Reducing Non-Recurrent Urban Traffic Congestion using Vehicle Re-routing

By

Shen Wang

M.Eng., B.Eng.

A Dissertation submitted in fulfilment of the requirements for the award of Doctor of Philosophy (Ph.D.)
to the

Dublin City University
School of Electronic Engineering

Supervisors: Dr. Jennifer McManis and
Dr. Soufiene Djahel (Manchester Metropolitan University)

September, 2016

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Ph.D. is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: _____

Student ID:_____11101385_____

Date: _____

Acknowledgement

It is an amazing journey for someone like me who was raised up by a family without even a local college degree in China, can eventually complete a PhD program in the "Silicon Valley of Europe". My greatest thank to my dearest parents who fully understand and always support me to do what I am interested in. Also to my beloved wife for the wonderful and stable marriage life we have built together since we met three years ago.

During the last four years, I feel grateful to the people I have been working with in Performance Engineering Laboratory (PEL). As my co-supervisor, in fact more like a friend, Dr. Soufiene Djahel is very professional and surprisingly patient to spend his time and energy on answering my questions ranging from "how to be a good researcher" to "where should I get Halal food". My sincere gratitude goes to Dr. Jennifer McManis, my principal supervisor, for her valuable and elaborate comments on each of my significant submissions, as well as her trust on me, a non-native English speaker, for the opportunity to give guest lectures on her module. My thank also goes to the weekly PEL group meeting, in which my PhD progress had been discussed critically for 15 times by many PEL members: Prof. Liam Murphy, Prof. John Murphy, Dr. Phillip Perry, Dr. Gabriel-Miro Muntean, Dr. Anthony Ventresque, Prof. Damien Magoni and so on. I've also set up friendships with my colleagues in PEL: like Yi Han and Imane since they helped me adapted to the new environment; like Sofiane since we had an unforgettable conference trip in 2014; also like Dr. Michal Vondra, who was a visiting PhD in PEL from Czech Republic, since the conversation with him that initiated the main contribution of my PhD.

I highly appreciate the industry experiences in IBM that Dublin City University and PEL offer me. Cormac McKenna in IBM Software Group encouraged me to think research problem from industry perspective. Takashi Imamichi in IBM Research helped me to improve my technical and management skills, and broaden my research vision. I am also thankful to Xiaowen, Jiayuan, Liping, Matthias, Alex, Luca, Renan, Pedro, and Stephan for exciting soccer games and travelling experiences we had in Ireland and Brazil.

Dublin, May 2016
Shen Wang

List of Publications

[Journal]

- **Shen Wang**, Soufiene Djahel, Zonghua Zhang and Jennifer McManis, "Next Road Rerouting: A Multi-Agent System for Mitigating Unexpected Urban Traffic Congestion", accepted at *IEEE Transactions on Intelligent Transportation Systems*, 2016.

[Conferences]

- **Shen Wang**, Soufiene Djahel, and Jennifer McManis, "An Adaptive and VANETs-based Next Road Rerouting System for Unexpected Urban Traffic Congestion Avoidance", *the 7th IEEE Vehicular Networking Conference (VNC)*, Kyoto, Japan, December, 2015.
- **Shen Wang**, Soufiene Djahel, and Jennifer McManis, "A Multi-Agent Based Vehicle Re-routing System for Unexpected Traffic Congestion Avoidance", *the 17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Qingdao, China, October, 2014.
- **Shen Wang**, Soufiene Djahel, Jennifer McManis, Cormac McKenna and Liam Murphy, "Comprehensive Performance Analysis and Comparison of Vehicle Routing Algorithms in Smart Cities", *the 5th Global Information Infrastructure Symposium(GIIS), IEEE,* Trento, Italy, October, 2013.
- **Shen Wang**, Soufiene Djahel, and Jennifer McManis, "A Hybrid Vehicular Re-routing Strategy with Dynamic Time Constraints for Road Traffic Congestion Avoidance", *the 12th Information Technology & Telecommunications (IT&T) Conference*, Athlone, Ireland, May, 2013

Table of Contents

List of Figures

List of Tables

List of Equations

List of Abbreviations

AD*: Anytime D*.

API: Application Programming Interface.

ARA*: Anytime Re-planning A*.

ATSC: Adaptive Traffic Signal Control

ATT: Average Travel Time.

BPR: Bureau of Public Road.

BSM: Basic Safety Message.

CAM: Cooperative Awareness Message.

CPU: Central Processing Unit.

CV: Coefficient of Variance.

DA: Dijkstra's Algorithm.

DIKB: DA using Buckets.

DIKF: DA using Fibonacci Heap.

DTA: Dynamic Traffic Assignment.

ERE: Scenario of En-Route Event.

ETSI: European Telecommunications Standards Institute.

FCD: Floating Car Data.

GC: Geographic Closeness of Congestion.

GD: Geographic Distance to Destination.

IBM: International Business Machines Cooperation.

ICT: Information and Communication Technology.

IEEE: Institute for Electrical and Electronics Engineers.

iTL: Intelligent Traffic Light.

ITS: Intelligent Transportation Systems.

MAS: Multi Agent System.

NRR: Next Road Rerouting.

O/D: Origin / Destination.

ORG: Scenario of Original Traffic.

PTI: Planning Time Index.

RO: Road Occupancy.

RSU: Road Side Unit.

SCATS: Sydney Coordinated Adaptive Traffic System.

SCOOT: Split Cycle Offset Optimization Technique.

SI: System Instability.

SO: System Optimum.

STA: Static Traffic Assignment.

STD: Standard Deviation.

SUMO: Simulation of Urban Mobility.

TAPAS: Travel and Activity PAtterns Simulation.

TCP: Transmission Control Protocol.

TMS: Traffic Management Systems.

TOC: Transportation Operation Centre.

TraCI: Traffic Control Interface.

TT: Travel Time.

TTI: Travel Time Index.

UDP: User Datagram Protocol.

UE: User Experience.

V2I: Vehicle to Infrastructure.

V2V: Vehicle to Vehicle.

VANET: Vehicular Ad-hoc Network.

VDF: Volume-Delay-Function.

VNS: Vehicle Navigation Systems.

WAVE: Wireless Access in Vehicular Environment.

Wi-Fi: Wireless Fidelity.

Abstract

Title: Reducing Non-Recurrent Urban Traffic Congestion using Vehicle Re-routing

PhD candidate: Shen Wang

Recently, with the trend of world-wide urbanization, some of the accompanying problems are getting serious, including road traffic congestion. To deal with this problem, city planners now resort to the application of the latest information and communications technologies. One example is the adaptive traffic signal control system (e.g. SCATS, SCOOT). To increase the throughput of each main intersection, it dynamically adjusts the traffic light phases according to real-time traffic conditions collected by widely deployed induction loops and sensors. Another typical application is the on-board vehicle navigation system. It can provide drivers with a personalized route according to their preferences (e.g. shortest/fastest/easiest), utilizing comprehensive geo-map data and floating car data. Dynamic traffic assignment is also one of the key proposed methodologies, as it not only benefits the individual driver, but can also provide a route assignment solution for all vehicles with guaranteed minimum average travel time.

However, the non-recurrent road traffic congestion problem is still not addressed properly. Unlike the recurrent traffic congestion, which is predictable by capturing the daily traffic pattern, unexpected road traffic congestion caused by unexpected en-route events (e.g. road maintenance, an unplanned parade, car crashes, etc.), often propagates to larger areas in very short time. Consequently, the congestion level of areas around the event location will be significantly degraded. Unfortunately, the three aforementioned methods cannot reduce this unexpected congestion in real time.

The contribution of this thesis firstly lