under congestion [16]. It is understood that a few bad experiences with the system could impact its adoption rate. Therefore, future studies should aim to investigate strategies to lower the number of drivers with increased travel time and to bound this increase to low values.

5.2.4 CPU Time

At this point, the results indicate that AR* produces the best average travel times (near to the DTA times), followed closely by EBkSP, FBkSP, and in some cases, by RkSP. An important question is what is the computational performance among all the proposed five strategies. If the computational complexity of the algorithms that the strategies are based on is considered, the complexities of the Dijkstra shortest path (used by DSP), k loopless shortest paths (used by RkSP, EBkSP and FBkSP) and A* (used by AR*) algorithms must be evaluated. Dijkstra shortest path and k loopless shortest paths require $O(E + Vlog(V))$ and $O(kV(E + Vlog(V)))$, respectively, while A* was proven to be faster than Dijkstra [75]. However, this complexity analysis is pertinent only when the selection of an alternative path for one single vehicle is considered. From the system point of view, the global computational complexity also



(a) CPU time



(b) Number of OD pairs

Figure 5.7 CPU time for both the networks(L=3, k=4, urgency=ACI, period=450s, δ=0.7, β=0.05).

depends on the number of re-routings processed in a time window; this number is a function of the number of congested road segments and the congestion severity (i.e., how many vehicles are selected for re-routing). Moreover, DSP, RkSP, FBkSP and EBkSP compute shortest paths after grouping the vehicles on their origin-destination, whereas AR* calculates a new path for each vehicle. Therefore, AR* could require a larger computation time than the other methods.

Figure 7.3 (a) shows the global CPU time consumed for re-routing by the five methods and by DTA. Note that the experiments were conducted on a 64 bit Ubuntu machine with Intel Core i5-2467M CPU (1.6GHz) and 4GB of memory. The following four observations were noted regarding the CPU time results:

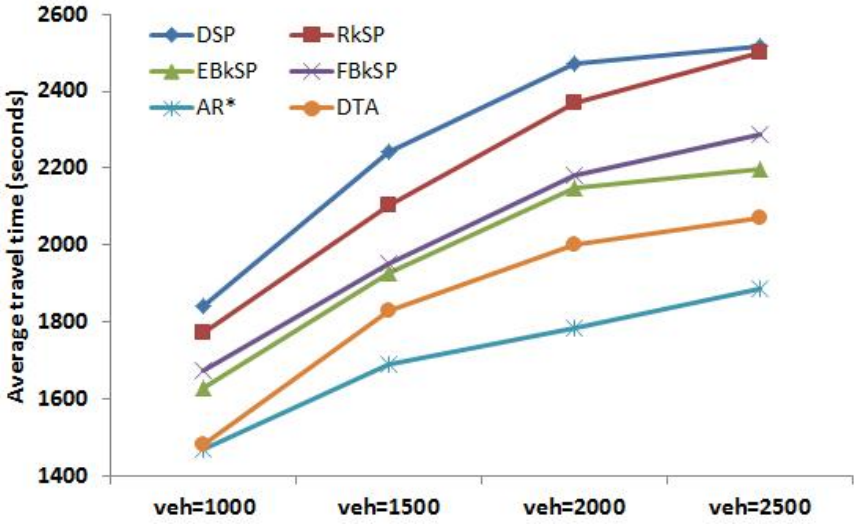
- DSP requires the least CPU time for re-routing, mainly due to the low complexity of the shortest path algorithm (compared to the k-shortest paths algorithm) and to grouping the re-routed vehicles.
- AR* consumes significantly more CPU time. Specifically, it requires 2.0 and 2.3 times more CPU time than RkSP and EBkSP on Brooklyn. The main reason is that AR* cannot group the re-routed vehicles like the other methods as stated in Section 4.1.2.
- EBkSP, FBkSP and RkSP are situated in between the above mentioned methods from the CPU time point of view. Interestingly, EBkSP and FBkSP require less computation time than RkSP even though they execute more complex path selection algorithms in addition to the k-shortest path computation. The explanation is that EBkSP and FBkSP decrease the total number of re-routings processed in a period. This decrease becomes apparent when the number of origin-destination (OD) pairs involved in the computation was examined, as indicated in Figure 7.3 (b). The total number of OD pairs is lower for EBkSP and FBkSP than for RkSP. Figure 7.3 (b) also shows that the although DSP leads to the largest number of OD pairs, it still has the lowest CPU time because of the much lower computational complexity of Dijkstra algorithm compared to the k-shortest path algorithm.
- DTA has the largest CPU time and scales poorly with an increasing number of vehicles (in terms of CPU time) when compared to AR* or the other proposed methods (as shown in Figure 5.8 (b)). Also, it is worth noticing that DTA assumes all vehicles in the system known at the beginning (i.e., when it computes its routes). However, in real life, vehicles may appear at any time, and DTA would be required to perform its expensive computation over and over again. Therefore, due to its very high computational cost in real life, DTA may be

impractical (i.e., it may not be able to compute alternative routes fast enough in order to mitigate congestions).

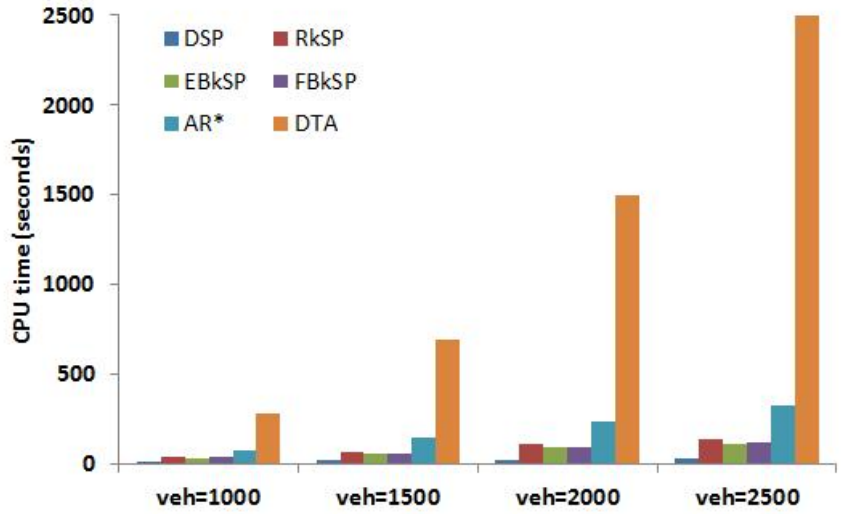
In conclusion, if both the average travel time and the CPU time are considered, EBkSP and FBkSP appear to be the best strategies since they offer the best trade-off between re-routing effectiveness and computational efficiency. If computational cost is not an issue, one can use the AR* strategy, while in the opposite case, DSP is the most appropriate choice.

5.2.5 Traffic Density

The results presented up to here already offer a good idea about the capabilities of the proposed re-routing strategies to alleviate traffic congestions. Yet there is an important aspect that still needs to be explored, i.e., how the proposed methods scale with the increase of the traffic volume. To respond to this question, another set of experiments were conducted on the Brooklyn network, where the number of vehicles were increased from 1000 to 2500. Figure 5.8 shows the obtained results both for the average travel time and the CPU time for different traffic densities. AR* and DTA present the best scalability from the average travel time point of view. However, these methods are also the least scalable from the CPU time point of view. It was apparent that DTA exhibits particularly poor scalability compared to the proposed strategies, confirming the hypothesis that DTA is not yet a suitable approach for real-time traffic management. Also, somewhat interestingly, AR* obtained better average travel times than DTA (see Figure 5.8 (a)) when the number of vehicles was above 1500. This is certainly due to the fact that the 50 iterations limit, set in the DTA tool is not sufficient to achieve user equilibrium for higher traffic densities. Therefore, a higher number of iteration is needed in this case, which will lead evidently to even higher CPU times.

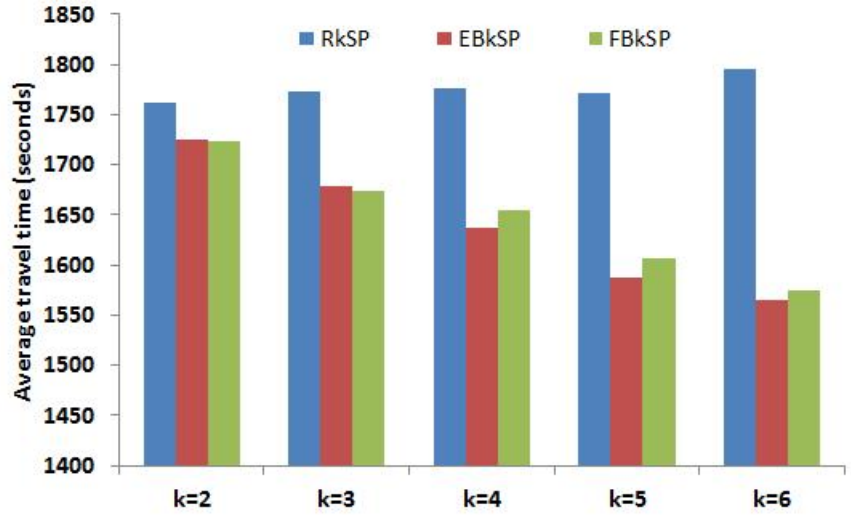


(a) Average travel time

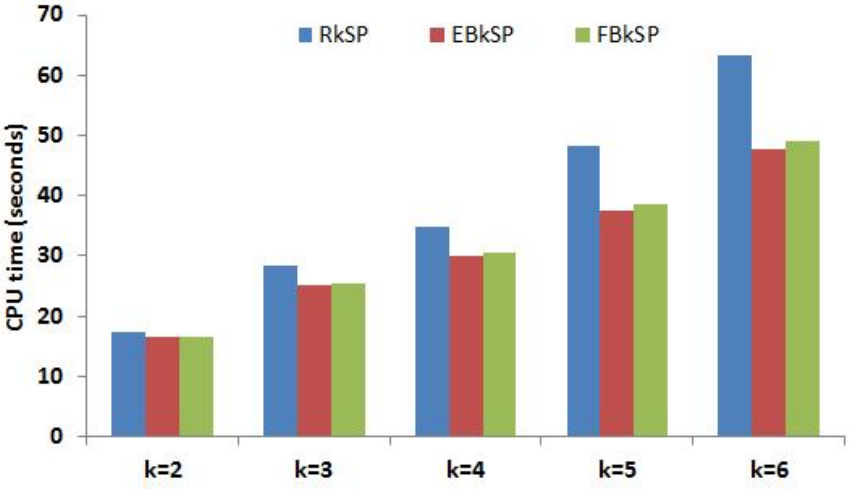


(b) CPU time

Figure 5.8 The average travel time and CPU time for Brooklyn network for different traffic densities (L=3, k=4, urgency=ACI, period=450s, δ=0.7, β=0.05).



(a) Average travel time



(b) CPU time

Figure 5.9 Average travel time, CPU time for RkSP, EBkSP and FBkSP as function of k for the Brooklyn network (L=3, k=(2, 3, 4, 5, 6), urgency=ACI, period=450s, δ=0.7).

5.2.6 Number of Alternative Paths

k is a determinant parameter for the performance of RkSP, FBkSP and EBkSP, which require k-shortest paths computation. A larger k value allows for better traffic balancing but introduces higher computational complexity. Furthermore, the maximum allowed difference between the slowest path and the fastest path is 20% in the setting. Therefore, large k values may not be necessary because they would lead to computing many useless paths. Figure 5.9 compares the performance of RkSP, EBkSP and FBkSP with different k values on the Brooklyn network. The k value is irrelevant for DSP and AR*.

It was observed that RkSP does not exhibit any performance improvement for $k > 2$, while both EBkSP and FBkSP consistently produce lower travel times with higher k values. From Figure 5.9 (b), it is apparent that the computational cost increases linearly with k for all the kSP methods. However, EBkSP and FBkSP are more scalable than RkSP especially for larger k values (e.g., EBkSP requires 32% less CPU time than RkSP when k equals 6). The efficiency of EBkSP and FBkSP is due to the reduction of the number of OD pairs.

In conclusion, EBkSP and FBkSP have a much more robust and efficient performance than RkSP with respect to different k values. Choosing the k value is a matter of trade-off between improving the average travel time and increasing the computational cost.

5.2.7 Urgency Function

Among the five proposed algorithms, EBkSP, FBkSP and AR* use an urgency function to sort the list of vehicles selected for re-routing (cf. Section 5.2.7). To measure the performance difference between the two proposed ranking policies – RCI and ACI (cf. Definition 1), the ANOVA statistics test was conducted over the average travel time from 30 simulations with EBkSP and FBkSP. The results show that ACI produces

lower average travel times than RCI ($p < 0.01$) in a 95% confidence interval. The result confirmed the previous analysis in Section . Thus, since using the absolute congestion impact on the travel time as priority measure for vehicle re-routing leads to a better global performance, ACI is used as the default urgency function in all the following experiments.

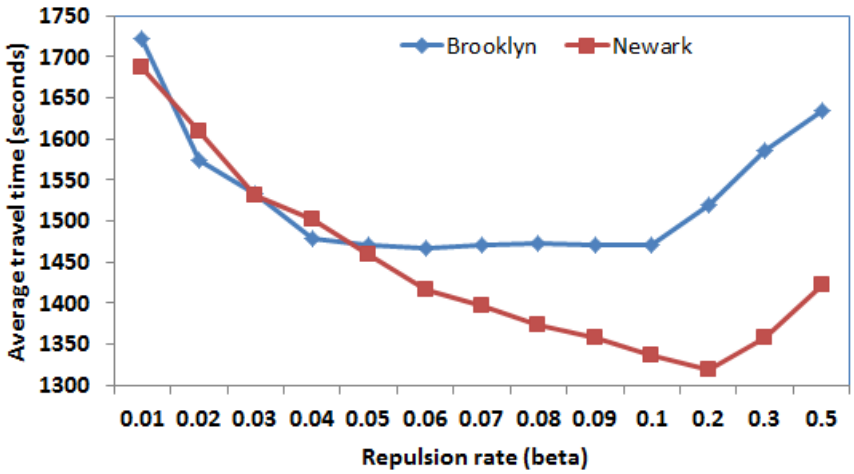
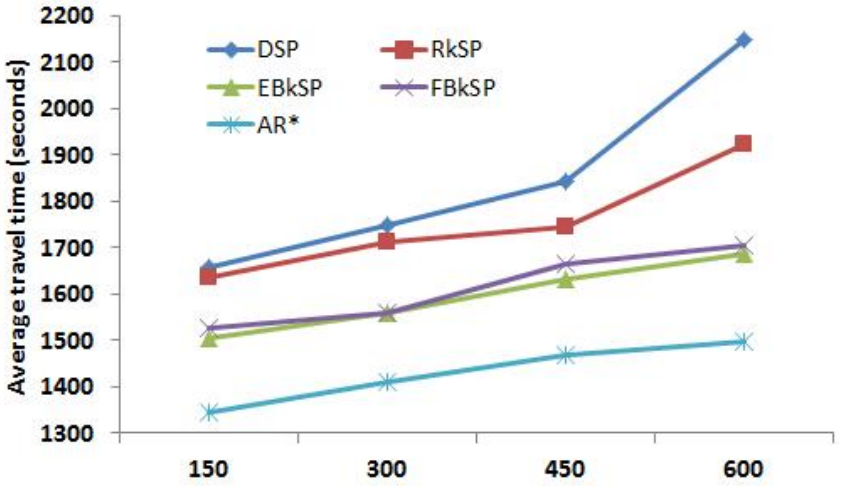


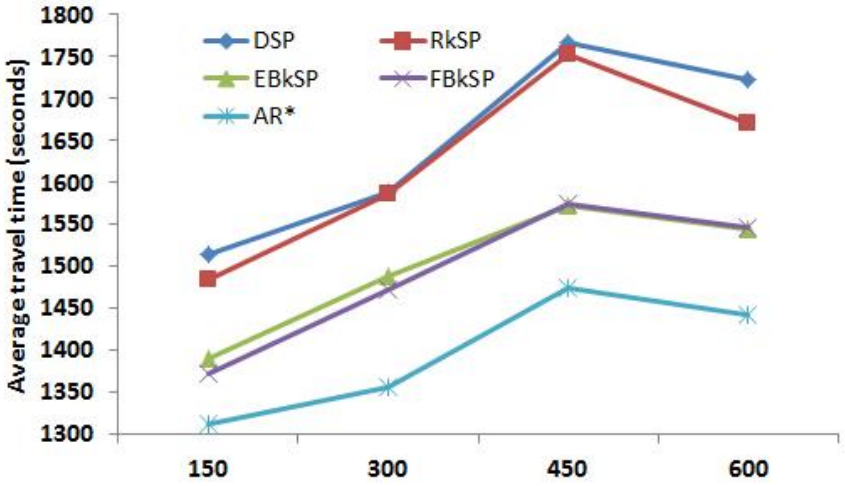
Figure 5.10 Average travel time as function of β for both networks. (L=3, k=4, urgency=ACI, period=450s, δ=0.7, $\beta \in$[0.01,0.5]).

5.2.8 The Weight of Repulsion in AR*

The β parameters directly impact the effectiveness of the AR* algorithm. Recall that in the original AR* algorithm, F(x)=(1-β)(G(x)+H(x))+β*R(x). If beta is too high, the resulting path will diverted too far away from the optimal. Similarly, if beta is too low, it will be too closer to the naive shortest path (DSP) strategy. Therefore, to determine a good β, beta value was varied from 0.01 towards 0.5 and measured the travel time. Figure 5.10 shows the results for both networks. An observation shows that value in [0.05, 0.2] leads to the lowest travel time on both networks. For all the experiments in the text, 0.05 was used as the beta value.



(a) Brooklyn



(b) Newark

Figure 5.11 Average travel time as function of the re-routing period for the Brooklyn network (L=3, k=4, urgency=ACI, period=(150s,300s,450s,600s,750s), δ=0.7).

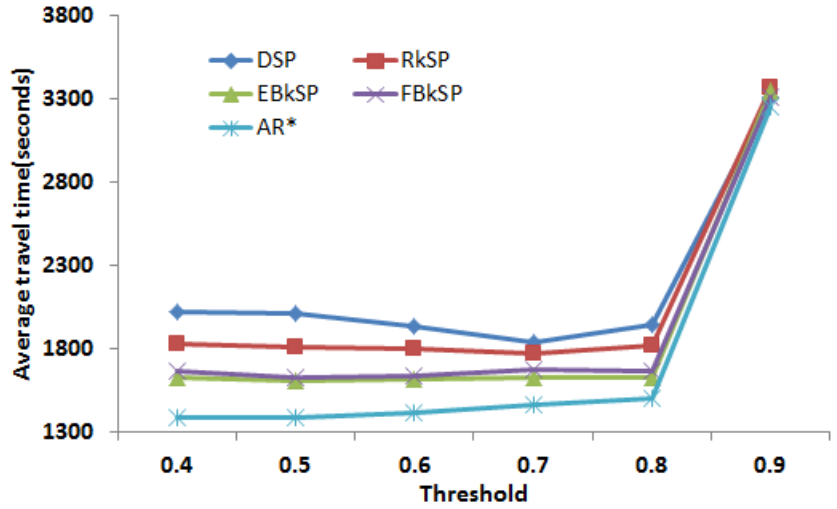
5.2.9 Re-routing Period

Within the traffic guidance system, the re-routing process is triggered periodically at a pre-defined time interval. A shorter re-routing period leads to higher reactivity of the system, and thus to better travel times. However, the price to pay is increased computation cost, communication overhead, and potentially re-routings. At extreme cases, it might not even be possible to compute the alternative routes fast enough to push them to vehicles before they reach the re-routing intersections.

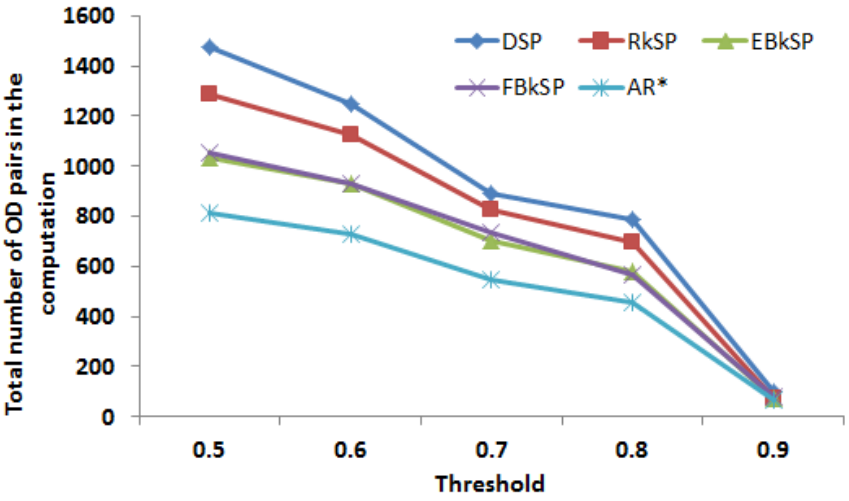
Figure 5.11 shows the average travel time for different re-routing periods. Generally, the lower the period is, the lower the average travel time is. The difference between all algorithms tends to be smaller when the re-routing period is low since the system is more reactive to congestions. As the period increases, the system's reactivity decreases, which translates into increased travel times. In particular, DSP and AR* were more sensitive to the variations of the re-routing period parameter especially in the Brooklyn experiments, while the KSP methods were generally more robust to such variations. To obtain a good balance between the average travel time on the one side and the computation cost, communication overhead, and number of re-routings on the other side, a period of 450 seconds was used on the experiments.

5.2.10 Congestion Threshold

Another key element determining the system reactivity is the congestion threshold (i.e., the vehicle density value on a road segment above which the system considers it as sign of congestion). Choosing a low density threshold may trigger unnecessary re-routing, which in turn leads to increased computational burden and number of re-routings. On the other hand, if the threshold is too large, congestion may be detected too late and the re-routing could be less effective. Figure 5.12 (a) confirms these hypotheses. When the threshold value is $\delta = 0.9$, the travel time increases dramatically because the congestion relief mechanism is triggered too late.



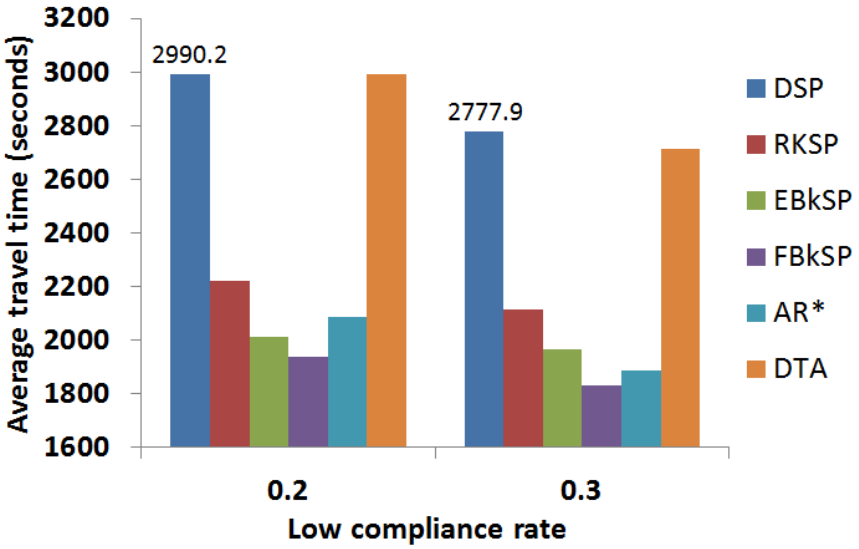
(a) Threshold



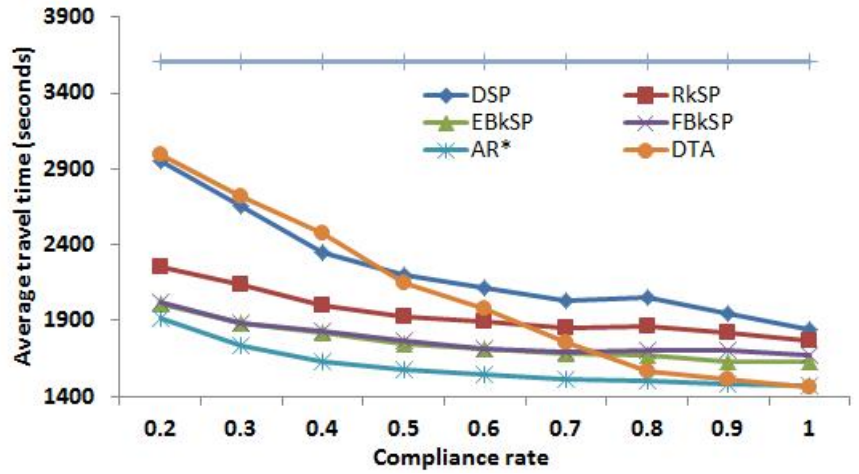
(b) OD pairs

Figure 5.12 Average travel time as function of the congestion threshold for the Brooklyn network (L=3, k=4, urgency=ACI, period=450s, δ=(0.4, 0.5, 0.6, 0.7, 0.8, 0.9)).

When the threshold value is too small, the number of re-routings increases as well as the computational burden. Figure 5.12 (b) shows the total number of origin and destination pairs that have to be computed in a simulation depending on the congestion threshold. The results show that the number of OD pairs is reduced by 58% when δ varies from 0.4 to 0.8. Therefore, $\delta = 0.7, 0.8$ are the preferred congestion threshold values since they offer smaller average travel times at reasonable computational cost.



(a) Low compliance rate



(b) All compliance rates

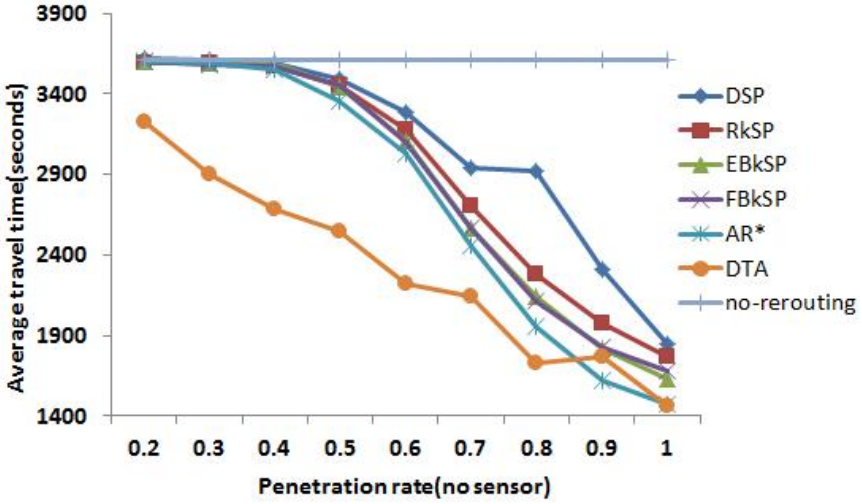
Figure 5.13 Average travel time as a function of the compliance rate on Brooklyn network. (L=3, k=4, urgency=ACI, period=450s, δ=0.7, β=0.05).

5.2.11 Compliance Rate

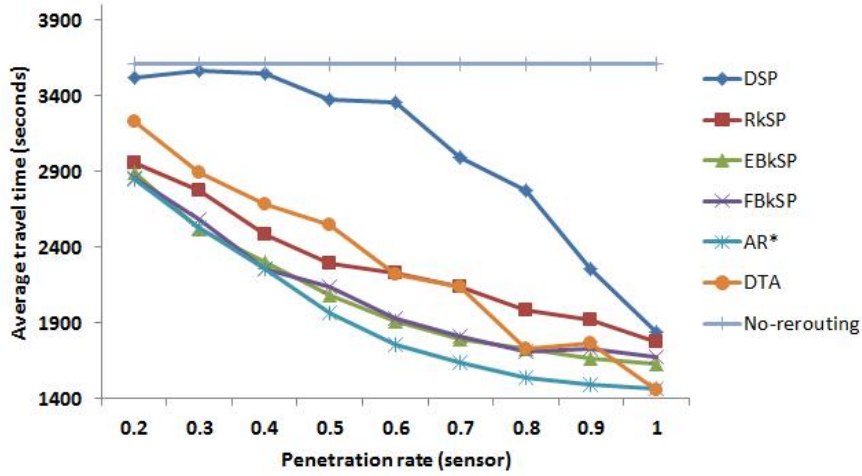
It is unrealistic to assume that every driver follows the re-routing guidance. The drivers' compliance rate (i.e., the possibility for the driver to accept the guidance) is an important factor for the re-routing strategy design. Therefore, the average travel time was measured, while varying the compliance rate for the five proposed strategies and for DTA. Specifically, for the strategies, given a compliance rate of $x\%$, at each re-routing period, each of the selected vehicles change their routes with x% possibility. As for DTA, $x\%$ of the vehicles are randomly selected to follow the DTA assigned route, while the rest of the vehicles follow the shortest time route.

Figure 5.13 (a) indicates that the average travel time can be significantly improved by all five strategies even under low compliance rates. This is due to the fact that even under low compliance rates, the drivers who comply with the guidance can still receive more rapid routes, which in turn can improve the travel time for the rest of the drivers. Figure 5.13 (b) shows the average travel time for a wide range of compliance rates.

In particular, when the compliance rate is low, RkSP, EBkSP, FBkSP and AR* show significantly better travel times than DTA. The reason is that when compliance is low, the drivers who comply benefit much more from this guidance than from the DTA guidance. In the DTA approach, the route computation is done once before any vehicle enters the network. If the compliance rate is low, the DTA computed routes are far from a user equilibrium, inclusively for the compliant drivers. Differently, these strategies can adjust the vehicles' routes periodically based on the current traffic information. Therefore, although the non-compliant drivers create congestion in the network, the compliant ones can still receive fairly good paths, which implicitly reduces the congestion level in the network.



(a) Sensors: Vehicles Only



(b) Sensors: Vehicles & Road-Side

Figure 5.14 Average travel time as a function of the penetration rate on Brooklyn network (L=3, k=4, urgency=ACI, period=450s, δ=0.7, β=0.05).

5.2.12 Penetration Rate

To understand how easy is to deploy this solution in real life, the effect of the penetration rate on the average travel time was studied. Specifically, before the system starts, each vehicle was predefined with x% possibility of owning the system(e.g. providing position information and receiving guidance). Once the system starts, only those the x% vehicle will have the chance to be rerouted. The penetration rate is a parameter of major importance for two reasons. First, since only the vehicles which have the system provide position and route information to the server, the accuracy of congestion detection and road travel time estimation depend directly on this parameter (i.e., the lower the penetration rate is, the lower the accuracy is and vice-versa). Second, similar to the compliance rate, the effectiveness of the load balancing mechanism implemented by the re-routing strategies increases with the percentage of the vehicles that use the system.

Figure 5.14 (a) shows the average travel time for various penetration rates, when traffic data are collected only from vehicles (i.e., no support from road-side sensors). When the penetration rate is low, the performance of the proposed methods is the same as "no-reroute". In this case, the service does not have enough data to accurately detect signs of congestion. Once the penetration rate is greater than 0.4, the system is able to improve the travel time. For penetration rates above 0.6, EBkSP, FBkSP and AR* start to perform better than DSP and RkSP, since a larger number of vehicles are re-routed, which requires a more advanced load balancing mechanism. Compared to these methods, DTA performs better under low penetration rates since the DTA re-routing is not triggered by the congestion detection as in this approach.

To boost the adoption of the system, it is possible that data from road-side sensors (in conjunction with data from vehicles) can be leveraged to detect congestion more accurately in case of low penetration rates. When road sensors are present, the road traffic density can be measured, the congestion can be detected and the

Table 5.3 Comparison between Compliance and Penetration Rate

	Road traffic density	**Congestion detection**	**Re-routing**	**Load balancing**
Compliance rate	Accurate	Accurate	All vehicles provide positioning information. At each re-routing, only x% of vehicles follow the guidance	Considers all vehicles (except DSP)
Penetration rate without sensors	Inaccurate	Inaccurate	Predefined x% of vehicles that provide positioning information receive guidance and comply with the guidance	Considers only the x% of vehicles (except DSP)
Penetration rate with sensors	Accurate	Accurate	Predefined x% of vehicles that receive guidance and comply with the guidance. The system obtains accurate traffic data from sensors and vehicles	Considers only the x% of vehicles (except DSP)

travel time can be estimated more accurately. Figure 5.14 (b) demonstrates that these methods can significantly improve the travel time even under low penetration rates if road-side sensor information is available. Moreover, EBkSP, FBkSP and AR* perform better than DTA in this case. When penetration rate is low (x%), DTA distributes evenly the x% vehicles without considering the rest, which can still create congestion. Therefore, the alternative routes proposed by DTA are not as effective in alleviating congestion. By comparison, these strategies can take advantage of the sensor information to divert the x% of the drivers and to reduce congestion.

To make it clear, the Table 5.3 lists the difference of these methodologies among compliance rate, penetration rate with sensor and no sensor.

5.3 Chapter Summary

In this chapter, an extensive evaluation of the proposed re-routing strategies was conducted. The objectives of the experiments are threefold. First, the effectiveness of the proposed methods to alleviate congestion and to improve the driver's travel time was measured. Then, the computational efficiency and the scalability of these re-routing methods was evaluated. To have a more precise idea of the performance level offered by the proposed methods, a state-of-the-art dynamic traffic assignment tool was used as a baseline in these experiments. The experimental results show that these methods are very effective in fighting congestion, being capable of improving the travel time as much as a DTA approach. In addition, these methods are much more computational efficient and scalable than DTA, which makes them more appropriate for use in real-life applications. Moreover, the palette of proposed methods offers a wide range of choices having different trade-offs between effectiveness and efficiency, which responds to a large variety of real-life scenarios.

Second, a more in-depth set of experiments was conducted to understand how the parameters used by these methods influence their performance. The experimental results give a good indication as to which are the most suitable values for some of these parameters (i.e., urgency function, re-routing period, congestion threshold) in conjunction with the employed method. For other parameters (i.e., number of alternative paths k) and algorithms (i.e., EBkSP and FBkSP), it is still a matter of choice between effectiveness and efficiency, opening the possibility to even finer system tuning.

Third, a more realistic scenario was considered and assessed the robustness of this system under compliance and penetration rates below 100%. The results indicate that this system can still improve significantly the travel time even under low compliance rates and that this approach is more robust than DTA. Besides, if accurate

traffic data (e.g., collected by road side sensors) can be provided to the system, then the system exhibits good robustness with various penetration rates.

Table 5.4 Comparison of all the Five Centralized Strategies

Effectiveness	AR* >(EBkSP, FBkSP)>RkSP>DSP
Efficiency	DSP>(EBkSP, FBkSP)>RkSP>AR*
Re-routing frequency	AR* >(EBkSP, FBkSP)>RkSP>DSP
Robustness	(AR*, EBkSP, FBkSP)>RkSP>DSP

To summarize the performance of the proposed strategies, they are ranked according to four criteria in Table 5.4, where $A > B$ means that strategy A is better than strategy B. While each method has its own advantages and shortcomings, the experimental results made us conclude that the entropy method (EBkSP) and the local optimization method (FBkSP), which led to very similar results, should be the preferred strategies, as they offer the best trade-off between performance and computation cost.

CHAPTER 6

HYBRID RE-ROUTING SYSTEM

Previously, the progress based on the centralized architecture was discussed, where several re-routing strategies were implemented to assign a new route to each re-routed vehicle, based on actual travel time in the road network. The re-routing strategies use load balancing heuristics to compute the optimal path for a given vehicle, to mitigate the potential congestion and to reduce the average travel time for all vehicles.

Despite these benefits, centralized solutions suffer from two intrinsic problems. Firstly, the central server has to perform intensive computation (to re-assign vehicles to new paths) and communication with the vehicles (to send the paths and to receive location updates) in real-time. This can make the centralized system infeasible because it may not be able to scale to large regions with many vehicles. Second, in the centralized system, the server requires the real-time locations as well as the origins and destinations of the vehicles to estimate the traffic conditions and provide effective individual re-routing guidance. This leads to major privacy concerns for the drivers and may prevent the adoption of such solutions. Increasing the computation and communication power of the server can only solve the first problem(at a significant monetary cost). As long as vehicles' traces are fully disclosed, user's identity can be easily inferred even if pseudonyms are used to communicate with the server [30]. This is due to the fact that location can contain identity information [68]. For example, if there is only one location update coming from a private property, then most likely it is sent by the owner of the property. Moreover, a sequence of location samples will eventually reveal the vehicle's identity [92]. Therefore, it is important to make the system work without disclosing the users' origin-destination (OD) pairs and with the least number of location updates along a user trip.

These requirements suggest a distributed system architecture. However, a fully decentralized architecture is not suitable for a proactive re-routing system. By creating ad hoc networks such as in [20,62], the vehicles can exchange information using multi-hop ad hoc communication. Therefore, vehicles can detect signs of congestion in small regions while preserving their privacy. However, ad hoc networks do not permit vehicles to get an accurate global traffic view in the road network, resulting in sub-optimal re-routing decisions. A Peer-to-Peer design [72] has the benefits of the ad hoc architecture and at the same time can acquire a global view of the traffic. Vehicles communicate with each other over the Internet and form peer-to-peer (P2P) networks for data collection and sharing. However, if designed naively, the system may not scale due to potentially large routing latencies in the P2P network. Besides, in a fully distributed architecture, due to the lack of a coordinator, the vehicles cannot take synchronized actions at the same time, which makes it difficult to make a collaborative decision.

To tackle all these problems, in this chapter a hybrid vehicular re-routing system is proposed for congestion avoidance. This is a hybrid system because it still uses a server to determine an accurate global view of the traffic. The centralized server acts as a coordinator that collects location reports, detects traffic congestion and distributes re-routing notifications (i.e., updated travel times in the road network) to the vehicles. However, the system off-loads, a large part of the re-routing computation at the vehicles and thus the re-routing process becomes practical in real-time. To make collaborative re-routing decisions, the vehicles situated in the same region exchange messages over VANETs. Also, the hybrid system implements a privacy enhancement protocol to protect the users' privacy, where each vehicle detects the road density locally using VANETs and anonymously reports data with a certain probability only from high traffic density roads. When signs of congestion are detected, the server sends the traffic map only to the vehicles that sent the latest updates. Subsequently,

those vehicles need to disseminate the traffic data received from the server in their region. User privacy is greatly improved, since this protocol dramatically reduces the number of vehicle location updates to the server, and therefore, the driver exposure and identification risks. Moreover, in the hybrid architecture, the server does not know the OD pairs of the users.

The remainder of the chapter is organized as below: Section 6.1 states the challenges of building a hybrid re-routing system. Section 6.2 presents the overview of the hybrid system followed by the privacy-aware traffic reporting in Section 6.3 and distributed re-routing strategies in Section 6.4. Afterwards, four optimization methods are described in Section 6.5 to reduce the VANETs communication overhead and improve network throughput. The chapter concludes in Section 6.6.

6.1 Challenges of the Hybrid System

The first challenge in the hybrid design is how to make the distributed strategies achieve similar effectiveness compared to the centralized version. Due to network issues such as contention and packet collisions, obstacles, disconnections, and limited throughput of longer multi-hop paths, each vehicle can only obtain a limited view of the other vehicles. However, the more information each vehicle can gather, the more accurate and optimal paths each vehicle can find. Therefore, the question is how to optimize the communication efficiency, to maximize the amount of information each vehicle can receive based on the current network condition. Several techniques are investigated in Section 6.5.

The traffic guidance system belongs to real-time continuous location based services, where location privacy protection is always the paramount challenge since the services require frequent location updates. One solution is to ensure each reported location is a cloaking box containing at least k users. However, this method does not provide a user k-anonymity protection, since a time-series sequence of cloaking boxes

forms a trajectory, which may reveal the real identity of the user if it links to the user's personal locations such as the home or office. Associating each cloaking box with a different pseudonym can not help in this case either: since successive locations are highly correlative, they could be re-linked based on a common trajectory without the need to know any identifiers [91]. Besides, existing methods use a proxy which must be trusted. Such an assumption should be avoided since any central entity could either become computation bottleneck or been comprised by adversaries. Therefore, new privacy methods are required by our system. These methods must protect user privacy, while at the same time, should not render the system ineffective.

There is a need to define a formal privacy measure to compare the privacy leakage between the centralized and the hybrid version of the system. The privacy measurement should consider the importance of each location update. Basically, each location update should be associated with a weight since some locations are more sensitive. For example, if there is only one location update in a private area, then the vehicle's identify is revealed. The more popular the location is, the more difficult it is for an adversary to single out the driver identity. Such index can be used to evaluate the degree of privacy leakage of existing and forthcoming location anonymization techniques, in terms of the trade-off between privacy and system performance. In this dissertation, Chapter 6.3 presents our privacy solution and metric.

6.2 System Overview

This section provides an overview of the proposed hybrid system. The design principles are described initially, and then the system architecture is presented.

6.2.1 Design Principles

The proposed system is built around two design principles corresponding to the two major requirements. First, the re-routing path computation should be off-loaded

from the central server to the vehicles to reduce the computation and communication burden on the server and achieve better scalability. Therefore, the alternative routes should be compute by vehicles themselves when there are signs of congestion in the road network. At the same time, the re-routing computation should be collaborative in order to achieve a better effectiveness of the re-routing process, as close as possible to the centralized re-routing effectiveness. To this end, the vehicles could exchange messages over VANETs and implement a distributed re-routing process. Second, the system should be designed to respect the privacy of the users from its conception, i.e., a privacy-by-design system, which can be essential for the wide acceptance of the system. Implicitly, by offloading the path computation to the vehicles, the drivers' exposure is reduced significantly, since very sensitive information (i.e., the OD pair of the vehicles) is not sent to the server anymore. Nevertheless, protecting only the OD of a vehicle is not sufficient. The vehicles still have to send traffic reports that are used by the server to obtain an accurate view of the traffic in the network. Frequent location updates may reveal the vehicle's identity [92]. Hence, a mechanism is also needed that protects the identity of vehicles while reporting data. The work in [35] provides a method for privacy-aware traffic reporting, based on virtual trip lines, and an associated cloaking technique. However, in [35] the vehicles only provide location updates from pre-defined geographical markers. On contrary, the system in this dissertation requires vehicles' location updates from any point in the network that is close to congestion spots in order to notify the concerned vehicles. Besides, the system proposed in [35] assumes a trusted proxy, which could become a communication bottleneck. Hence, a new solution is required.

6.2.2 System Architecture

Given the above described design principles, a hybrid architecture is proposed to implement the re-routing service as illustrated in Figure 6.1. The architecture is

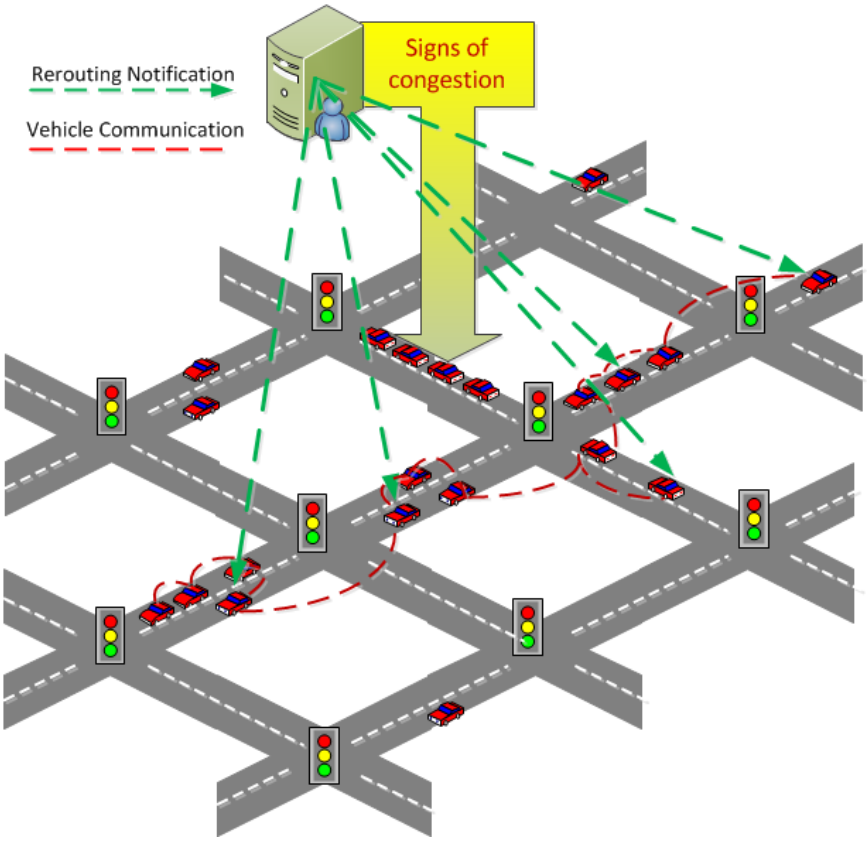


Figure 6.1 The hybrid system.

composed of a central server and a software stack running on an on-board device (e.g., a smartphone) in each vehicle that participates to the system. The system uses two types of communications. The vehicles communicate with the server over a 3/4G network to report local traffic density data and to receive the global traffic density in the road network (i.e., the green lines in Figure 6.1). The vehicles report data according to a privacy-aware algorithm that is detailed in Section 6.3. Also, the vehicles that are closely located communicate with each other over VANETs to determine the local traffic density, to disseminate the traffic data received from the server and to implement a collaborative and distributed re-routing strategy (i.e., the red lines in Figure 6.1) as detailed in Section 6.4.

The server uses the vehicle traffic reports to build an accurate and global view of the road network traffic. Each time new road segments exhibit congestion signs,

the server sends the weighted graph to the cars that reported recently, and are close to the congestion segments (see Section 6.3).

These vehicles disseminate the information (i.e., traffic graph and vehicle route) in their regions with a limited number of hops, to avoid excessive flooding. The dissemination also has a timeout (e.g, 0.2s), which is a constant parameter in the proposed system. When the time is up, based on the traffic graph and route information shared by other vehicles, each vehicle, whose current path traverses the congestion spot, locally computes a new route to its destination. This re-routing process is presented in Section 6.4.

6.3 Privacy-aware Traffic Reporting

The hybrid system described above distributes the re-routing computation to the vehicles. Consequently, the system does not require the OD pair of the vehicles. Yet, the system still requires each vehicle to periodically report their location to the central server, to compute the traffic density and estimate the travel time in the road network. The periodical location reporting, even anonymously, could lead to identity disclosure as proven in several papers [30, 68]. Therefore, to increase the vehicles' location privacy and hence protect the driver identity, this section introduces a privacy-aware communication protocol between the vehicles and the server. Firstly the privacy metric is described and then the density-based traffic reporting and the travel time computation by the server is presented.

6.3.1 Privacy Metric

In order to measure how much location privacy is lost with each location update, each location report is associated with a weight. Similar to [92], the weight of a location report depends on the popularity of the location road segment. That is, the more popular a spatial region is, the more difficult it is for an adversary to single out the

report sender. However, the number of vehicles along the segment is not sufficient to quantify its popularity, because some vehicles may have a dominant presence in that space. Instead, a metric is applied that is based on the entropy of the road segment.

Definition 4 - Road segment popularity. *Let rs be a road segment and $S(rs) = \{v_1, v_2, ..., v_m\}$ be the set of vehicles that send location updates in rs. Let n_i $(1 \leq i \leq m)$ be the number of location updates that user u_i sent from rs and $N = \sum_{i=1}^{m} n_i$. Then the entropy of the road segment rs is $E(rs) = -\sum_{i=1}^{m} \frac{n_i}{N} \log \frac{n_i}{N}$ and the popularity of rs is $P(rs) = 2^{E(rs)}$.*

Definition 5 - Privacy leakage. *Given the above defined popularity measure of a location, the global privacy leakage for vehicle v_i is defined as:*

$$pleak_{v_i} = \sum_{all\ updates} \frac{update_i}{P(location\ of\ update_i)}. \tag{6.1}$$

Definition 6 - Average privacy leakage. *The average privacy leakage for all the vehicles is:*

$$pleak_{avg} = \frac{\sum_{all\ vehicles} pleak_{v_i}}{total\ number\ of\ vehicles} \tag{6.2}$$

6.3.2 Density Reporting

Privacy-aware reporting is based on the observations that in dense areas, vehicles naturally experience a higher degree of anonymity similar to a person walking through an inner-city crowd. Hence, sending location reports from high density areas decreases the risk of vehicles being identified by the untrusted server. At the same time, the server has to compute the road network travel time and send it back to the vehicles. The re-routing effectiveness is highly dependent on the accuracy of the travel time

estimation in the network, as indicated in Chapter 4. Most importantly, the system has to be capable to detect the congestion signs to be effective. Therefore, a density-based traffic reporting mechanism is proposed wherein vehicles report to the server only if the road density is higher than a predefined threshold. This is beneficial for both the re-routing effectiveness and the vehicle privacy, since the server can still accurately detect the congestion signs at the cost of lower user privacy exposure.

A second important observation is that vehicles use VANETs to locally exchange messages. Therefore, each vehicle can obtain information about its neighbor vehicles by the periodic exchange of small presence messages called beacons. Based on the beacons, the density of the current situated road segment can be estimated locally by each vehicle [9]. The idea is to employ this mechanism to minimize the number of vehicle reports, i.e., only a fraction of the vehicles situated on a road segment will send traffic reports. Specifically, density reports are sent to the server conform to the following rules: (1) cars submit reports only when they perceive that the density on the road segments is above a threshold that would signal a chance of congestion, (2) cars decide probabilistically when to submit data as function of the density - i.e., the more cars there are, the fewer reports each car submits as the reports are distributed among the cars on the segment, (3) they send their messages through anonymizers (e.g., Tor [19]) to protect their identities.

Each vehicle periodically checks the number of vehicles N_i on the current road r_i. If $N_i \geq \theta * N_{max}$, where N_{max} is the maximal allowed vehicles on road r_i, then the vehicle will report the current detected density to the central server with probability $p = 1/N_i$, where θ is a system parameter threshold (e.g., 0.5). The server computes the traffic density on r_i as $d_i = N_i/L_i$, where L_i is the length of r_i. If every vehicle applies the same reporting procedure, then the probability for the server to receive x location reports from r_i is $\binom{n}{k}(1-p)^{N-x}p^x$. The expected number of updates on this road r_i is $nu_i = \sum_{x=0}^{N} x\binom{n}{k}(1-p)^{N-x}p^x$.

6.3.3 Report Collection and Travel Time Computation

The server receives reports from vehicles indicating the number of vehicles on a road segment and computes the traffic density on the roads. Every time the server receives a report concerning a road r_i, it will smooth the computed density value d_i using the following formula: $d_i^{current} = \alpha * d_i^{old} + (1 - \alpha) * d_i^{new}$. The value of α is experimentally chosen to be 0.05. The server then estimates the travel time of each road segment by considering the following three cases.

Case 1:
For the road segments without any traffic reports, the server estimates the travel time to be the free flow travel time. Note that this travel time is an approximate value for some roads, as vehicles only report when the vehicle density is above the threshold density.

Case 2:
For the road segments with a non-zero traffic density but for which the last report time is older than a predefined time interval, the server does an expiration operation. Specifically, if the difference between the last update time and the current time is greater than τ times the free flow travel time of the road, then the road density is reset to zero. The value for τ is also based on empirical results and is chosen to be equal to 4 in this system.

Case 3:
For the road segments that do not fit in Cases 1 and 2, the server uses the Greenshield model 3.1 presented in Section 3.1.2 to estimate the travel time.

Note that, because of the density-based reporting policy, the traffic density may not be fully accurate. However, the higher the traffic density of a road is, the more accurately the traffic density will be estimated. This is important since the re-routing effectiveness mainly depends on the traffic accuracy of the dense traffic roads. Chapter 7 shows that the loss of accuracy in the traffic view has only a marginal effect on the re-routing effectiveness, but greatly improves the privacy protection of the users.

6.4 Distributed Re-routing Strategies

If the server detects signs of congestion in the road network, it will alert the vehicles by sending the new computed travel times in the road network for all the roads that have a travel time different from the free flow travel time. The server sends messages only to the vehicles that reported most recently and that are located near a congestion spot, i.e., no further than three road segments from the congested segment. The server notification triggers the re-routing process that consists in a dissemination phase and a route computation phase. Besides, the dissemination phase has two sub-phases as presented in Algorithm 4. When a vehicle receives such a notification message either directly from the server or from the surrounding vehicles, it executes the procedure described in Algorithm 4. The first part of the procedure (lines 2-4 in Algorithm 4) consists in disseminating to other vehicles the updated travel times in the network. In the second part of the dissemination phase, the vehicles that received the notification broadcast personal route information to the other vehicles. The route information depends on the re-routing strategy employed by the system, i.e., either the k-shortest paths or the OD pair of the vehicle (lines 6-10 and 11-15 in Algorithm 4).

Algorithm 4 When Vehicle Receives a Congestion Notification

```
 1: procedure onCongestionNotification(updatedMap)
 2: updateTravelTime(updatedMap) {update the travel time of the map}
 3: T←computeBroadcastTime(this.rank) {compute when to start the broadcast based on
    this vehicle's rank}
 4: broadcastUpdatedMap(T) {broadcast the updated travel time map}
 5: if this.currentPath intersects congestionSpots then
 6:    if dEBkSP then
 7:       computekShortestPaths(this,k) {compute the k shortest path for itself}
 8:       wait(T_mapBroadcast) {wait until the map broadcast phase finishes}
 9:       broadcastkShortestPaths(T) {broadcast the k shortest paths at time T}
10:    end if
11:    if dAR* then
12:       getODPair(this) {get the OD pair for itself}
13:       wait(T_mapBroadcast) {wait until the map broadcast phase finishes}
14:       broadcastODPair(T) {broadcast OD pair at time T}
15:    end if
16: end if
```

On receiving a route information message, the vehicles store the received data as indicated in Algorithm 5. The received data will be used in the route computation phase to compute a new best path for the current vehicle. Note that only the vehicles whose current paths traverse a congestion spot participate in the route dissemination and computation phases (line 5 in Algorithm 4). In the remainder of this section two distributed re-routing strategies are presented allowing the vehicles to compute alternative paths in a collaborative manner when congestion happens.

Algorithm 5 When Vehicle Receives the Broadcast

1: **procedure** onReceived(vehiclemsg)
2: v=unpack(vehiclemsg) {unpack the message and extract the vehicle data, e.g., rank, k paths or OD pair}
3: receiveddata.push(v) {put the vehicle data into the priority queue based on the rank}

In particular, two strategies have proven to be the most effective in alleviating congestion: AR* and EBkSP. Thus, we implement their distributed versions.

EBkSP and AR* greatly improve the vehicles' travel time. However, these strategies are designed for a centralized architecture in which all the re-routing computation is done at the server side. The objective in this section is twofold. First, re-routing strategies are provided. they are based on the same ideas as the effective centralized strategies, yet can run in the proposed hybrid architecture. This is challenging, since the whole computation can only be done by the vehicles in order to comply with the system design principles (see Section 6.2.1). Second, the distributed re-routing should ideally have similar effectiveness as the centralized re-routing. In this section two distributed re-routing algorithms are introduced that are call dEBkSP (distributed EBkSP) and dAR* (distributed AR*). Subsequently several techniques in Section 6.5 are presented to optimize these strategies.

Both dEBkSP and dAR* require the re-routed vehicles to be ranked. In the centralized version, the rank of each vehicle is assigned by the server. Here, each vehicle picks a rank value that is randomly selected based on the estimated total

number of re-routed vehicles. A vehicle of a certain rank computes a new route by considering the higher ranked vehicle paths. Specifically, in the dEBkSP strategy each vehicle affected by congestion calculates k loop-less shortest paths based on its current origin and destination and the updated travel times in the road network. Then, the vehicles disseminate their rank and k shortest paths in their region for a predefined time interval (see Section 6.5). At the end of the route dissemination phase, each vehicle receives the k shortest paths of a certain number of vehicles in the region. However, given the nature of the dissemination process, the information gathered by a vehicle can be incomplete and different from one vehicle to another. In the final route computation phase, each vehicle iterates through the local sorted list of vehicles. It selects the (potentially) best path based on original EBkSP algorithm for each received vehicle with a higher rank and eventually selects the best path for itself (i.e., the least popular path among its k shortest paths).

Similarly, in the case of dAR* algorithm, the notified vehicle chooses a random rank but it does not compute the k shortest path. Instead, the notified vehicles only broadcast their OD pair. In the event of a broadcast timeout, for each received OD pair in the buffer, the vehicle applies the original AR* algorithm to compute a fake path. The current vehicle assumes that the vehicle with that OD pair will take that path. By the end of the process, the current vehicle computes a best shortest path for itself based on other vehicles' paths.

Algorithm 6 describes how dEBkSP and dAR* are executed when the information dissemination phase ends (i.e., timeout).

Algorithm 6 Distributed EBkSP and AR*

```
1: procedure onBroadcastTimeOut()
2: receiveddata {all the received data}
3: Q = empty {a queue that stores the vehicle objects that have already been processed}
4: while v_i!=this do
5:    v_i=receiveddata.pop()
6:    if dEBkSP then
7:       getkShortestPath(v_i) {get the k shortest path for this vehicle}
8:       doEBkSP(v_i, Q) {pick a path from the k paths for vehicle v_i based on vehicles'
         paths with higher rank}
9:       Q.push(v_i){label the vehicle v_i as a processed vehicle}
10:   end if
11:   if dAR* then
12:      getODpair(v_i) {get the OD pair for this vehicle}
13:      doAR(v_i, Q) {Compute a A star shortest path with repulsion for this vehicle based
         on vehicles' paths with higher rank }
14:      Q.push(v_i){label the vehicle v_i as a processed vehicle}
15:   end if
16: end while
17: if dEBkSP then
18:   getkShortestPath(this) {get the k shortest path for itself}
19:   doEBkSP(this, Q) {pick a path from the k paths for itself based on vehicles' paths
      with higher rank}
20: end if
21: if dAR* then
22:   getODpair(this) {get the OD pair for itself}
23:   doAR(this, Q) {Compute a A star shortest path with repulsion for itself based on
      vehicles' paths with higher rank }
24: end if
```

The key difference between dEBkSP and dAR* is the communication overhead and computation time. dEBkSP makes all the vehicles get involved in the computation process (k shortest paths). In the final timeout session, each vehicle only needs to pick a path. However, depending on the k value and path length, the packet is much larger than dAR* which only requires to broadcast its OD pair. Therefore, dAR* requires less communication overhead. In terms of computation time, dEBkSP gets all the vehicles involved in the path computation but dAR* requires the calculation of a path for each individual vehicle. Therefore, dEBkSP requires less computation time.

6.5 VANET Optimizations for Re-routing Information Sharing

The effectiveness of the distributed re-routing strategies presented in Section 6.4 mainly depends on the amplitude of the re-routing information dissemination among vehicles. The re-routing information dissemination has two dimensions that are related. The first dimension is represented by the total number of vehicles that receive re-routing information in a congested region. The second dimension regards the average volume of information received by the vehicles. Clearly, the higher the number of receiving vehicles and the higher the amount of information are, the more effective the re-routing process is. Ideally, each vehicle affected by congestion should receive re-routing information about all the vehicles in their region. In this case, the re-routing process can have a similar effectiveness with a centralized re-routing. However, achieving this level of dissemination in VANETs is challenging for two main reasons. First, the data dissemination has to be done in real-time and therefore the dissemination time interval is short. Typically, the data dissemination phase is limited 0.2 seconds in the system. Second, regular data dissemination in VANETs exhibits poor performance in congested areas because of contention (i.e., many vehicles try to communicate at the same time) [63]. In this section four optimization techniques are presented and implemented in the system to improve the data dissemination efficiency. These techniques are i) the prioritized broadcast, ii) the k-shortest paths compression, iii)the packet XOR for loss recovery and iv) the distance-based timer.

6.5.1 Prioritized Dissemination

The proposed system uses a prioritized dissemination to avoid that all the notified vehicles in a region start to broadcast at the same time, and thus reduce the network contention. When receiving a congestion notification, vehicle v_i waits T_i seconds before broadcasting its OD pair or its k-shortest paths. The waiting time is determined based on the rank of the vehicle defined in Section 6.4. The rationale is that in the

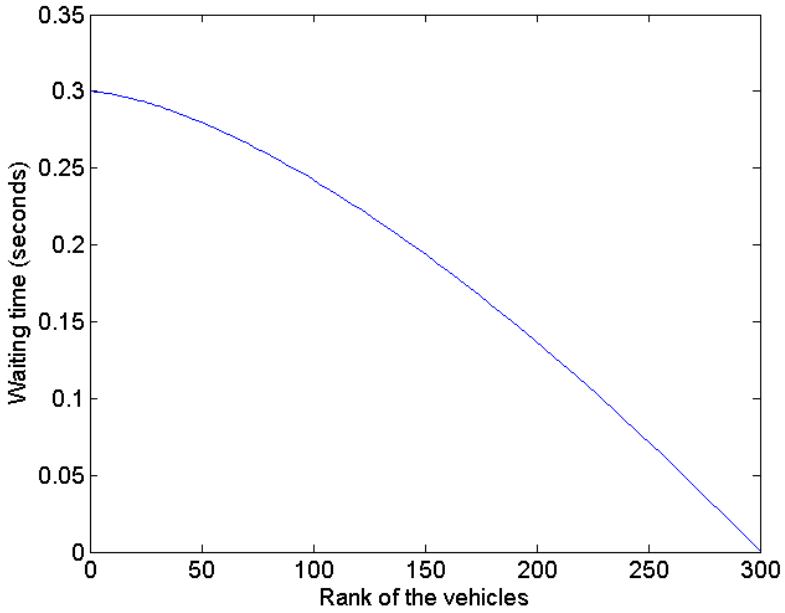


Figure 6.2 The ranking function.

proposed re-routing strategies the higher rank vehicle information is more important since each vehicle computes its own path based on the higher ranked vehicle paths. Therefore, the waiting time function in Formula 6.3 gives the higher ranked vehicles more time to disseminate their route data:

$$t_i = \alpha * rank_i^{\gamma} + T_{max}, \alpha = -T_{max}/rank_{max}^{\gamma} \tag{6.3}$$

T_{max} is the total dissemination time introduced in Section 6.2.2, i.e., the time after which everyone stops disseminating and starts computing the new route. $rank_i$ is the rank of vehicle v_i and $rank_{max}$ is the maximum rank of all vehicles. $rank_{max}$ is estimated by each vehicle from the road network density data received at the beginning of the re-routing process. γ is a predefined system parameter that is set to be 1.5 in the system settings. The waiting function has the following properties: i) the waiting time t_i for each vehicle is a value in the interval [0, T_{max}]; ii) the higher ranked vehicles

wait less time than the lower ranked vehicles. Specifically, the vehicle with maximum rank transmits without waiting, while the vehicle with lowest rank waits T_{max} time.

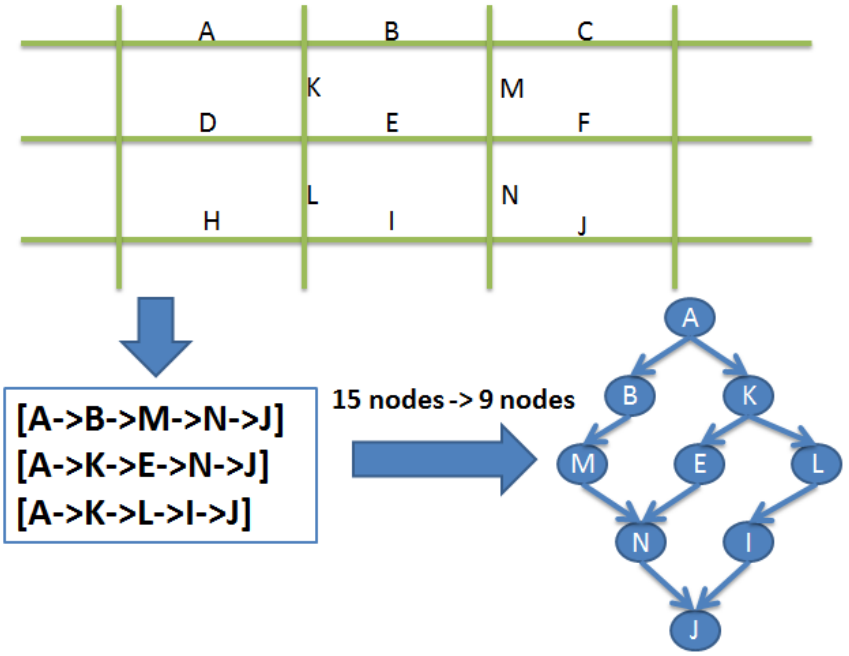
Figure 6.2 illustrates curve of the ranking function when T_{max} is 0.3s and $rank_{max}$ is 300. As shown, if the vehicle has higher rank such as 300, it waits zero time and starts broadcast immediately. Conversely, if the vehicle's rank is low, it it waits a long time before broadcasting.

6.5.2 K Path Compression

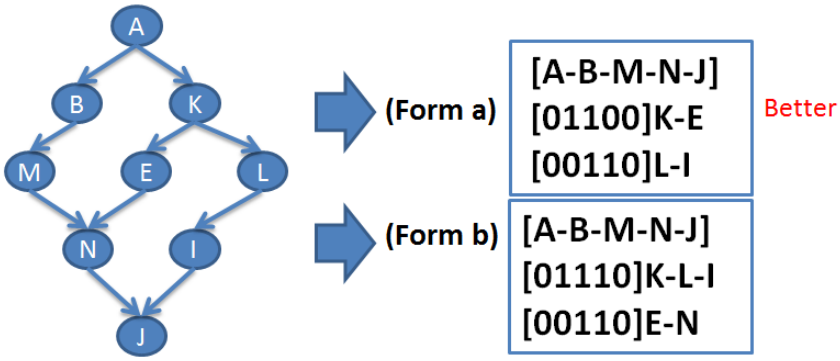
Another option to optimize the information dissemination between vehicles is data compression. On the one hand, a large packet size increases the communication overhead and decreases the dissemination effectiveness. On the other hand, as indicated in [62], the MAC Layer protocol (e.g., 801.11b protocol) limits the payload of a packet that can be sent on the communication channel (e.g., the maximum packet size is 2312 bytes in 802.11b protocol). The dAR* re-routing strategy is very efficient from the communication point of view since vehicles only disseminate their OD pair. However, this is not the case of dEBkSP algorithm that requires vehicles to transmit their k-shortest paths. Hence, depending on the k value and the distance between the origin and destination, the size of the messages can be large. Since the k-shortest paths generally present a high degree of overlapping, a compression algorithm is proposed to exploit this feature of the k-shortest paths.

Figure 6.3(a) shows a simple example of the potential space saving that could be obtained for three paths between the roads A and J. If all the three paths are naively broadcasted, 15 spaces are needed in total. However, the three paths only cover 9 distinct roads and therefore optimally need 9 spaces to be transmitted. Therefore, the question is how to obtain a compact representation of the k paths without any loss of information. The idea is to represent only the differences between the k paths. Specifically, if considering the first (shortest) path, for the next path only the edges

that are different from the previous path are needed to be indicated. Furthermore, a bit vector is used to represent the position of the edges. As depicted in Figure 6.3(b), the three paths can be represented either in form (a) or form (b). The '1' in the bit vector of each row indicates that the edge in that position is a different edge compared to the previous path. Obviously, form (a) is better than form (b), since form (a) uses one space less. Then, the problem comes down to finding the sequence of the k



(a)



(b)

Figure 6.3 K path compression.

paths resulting in the best compression. However, this problem is reducible to the Hamilton Path problem and therefore it is NP complete. Hence, a "greedy" algorithm is described to iteratively compress the k path, based on the number of overlapping edges as show in Algorithm 7.

Algorithm 7 K Path Compression

```
1: procedure compresskpath()
2: k=KPahths.size() {the number of k}
3: P= KPaths[0] {the shortest path}
4: while k>0 do
5:    P_k=FindMostOverlappingPath(P)
6:    P=Compress(P,P_k)
7:    k=k-1
8: end while
```

6.5.3 XOR Coding for Packet Loss Recovery

Data dissemination in VANETs can be significantly affected by packet losses. To allow vehicles to recover lost packets, a supplementary XOR coding field is appended to each transmitted packet similar to the work in [41]. Specifically, before a packet is forwarded by a vehicle, the vehicle searches possible XOR codings among the received messages in its buffer and appends the result to the packet. When receiving a packet, a vehicle may recover one of the lost packets from the XOR field. This technique helps reducing the number of message transmissions and also improves vehicles' knowledge coverage.

Figure 6.4 illustrates this concept wherein four vehicles v1, v2, v3, v4 are presented on road A and B. At T0 packet p3 arrives at v1. Because of packet loss, the packets that vehicle has are: v1: p0, p1, p2, v2: p1, p2, v3: p0, p1, and v4: p0, p2. At time T1, v1 broadcasts the latest received packet p3 with an appended field p0$\oplus$p1$\oplus$p2. Hence, both v2 and v3 can recover their missing packet p0 and p2 respectively from p3. Without a XOR coding field, two messages would be required to recover p0 in v2 and p2 in v3. Similarly, at T2, v2 broadcasts p3 with the coding field

p0⊕p1, so that v4 can recover p1 as well. The question is how to figure out which packets should be coded together? For example, if v1 appends coding field p0⊕p1 instead of p0⊕p1⊕p2, then only v2 can recover packet p0, but v3 can not recover p2. Therefore, to find the most efficient coding, each vehicle needs to be aware of the other vehicle's packets.

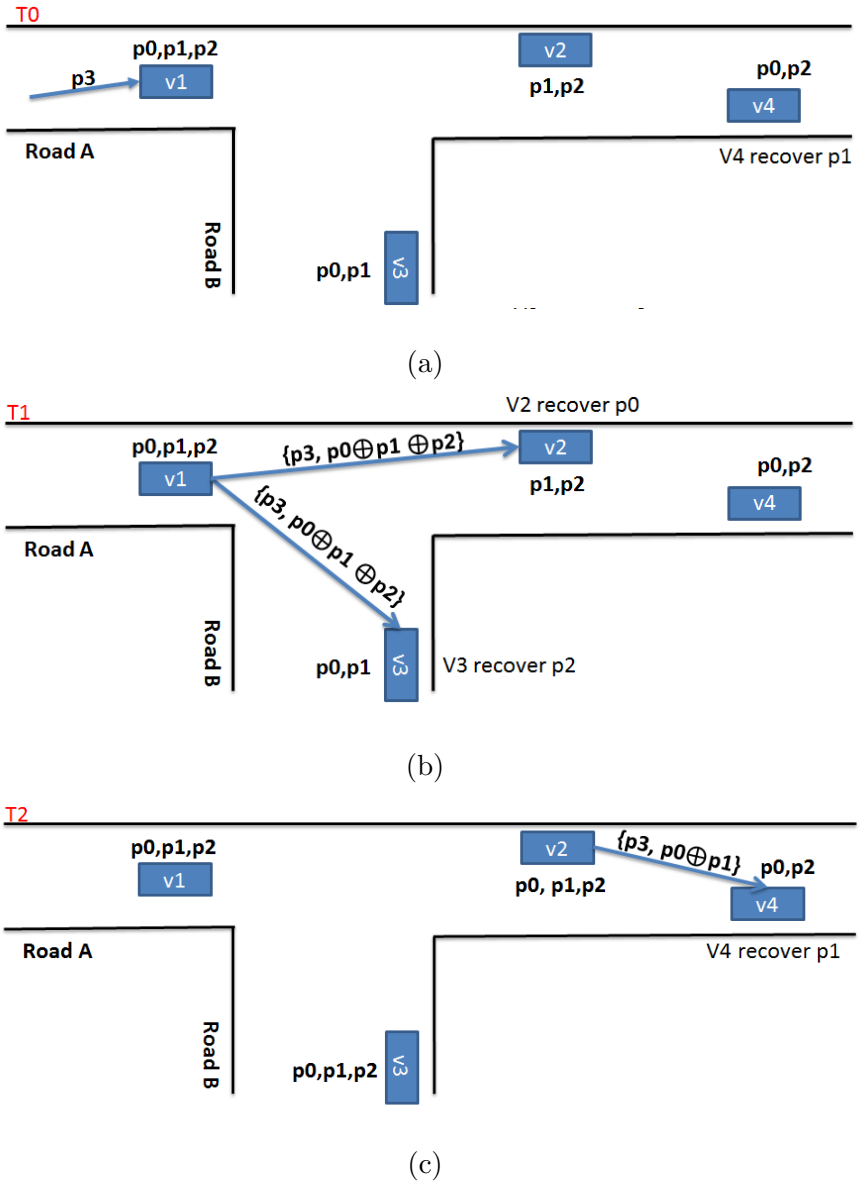


Figure 6.4 Message forwarding with network coding field.

Since wireless networks are based on broadcast, from channel eavesdropping, vehicles can obtains a certain amount of knowledge of the neighborhood. To further improve this knowledge, a Bloom filter is used here for a compact representation of the knowledge of each vehicle. Each time a packet is sent, the forwarding vehicle attaches a Bloom filter encoding the set of vehicles' identifiers it has received so far. This is similar to the distributed caching solution in [15]. When a vehicle A receives a packet from a neighbor vehicle B, it obtains the current knowledge of vehicle B from the Bloom filter. In this way, each vehicle can build up a neighbor table containing the knowledge of its neighbors. This table is essential in finding the best coding according to COPE [41], and it is employed as described in Algorithm 8. Specifically, to determine the best XOR coding field, each vehicle uses its neighbor table to check if a neighbor can decode P$\bigoplus$Q (line 6), where P and Q are two vehicle identifiers in the received data buffer.

Algorithm 8 Append Coding Field

```
 1: procedure setcodingfield()
 2: receiveddata {all the received data}
 3: P = receiveddata.pop()
 4: For each neighbor i in neighbor table
 5: repeat
 6:    while Q=receiveddata.pop() do
 7:       if neighour i candecode (P xor Q) then
 8:          P=P xor Q
 9:       end if
10:    end while
11:    i=i+1
12: until i=neighourtable.end
13: End for
```

6.5.4 Distance-based Timer for Message Broadcast

In the system, a distance-based timer approach is proposed similarly to [64] to reduce excessive broadcasting when multiple vehicles are within communication range. After receiving a broadcast message, the vehicle waits for a certain time period until re-broadcasting the message. The waiting time period is inversely proportional to the

distance between the receiving vehicle and the source vehicle. Therefore, a vehicle that is further from the message source should re-broadcast the message earlier. During the waiting period, if the current vehicle receives copies of the message, it means that some other vehicle has already forwarded the message. Thus, the current vehicle drops the message from its own buffer.

6.6 Chapter Summary

In this chapter, a hybrid re-routing system is detailed to provide better scalability and privacy. This system is hybrid since it utilizes both Internet and VANET communication, and a central server is still involved to estimate the global traffic view. Vehicles send privacy-aware traffic reports to the central server and path computation is off-loaded to VANETs. The privacy-aware traffic reporting is a function of road density. While distributed re-routing algorithms run on individual vehicles, four optimization approaches are exploited to improve the efficiency of the re-routing information dissemination in VANETs.

CHAPTER 7

EVALUATION OF HYBRID SYSTEM

The main objective of the simulation-based evaluation is to study the performance of the distributed re-routing strategies in the hybrid system. Specifically, the evaluation has four goals:

- Assess the effectiveness and efficiency of the proposed hybrid system compared to the centralized one.
- Investigate the performance difference between the hybrid system with and without privacy-aware traffic reporting.
- Quantify the strength of the privacy protection mechanism.
- Understand which VANET optimizations provide the most benefits.

The remainder of the chapter is organized as below: Section 7.1 describes the simulation settings. Section 7.2 presents the experiment results, analysis and discussion. The chapter concludes in Section 7.3.

7.1 Experimental Settings

The simulator veins [6] is used to implement the experiments. Veins is an open source Inter-Vehicular Communication simulation framework composed of an event-based network simulator Omnet++ [7] and a road traffic micro-simulation model SUMO [11]. It provides perfect interfaces for both communication among vehicles through VANETs and vehicles to SUMO [86]. Road traffic simulation is performed by SUMO 16.0, which is well-established in the domain of traffic engineering. Network simulation is performed by Omnet++ along with the physical layer modeling toolkit MiXiM [88].

The Brooklyn road network described in Chapter 5 is used for these experiments in this case. The traffic flow and network graph is the same as in the centralized

experiments. Table 7.1 and 7.2 shows the parameters used in the distributed re-routing algorithm and in the Omnet++ simulator, respectively.

7.2 Results and Analysis

7.2.1 Average Travel Time

Figure 7.1 shows the average travel time for the hybrid system compared to the centralized system. The hybrid scheme achieves similar travel time as the centralized version for both dEBkSP and dAR* strategies, and both distributed strategies improve the travel time by more than 200% compared to the no re-routing case. Specifically,

Table 7.1 Parameters in Distributed Re-routing Algorithms

period	The frequency of triggering the re-routing; by default period=450s
threshold δ	Congestion threshold; if $K_i/K_{jam} > \delta$, the road segment is considered congested; by default $\delta = 0.7$
level L	Network depth to select vehicles for re-routing starting from the congested segment; by default $L = 3$
# paths k	The max number of alternative paths for each vehicle; by default $k = 8$
repulsion weight β	The weight of repulsion in dAR*; by default $\beta = 0.05$
broadcast timeout T_{max}	The maximum duration of the broadcast of the trip data, by default $T_{max} = 0.2$s
privacy threshold θ	The privacy threshold; if $K_i/K_{jam} > \theta$, the vehicle starts to report the road density in the privacy enhancement component; by default $\theta = 0.5$

Table 7.2 Simulation Parameters

Channel frequency	5.890e9 Hz
Propagation model	Two ray
Fading model	Jakes' model rayleigh fading
Shadowing model	LogNormal
Antenna model	Omnidirectional
Transmission power	20mW
Propagation distance	500m
Maximum hop	10
PHY model	802.11p
MAC model	EDCA

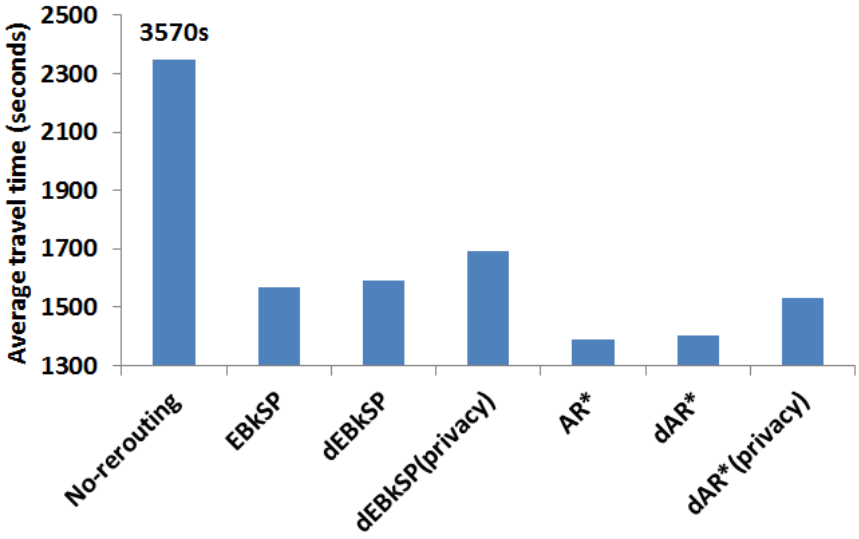


Figure 7.1 Average travel time (T_{max}=0.2s, k=8).

the travel time for dEBkSP without privacy-aware traffic reporting is only 1.4% less than the time for EBkSP. When privacy-aware reporting is used, the performance decrease is 6.6%. Similar results are obtained for dAR*, with performance decrease of 0.9% without privacy and 10.2% with privacy, respectively.

Three lessons are learned from the results: (1) Each vehicle only needs to know the trip information of its neighboring vehicles to make re-routing work effectively. Global information about all the vehicle routes brings minimal benefits. (2) The re-routing with privacy-aware reporting is less effective because the server may misestimate the travel time on certain road segments. However, the benefits of providing higher privacy overcome this small performance loss. (3) dAR* achieves better performance than dEBkSP. This is due to the tiny packet size (OD pairs) of the dAR*, which leads to less contention and thus fewer packet losses. Hence, more re-routing information is successfully shared among vehicles.

7.2.2 Distribution of Travel Time

The average travel time measures the performance of the system from a global point of view. Here, the performance from a driver point of view is investigated. Relative

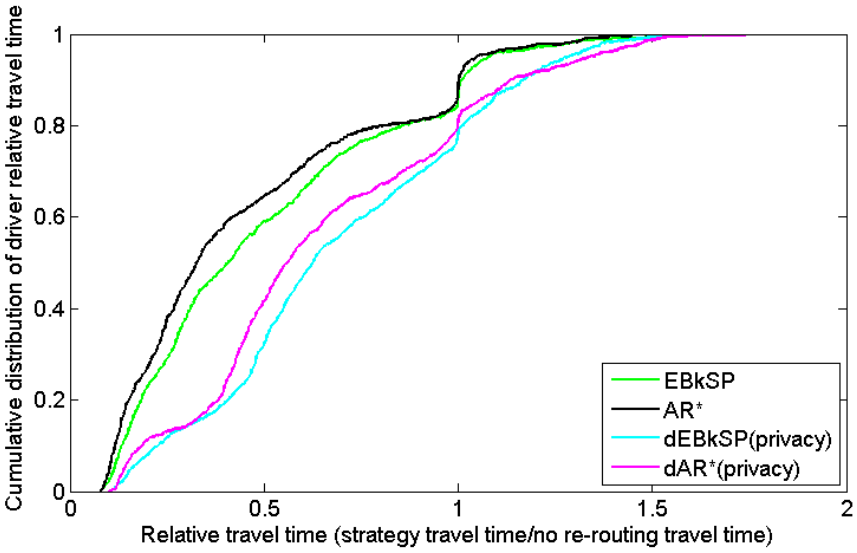


Figure 7.2 CDF of relative travel time (T_{max}=0.2s, k=8).

travel time (RelT) is defined as the actual travel time divided by the travel time without re-routing. It measures the travel time gains or losses for individual drivers. Figure 7.5 presents the CDF of RelT. It is observed that the system manages to improve the travel time for a large majority of drivers. However, there are a few drivers who experience increased travel time. This increase is limited to less than 50% for most of these drivers. From the figure, it is noticed that dAR* produces better results than dEBkSP (both include privacy-aware reporting). Compared to the centralized versions, both have just slightly worse results.

The explanation for the increase is that a system-wide optimization (i.e., reduce the average travel time) is focused in this dissertation, not user equilibrium which is known to be computationally expensive and difficult to achieve in the presence of congestion. From a practical point of view, a few bad experiences could impact the adoption rate. Therefore, investigating methods to bound this increase to low values is planned as future work.

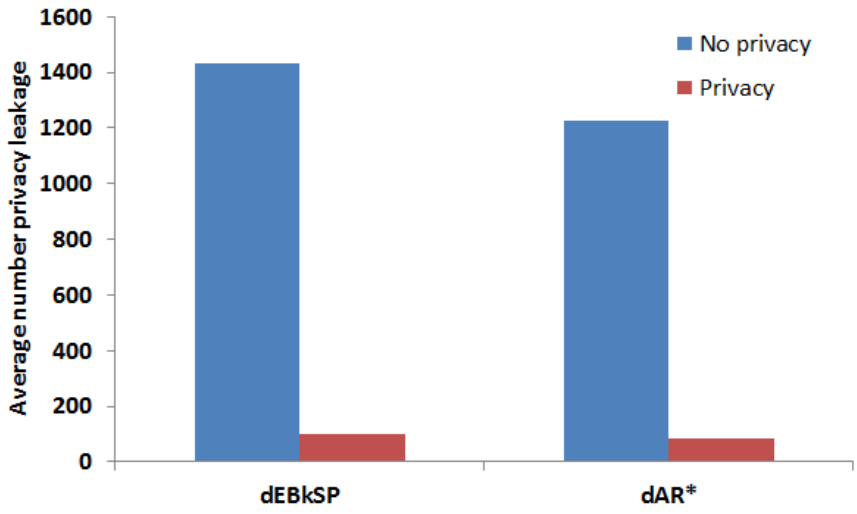


Figure 7.3 Privacy leakage in dEBkSP and dAR*.

7.2.3 Average Privacy Leakage

The formula described in Section 6.3.1 to quantify the privacy leakage as shown in Figure 7.3. In particular, the server can collect the location reports and output the trace data based on which the popularity of each road segment can be computed. From there, the importance of each vehicle report can be calculated. The sum of all the reports for one single vehicle is its privacy leakage as shown in equation 6.1. The average privacy leakage is defined in equation 6.2. These results demonstrate one of the major advantages of the hybrid system: it reduces the privacy leakage by up to 92% for both dEBkSP and dAR*. This is due to the fact that privacy-aware reporting avoids submitting location reports from highly sensitive low density roads and submits reports with low per-vehicle frequency in high density roads. Therefore, drivers are less prone to be identified by the un-trusted server and their location reports are difficult to be linked to each other; thus, driver location privacy is protected. On average, dAR* has less privacy leakage than dEBkSP because it has lower travel time: in dAR*, each vehicle finishes the journey faster, and thus there are fewer location reports.

7.2.4 Average Number of Re-routings

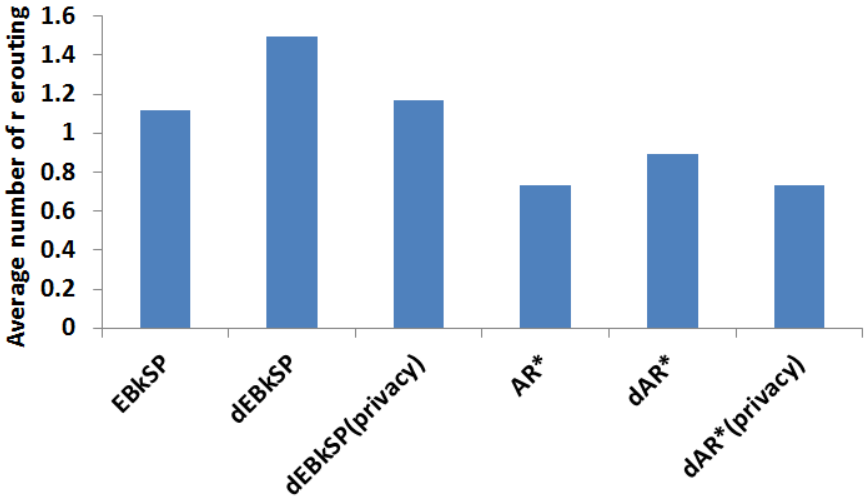


Figure 7.4 Average number of re-routing (T_{max}=0.2s, k=8).

From the system point of view, having a low number of re-routings means decreasing user distraction. Figure 7.4 shows the number of average re-routings compared to the centralized counterparts is dEBkSP > EBkSP(privacy)> EBkSP > dAR* > dAR*(privacy) > AR*. Intuitively, the centralized EBkSP and AR* have the least number of re-routings due to the global knowledge of the all vehicles' paths. Without privacy enhancement, dEBkSP and dAR* produce 34% and 22% more re-routings than the centralized ones. However, the good news is that privacy enhancement only results in 4.5% and 0.4% more re-routings. The reason is that, with privacy-aware reporting, one vehicle on each selected road segments is responsible for relaying the road map data to other vehicles. Certain vehicles may not be able to receive the map data due to the the network conditions such as packet losses. Thus, they are not re-routed.

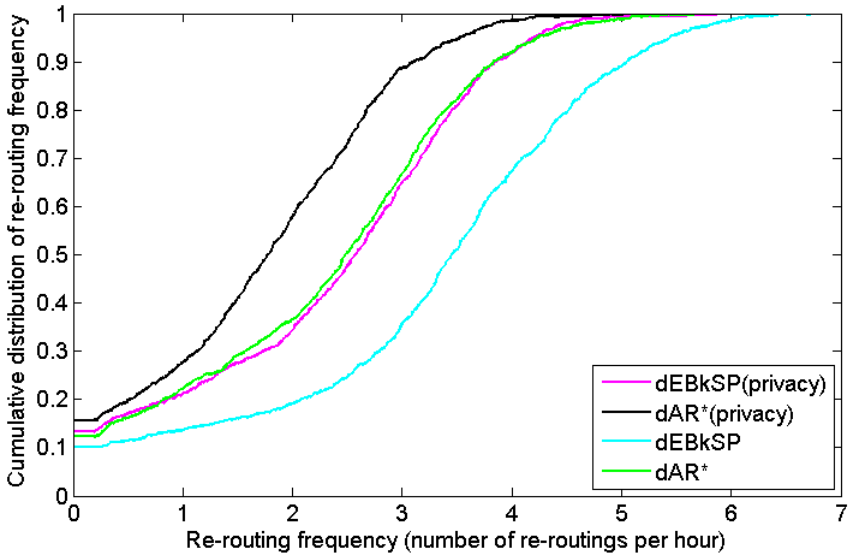


Figure 7.5 Re-routing frequency.

7.2.5 Distribution of Re-routing Frequency

As illustrated in figure 7.5, dAR* has less user distraction than dEBkSP. A large majority of drivers experience no more than 3 re-routings per hour, which is believed to be acceptable in city scenarios with heavy traffic.

Additionally, the strategies with privacy have fewer re-routings per hour than the ones without privacy. Specifically, for dEBkSP and dAR*, on average, the privacy-aware strategies produces 14% and 18% fewer number of re-routings. The reason is that not all the vehicles can receive re-routing notifications due to the network conditions. In the case of re-routings, this could be beneficial since it creates less distraction for each driver.

7.2.6 Computation Cost

In the hybrid system, the bottleneck of computing all the alternative paths is removed at the server. If computation time is high in the centralized system, the drivers would be informed too late about alternative paths, thus defeating the purpose of the system. In the hybrid system, each car performs its own path computation using information

received from neighboring vehicles. Therefore, this system is expected to have higher scalability. To prove this hypothesis, experiments are performed on a smart phone (which would be used as computing platform in the vehicles) and the obtained results are plugged into analytical formulas for the two distributed strategies.

The total time consumed for dAR* is the sum of the communication time among vehicles to collect information and the actual local computation time at a vehicle. Since the number of received trip data is a function of T_{max}, the communication time, the total time consumed for dAR* is $T^{total}_{dAR^*} = T_{max} + f(T_{max}) * C(AR^*)$, where $f(T_{max})$ is the number of received re-routing data items and $C(AR^*)$ is the cost of computation to perform AR* on one (origin, destination) pair.

For dEBkSP, $T^{total}_{dEBkSP} = C(k - paths) + T_{max} + f(T_{max}) * C(EBkSP))$, where $C(k - path)$ is the computation cost for k loopless shortest paths for one (origin, destination) pair and $C(EBkSP)$ is the cost to select one path from k paths based on the EBkSP algorithm. The complexity of EBkSP algorithm only depends on k and the average path length which can be considered as negligible. Therefore, dEBkSP has higher computational efficiency than dAR* since the computation time of dAR* is influenced heavily by the number of received re-routing data items whereas the computation time of dEBkSP is basically only $C(k - path)$.

In order to evaluate the time taken by these strategies on existing mobile platforms, a C++ Android application is developed on Samsung Galaxy SGH-T959V using Android NDK. The average computation time is measured on the initial (origin, destination) pairs for the 1000 vehicles in these experiments. Table 7.3 shows the computation cost of a single (origin, destination) pair for each algorithm. The maximum number of received trip data items when T_{max}=0.2s for dAR* is about

Table 7.3 Average Computation Time for One Pair of Origin and Destination

dijkstra	k shortest paths k=8	AR*
0.244s	0.386s	0.14s

82. Thus, the estimated computation time in worst case for dEBkSP and dAR* is 0.2+82*0.14=12s. The estimated computation time for dEBkSP is 0.386+0.2=0.6s. While both numbers demonstrate that the hybrid solution works well in practice for this scenario, it is clear that dEBkSP scales better. In larger regions with many vehicles, dAR* may not be able to meet the real time constraints.

7.2.7 Impact of K Path Compression

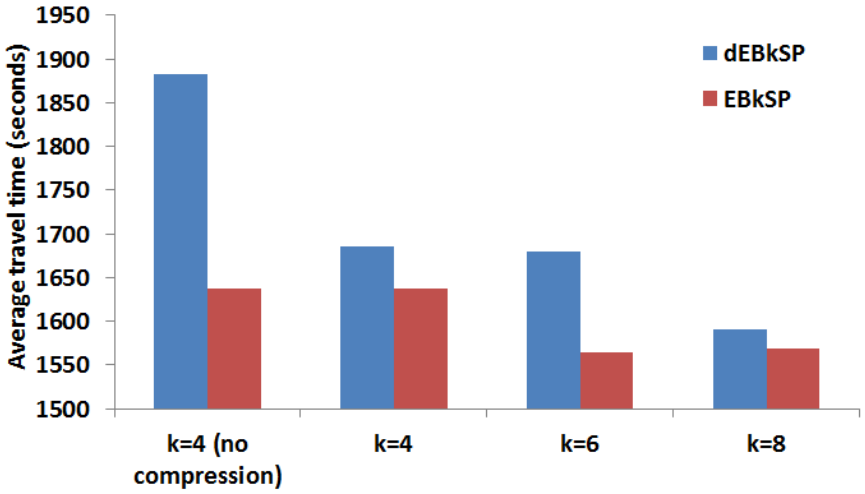


Figure 7.6 The average travel time with and without compression.

During the implementation and testing phase, it was noticed that the prioritized broadcast and the distance-based timer approach are essential in the proposed system because without them very little re-routing information is exchanged among the vehicles due to contention in congested regions. Among the remaining two optimizations, i.e., K path compression and XOR coding, the path compression brings the most benefits. Since, it is established that dEBkSP has higher scalability, figure 7.6 shows the benefits of compression on this strategy.

It is observed that compression improves the average travel time by 12% for k=4. This is due to the fact that compression reduces the packet size and improves the number of re-routing data items each vehicle can gather in VANET. When k increases,

dEBkSP continues to lowers the travel time. Due to the compression, when k turns from 4 to 8, the addition to the packet size remains small. Therefore, dEBkSP is able to achieve very similar performance as the centralized version.

It is noticed that only dEBkSP can take advantage of larger k values. The centralized version cannot: a larger k allows for better traffic balancing but introduces higher computational complexity since the centralized server needs to compute k paths for all the selected vehicles. This is not a problem for dEBkSP which distributes the path computation to individual vehicles. Therefore, dEBkSP can result in higher performance than EBkSP when higher k values are used.

7.2.8 Effectiveness of XOR Coding

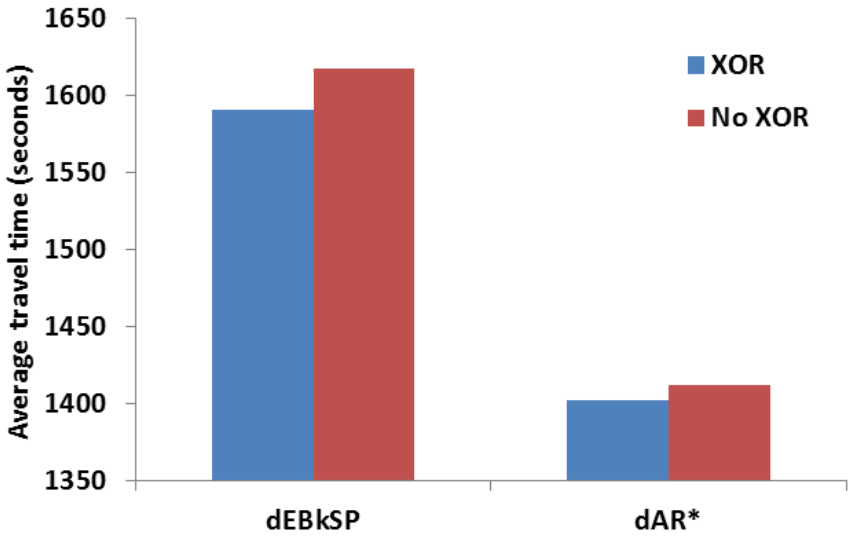


Figure 7.7 The average travel time with and without XOR coding.

The evaluation of XOR coding is illustrated in figure 7.7. With XOR coding, the average time is slightly decreased by 1.7% and 0.6%. This is due to the fact that XOR coding can improve the network throughput so that each vehicle is able to receive more routing information from the nearby vehicles. The more knowledge each vehicle can receive, the better decisions this vehicle can make. This analysis is confirmed

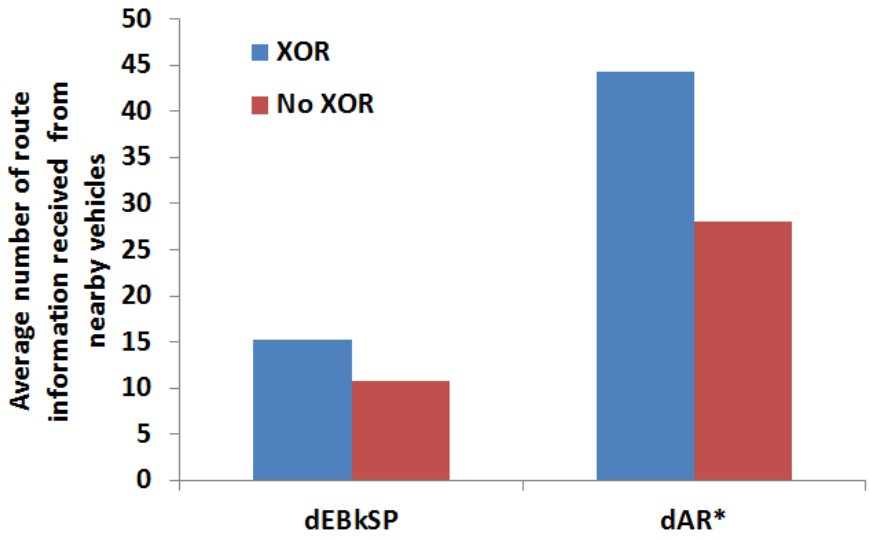


Figure 7.8 The average number of route information received from other vehicles.

by figure 7.8, which demonstrates that XOR coding increases the average number of received route information by 41% and 57% for dEBkSP and dAR*. Meanwhile, XOR coding improves the more throughput on dAR* than dEBkSP. This is because dEBkSP has much larger packet size and XOR coding creates much more burden for dEBkSP than dAR* due to the extra appended field.

7.2.9 Impact of Packet Size on Network Throughput

Figure 7.9 shows the impact of packet size on network throughput. dAR* has the smallest size of packet and dEBkSP packet size increases with k increases. Therefore, EBkSP (k=8) received the least trip data from surrounding vehicles. XOR coding approach improves network throughput by approximately 40%.

It is noticed that, although k increase for dEBkSP, the decrease of the received trip data is not as significant. It's due to the k-path compression algorithm applied in this scheme. Since k path compression only stores the the difference between the current path and previous path, even k increases, the packet size does not increase dramatically.

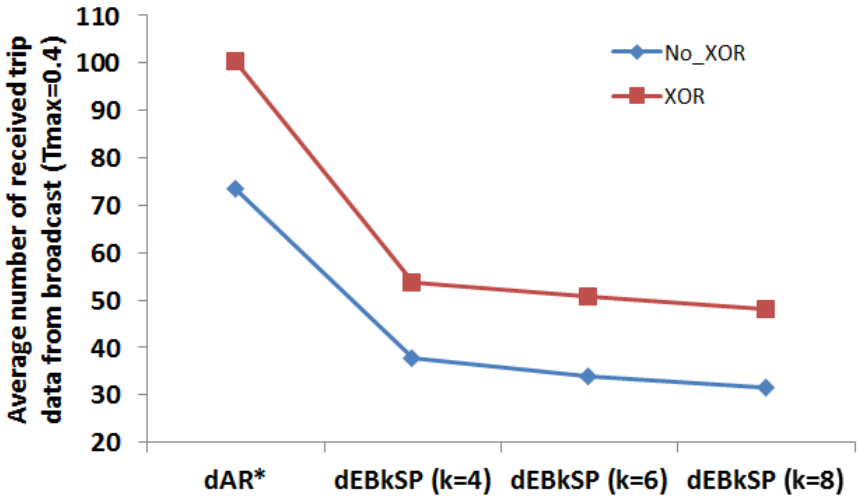
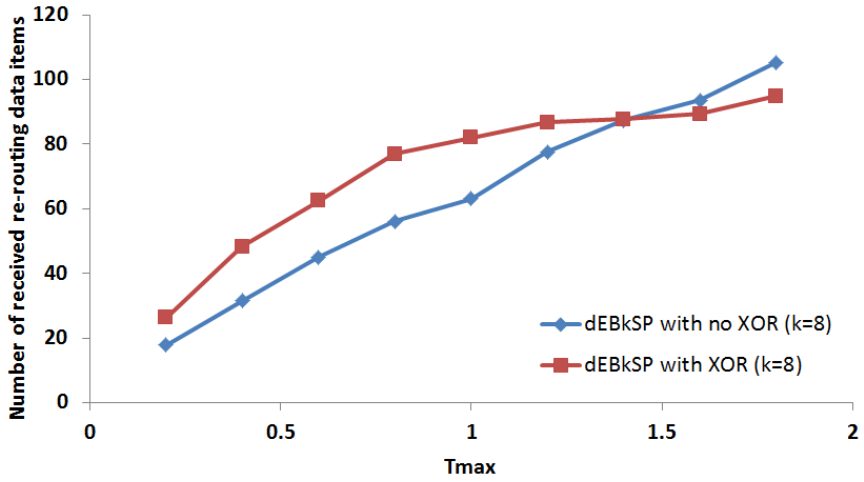


Figure 7.9 Average number of trip data gathered in the first re-routing period.

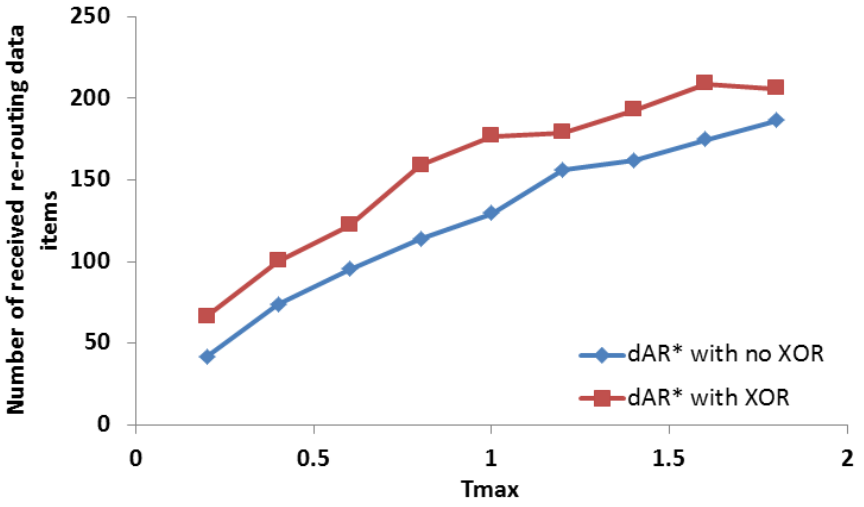
7.2.10 Impact of Broadcast Timeout Parameter

The parameter for broadcast timeout T_{max} is evaluated by varying the value from 0.2s to 1.8s as seen in figure 7.10 (a) and (b). The number of received re-routing data items from the surrounding vehicles increases as T_{max} increases. This is due to the prioritized broadcast: the larger T_{max} is, the larger the broadcast time interval is among all the vehicles. For example, if vehicle A broadcasts at T_0 while vehicle B starts at T_1, then the interval is T_1-T_0. The larger the interval, the smaller chance of contention and collision, thus the fewer packet losses.

The second observation is that when T_{max} is large enough (e.g., 1.4s), the XOR technique starts to show less efficiency in dEBkSP. Yet, for dAR*, the XOR still constantly performs better. It is due to the size difference of the packets in dEBkSP and dAR*. For dEBkSP, the packet size is already large. Adding the extra XOR field doubles the packet size and is not worthwhile to make up the packet loss. With regards to dAR*, due to the tiny packet size (e.g., OD pair), even if the size is doubled, the packet is still small. Therefore, the extra burden can be considered as negligible.



(a) dEBkSP



(b) dAR*

Figure 7.10 Number of received re-routing data items.

Table 7.4 Comparison of all the Distributed Re-routing Strategies

Effectiveness	dAR* > dAR*(privacy) > dEBkSP > dEBkSP (privacy)
Rerouting frequency	dAR*(privacy)>dAR* >dEBkSP(privacy)>dEBkSP
Computational cost	dEBkSP>dAR*; dEBkSP(privacy)>dAR*(privacy)

7.3 Chapter Summary

In this chapter, an extensive evaluation of the proposed distributed re-routing strategies is performed. The objectives of the experiments are threefold. First, the effectiveness of the proposed methods is measured. The experimental results indicate that the hybrid system can achieve similar performance as the centralized counterpart in terms of average travel time and average number of re-routings. Additionally, the computation is off-loaded to VANETs, which makes the hybrid system more appropriate for use in real-life applications. Moreover, with privacy-aware reporting, the privacy leakage is reduced up to 90%.

To summarize the performance of the distributed re-routing strategies, they are ranked according to three criteria in Table 7.4, where $A > B$ means that strategy A is better than strategy B. While each method has its own advantages and shortcomings, the experimental results made us conclude that the entropy method (EBkSP) should be the preferred strategy, as it offers the best trade-off between performance and computation cost.

CHAPTER 8

CONCLUSION

Heartened by the ubiquitousness of mobile devices such as smart phones or vehicle on-board units, this dissertation envisions a novel approach to tackle the ever more stringent problem of traffic congestion. This approach is based on a traffic guidance system that monitors traffic and proactively pushes individually-tailored re-routing guidance to vehicles when there are signs of congestion. The system is responsible for several functions such as traffic data representation, congestion prediction, and selection of the vehicles to be re-routed. Five re-routing algorithms are proposed to compute alternative routes for the re-routed vehicles. Then, an extensive set of simulation-based experiments are conducted to validate these approaches. The results show that the proposed re-routing algorithms are very effective in mitigating congestion and adapt well to the dynamic nature of the traffic, being also more efficient and scalable than existing approaches. In addition, the proposed traffic guidance system remains useful even with low compliance rate and moderate penetration rate. The experiments also demonstrate how the performance can be tuned by varying parameters such as re-routing method, re-routing period, number of alternative paths, and density threshold.

To improve scalability and privacy, a hybrid system is proposed which takes advantage of both 3/4G network (Internet communication) and VANET communication. Each vehicle reports privacy-aware location data to a central server through the 3/4G network. Once congestion is detected, the server pushes the global traffic view to vehicles who have reported recently and are close enough to the congestion spots. Then, individually tailored paths are computed distributively in VANET. The central server acts as the a coordinator and is responsible for constructing the global view

of the travel time. Each vehicle utilizes VANETs to exchange and share re-routing decisions to make best re-routing choices. Extensive experiments show that the hybrid distributed solutions can provide similar effective travel time compared to the centralized counterparts. Meanwhile, since the computation is off-loaded to VANETs, the system is much more scalable.

The results of this dissertation demonstrate that a practical, cost-effective, and effective traffic re-routing system can be implemented and deployed in real-life settings. Such a system will improve the daily life of all of us by mitigating the effects of traffic congestions.

BIBLIOGRAPHY

[1] http://www.google.com/mobile/. [Online; accessed on 14-Dec-2013].

[2] http://www.tomtom.com/en_gb/products/your-drive/smartphone-navigation/iphone/navigation-app-iphone-ipad/. [Online; accessed on 14-Dec-2013].

[3] http://www.inrix.com. [Online; accessed on 14-Dec-2013].

[4] http://www.autobahn.nrw.de. [Online; accessed on 14-Dec-2013].

[5] http://sumo.sourceforge.net/doc/current/docs/userdoc/Tools/Assign.html. [Online; accessed on 14-Dec-2013].

[6] http://veins.car2x.org/. [Online; accessed on 14-Dec-2013].

[7] http://www.omnetpp.org/. [Online; accessed on 14-Dec-2013].

[8] Car-to-car communication. http://www.bmw.com/com/en/insights/technology/technology_guide/articles/cartocar_communication.html, 2012. [Online; accessed on 14-Dec-2013].

[9] M. Artimy. Local density estimation and dynamic transmission-range assignment in vehicular ad hoc networks. *IEEE Transactions on Intelligent Transportation Systems*, 8(3):400–412, 2007.

[10] J.H. Banks. *Introduction to transportation engineering*. McGraw-Hill: New York, 2002.

[11] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz. Sumo - simulation of urban mobility: An overview. In *Proceedings of the 3rd International Conference on Advances in System Simulation (SIMUL 2011)*, Barcelona, Spain, 2011.

[12] M. Behrisch, D. Krajzewicz, and Y.P. Wang. Comparing performance and quality of traffic assignment techniques for microscopic road traffic simulations. In *Proceedings of international symposium on dynamic traffic assignment (DTA 2008)*, Leuven, Belgium, 2008.

[13] S. Biswas, R. Tatchikou, and F. Dion. Vehicle-to-vehicle wireless communication protocols for enhancing highway traffic safety. *IEEE Communications Magazine*, 44(1):74–82, 2006.

[14] D. Boston, S. Mardenfeld, J.S. Pan, Q. Jones, A. Iamnitchi, and C. Borcea. Leveraging bluetooth co-location traces in group discovery algorithms. *Pervasive and Mobile Computing*, 2012.

[15] A. Broder and M. Mitzenmacher. Network applications of bloom filters: A survey. *Internet Mathematics*, 1(4):485–509, 2004.

[16] Y.C. Chiu, J. Bottom, M. Mahut, A. Paz, R. Balakrishna, T. Waller, and J. Hicks. Dynamic traffic assignment: A primer. *Transportation Research E-Circular*, (E-C153), 2011.

[17] B.A. Coifman and R. Mallika. Distributed surveillance on freeways emphasizing incident detection and verification. *Transportation Research Part A: Policy and Practice*, 41(8):750–767, 2007.

[18] J. Cranshaw, E. Toch, J. Hong, A. Kittur, and N. Sadeh. Bridging the gap between physical location and online social networks. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pages 119–128, 2010.

[19] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004.

[20] S. Dornbush and A. Joshi. Streetsmart traffic: Discovering and disseminating automobile congestion using vanets. In *Vehicular Technology Conference, IEEE 65th*, pages 11–15, 2007.

[21] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan. The pothole patrol: using a mobile sensor network for road surface monitoring. In *Proceedings of the 6th international conference on Mobile systems, applications, and services (MobiSys 2008)*, pages 29–39. ACM, 2008.

[22] C. Fragouli, J. Le Boudec, and J. Widmer. Network coding: an instant primer. *ACM SIGCOMM Computer Communication Review*, 36(1):63–68, 2006.

[23] Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, July 1987.

[24] T.L. Friesz, D. Bernstein, T.E. Smith, R.L. Tobin, and BW Wie. A variational inequality formulation of the dynamic network user equilibrium problem. *Operations Research*, pages 179–191, 1993.

[25] C. Gawron. *Simulation-based traffic assignment–computing user equilibria in large street networks*. PhD thesis, University of Cologne, Germany, 1999.

[26] B. Gedik and L. Liu. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *IEEE Transactions on Mobile Computing*, 7(1):1–18, 2008.

[27] B. George, S. Kim, and S. Shekhar. Spatio-temporal network databases and routing algorithms: A summary of results. *Advances in Spatial and Temporal Databases*, pages 460–477, 2007.

[28] V. Gradinescu, C. Gorgorin, R. Diaconescu, V. Cristea, and L. Iftode. Adaptive traffic lights using car-to-car communication. In *Vehicular Technology Conference, IEEE 65th*, pages 21–25, 2007.

[29] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st international conference on mobile systems, applications and services*, pages 31–42. ACM, 2003.

[30] M. Gruteser and B. Hoh. On the anonymity of periodic location samples. In *Security in Pervasive Computing*, pages 179–192. Springer, 2005.

[31] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

[32] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[33] J.C. Herrera, D.B. Work, R. Herring, X.J. Ban, Q. Jacobson, and A.M. Bayen. Evaluation of traffic data obtained via gps-enabled mobile phones: The mobile century field experiment. *Transportation Research Part C: Emerging Technologies*, 18(4):568–583, 2010.

[34] B. Hoh and M. Gruteser. Protecting location privacy through path confusion. In *1st International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm 2005)*, pages 194–205. IEEE, 2005.

[35] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J.C. Herrera, A.M. Bayen, M. Annavaram, and Q. Jacobson. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pages 15–28. ACM, 2008.

[36] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady. Preserving privacy in gps traces via uncertainty-aware path cloaking. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 161–171, 2007.

[37] H.H. Hoos and T. Stützle. *Stochastic local search: Foundations and applications*. Access Online via Elsevier, 2005.

[38] E. Horvitz, J. Apacible, R. Sarin, and L. Liao. Prediction, expectation, and surprise: Methods, designs, and study of a deployed traffic forecasting service. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, pages 244–257, 2005.

[39] T. Hunter, R. Herring, P. Abbeel, and A. Bayen. Path and travel time inference from gps probe vehicle data. *NIPS Workshop on Analyzing Networks and Learning with Graphs*, 2009.

[40] D. Jiang and L. Delgrossi. Ieee 802.11 p: Towards an international standard for wireless access in vehicular environments. In *IEEE Vehicular Technology Conference*, pages 2036–2040, 2008.

[41] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. Xors in the air: practical wireless network coding. 36(4):243–254, 2006.

[42] B.S. Kerner. Optimum principle for a vehicular traffic network: minimum probability of congestion. *Journal of Physics A: Mathematical and Theoretical*, 44:092001, 2011.

[43] J. Kleinberg and E. Tardos. *Algorithm design*. Pearson Education India: Delhi, 2006.

[44] S. Krauss, P. Wagner, and C. Gawron. Metastable states in a microscopic model of traffic flow. *Physical Review E*, 55(5):5597, 1997.

[45] S. Kumar, L. Shi, N. Ahmed, S. Gil, D. Katabi, and D. Rus. Carspeak: a content-centric network for autonomous driving. In *Proceedings of the ACM conference on Applications, technologies, architectures, and protocols for computer communication (SIGCOMM 2012)*, pages 259–270, 2012.

[46] Wei L. and Jie W. On reducing broadcast redundancy in ad hoc wireless networks. *IEEE Transactions on Mobile Computing*, 1(2):111–122, 2002.

[47] S. Lämmer and D. Helbing. Self-stabilizing decentralized signal control of realistic, saturated network traffic. Technical report, Santa Fe Institute, Santa Fe, NM, 2010.

[48] E.L. Lawler. A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. *Management Science*, pages 401–405, 1972.

[49] S. Lim. *Congestion-aware traffic routing for large-scale mobile agent systems*. PhD thesis, Massachusetts Institute of Technology, 2012.

[50] C. Lochert, B. Scheuermann, C. Wewetzer, A. Luebke, and M. Mauve. Data aggregation and roadside unit placement for a VANET traffic information system. In *Proceedings of the fifth ACM international workshop on VehiculAr Inter-NETworking*, pages 58–65, 2008.

[51] S. Maerivoet. *Modelling Traffic on Motorways: State-of-the-Art, Numerical Data Analysis, and Dynamic Traffic Assignment*. PhD thesis, Katholieke Universiteit Leuven, 2006.

[52] H.S. Mahmassani, T-Y. Hu, and R. Jayakrishnan. Dynamic traffic assignment and simulation for advanced network informatics (dynasmart). In *Proceedings of the 2nd International CAPRI Seminar on Urban Traffic Networks*, Capri, Italy, 1992.

[53] N. Malviya, S. Madden, and A. Bhattacharya. A continuous query system for dynamic route planning. In *Proceedings of 27th IEEE International Conference on Data Engineering (ICDE 2011)*, pages 792–803, 2011.

[54] H.C. Manual. *Highway capacity manual*. Washington, DC, 2000.

[55] J. Manweiler, R. Scudellari, and L.P. Cox. Smile: Encounter-based trust for mobile social services. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 246–255, 2009.

[56] E.Q.V. Martins and M.M.B. Pascoal. A new implementation of yen's ranking loopless paths algorithm. *4OR: A Quarterly Journal of Operations Research*, 1(2):121–133, 2003.

[57] D.K. Merchant and G.L. Nemhauser. A model and an algorithm for the dynamic traffic assignment problems. *Transportation Science*, 12(3):183–199, 1978.

[58] D.K. Merchant and G.L. Nemhauser. Optimality conditions for a dynamic traffic assignment model. *Transportation Science*, 12(3):200–207, 1978.

[59] J. Meyerowitz and R. Roy Choudhury. Hiding stars with fireworks: location privacy through camouflage. In *Proceedings of the 15th annual international conference on Mobile computing and networking*, pages 345–356. ACM, 2009.

[60] P. Mittal and N. Borisov. Information leaks in structured peer-to-peer anonymous communication systems. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 267–278, 2008.

[61] P. Mohan, V.N. Padmanabhan, and R. Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 323–336, 2008.

[62] T. Nadeem, S. Dashtinezhad, C. Liao, and L. Iftode. Trafficview: traffic data dissemination using car-to-car communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 8(3):6–19, 2004.

[63] S.Y. Ni, Y.C. Tseng, Y.S. Chen, and J.P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 151–162, 1999.

[64] J. Nzouonta, N. Rajgure, G. Wang, and C. Borcea. VANET routing on city roads using real-time vehicular traffic information. *Vehicular Technology, IEEE Transactions on*, 58(7):3609–3626, 2009.

[65] L.G. Papaleondiou and M.D. Dikaiakos. Trafficmodeler: A graphical tool for programming microscopic traffic simulators through high-level abstractions. In *Vehicular Technology Conference, IEEE 69th*, pages 1–5, 2009.

[66] S. Peeta and A.K. Ziliaskopoulos. Foundations of dynamic traffic assignment: The past, the present and the future. *Networks and Spatial Economics*, 1(3):233–265, 2001.

[67] A. Perko. Implementation of algorithms for k shortest loopless paths. *Networks*, 16(2):149–160, 1986.

[68] R.A. Popa, A.J. Blumberg, H. Balakrishnan, and F.H. Li. Privacy and accountability for location-based aggregate statistics. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 653–666, 2011.

[69] H. Prothmann, H. Schmeck, S. Tomforde, J. Lyda, J. Hahner, C. Muller-Schloer, and J. Branke. Decentralized route guidance in organic traffic control. In *Proceedings of the 5th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2011)*, pages 219–220, 2011.

[70] O. Riva, T. Nadeem, C. Borcea, and L. Iftode. Context-aware migratory services in ad hoc networks. *IEEE Transactions on Mobile Computing*, 6(12):1313–1328, 2007.

[71] J. Rybicki, B. Scheuermann, W. Kiess, C. Lochert, P. Fallahi, and M. Mauve. Challenge: peers on wheels-a road to new traffic information systems. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 215–221, 2007.

[72] J. Rybicki, B. Scheuermann, M. Koegel, and M. Mauve. Peertis: a peer-to-peer traffic information system. In *Proceedings of the sixth ACM international workshop on VehiculAr InterNETworking*, pages 23–32, 2009.

[73] M. Santo. Toyota, audi promise driverless car demos at ces 2013. `http://www.examiner.com/article/toyota-audi-promise-driverless-car-demos-at-ces-2013`, 2013. [Online; accessed on 14-Dec-2013].

[74] D. Schrank, T. Lomax, and S. Turner. *TTI's Urban Mobility Report*. Texas Transportation Institute, Texas A & M University, 2011.

[75] D. Schultes. Route planning in road networks. *Karlsruhe: Universität Karlsruhe (TH). Fakultät für Informatik. Institut für Theoretische Informatik, Algorithmik II*, 2008.

[76] S. Senge and H. Wedde. Bee inspired online vehicle routing in large traffic systems. In *Proceedings of the 2nd International Conference on Adaptive and Self-Adaptive Systems and Applications (ADAPTIVE 2010)*, pages 78–83, 2010.

[77] P. Shankar, V. Ganapathy, and L. Iftode. Privately querying location-based services with sybilquery. In *Proceedings of the 11th international conference on Ubiquitous computing*, pages 31–40. ACM, 2009.

[78] C.E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[79] K. Suriyapaibonwattana and C. Pomavalai. An effective safety alert broadcast algorithm for VANET. In *Proceedings of International Symposium on Communications and Information Technologies*, pages 247–250. IEEE, 2008.

[80] B. Tatomir, S. Fitrianie, M. Paltanea, and L. Rothkrantz. Dynamic routing in traffic networks and manets using ant based algorithms. In *Proceedings of the 7th International Conference on Artificial Evolution, Lille, France*, October 2005.

[81] N.B Taylor. *CONTRAM 5, an enhanced traffic assignment model*. TRRL research report. Transport and Road Research Laboratory, Crowthorne, United Kingdom, 1990.

[82] M. Torrent-Moreno, J. Mittag, P. Santi, and H. Hartenstein. Vehicle-to-vehicle communication: fair transmit power control for safety-critical information. *IEEE Transactions on Vehicular Technology*, 58(7):3684–3703, 2009.

[83] V. Vijayenthiran. Ford powers ahead with development of car-to-car communication technology. `http://www.motorauthority.com/news`, 2012. [Online; accessed on 14-Dec-2013].

[84] V. Vijayenthiran. GM participating in U.S. car to car communications trial. `http://www.motorauthority.com/news`, 2012. [Online; accessed on 14-Dec-2013].

[85] J.G. Wardrop. Some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers, Part II*, 1(36):252–378, 1952.

[86] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J.P. Hubaux. Traci: an interface for coupling road traffic and network simulators. In *Proceedings of the 11th communications and networking simulation symposium*, pages 155–163. ACM, 2008.

[87] Y. Wen. *Scalability of dynamic traffic assignment*. PhD thesis, Massachusetts Institute of Technology, 2008.

[88] K. Wessel, M. Swigulski, A. Köpke, and D. Willkomm. Mixim: the physical layer an architecture overview. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, page 78. Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, 2009.

[89] N.P. Wisitpongphan, OK Tonguz, JS Parikh, P. Mudalige, F. Bai, and V. Sadekar. Broadcast storm mitigation techniques in vehicular ad hoc networks. *Wireless Communications, IEEE*, 14(6):84–94, 2007.

[90] D.B. Work, O.P. Tossavainen, S. Blandin, A.M. Bayen, T. Iwuchukwu, and K. Tracton. An ensemble kalman filtering approach to highway traffic estimation using gps enabled mobile devices. In *Proceedings of the 47th IEEE Conference on Decision and Control*, pages 5062–5068, 2008.

[91] T. Xu and Y. Cai. Exploring historical location data for anonymity preservation in location-based services. In *The 27th Conference on Computer Communications (INFOCOM 2008)*, pages 547–555. IEEE, 2008.

[92] T. Xu and Y. Cai. Feeling-based location privacy protection for location-based services. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 348–357, 2009.

[93] S. Yu, F. Ye, H. Wang, S. Mabu, K. Shimada, S. Yu, and K. Hirasawa. A global routing strategy in dynamic traffic environments with a combination of q value-based dynamic programming and boltzmann distribution. In *International conference on Instrumentation, Control, Information Technology and System Integration*, pages 623–627. IEEE, 2008.

[94] P. Zhou, T. Nadeem, P. Kang, C. Borcea, and L. Iftode. Ezcab: A cab booking application using short-range wireless communication. In *Third IEEE International Conference on pervasive Computing and Communications (PerCom 2005)*, pages 27–38. IEEE, 2005.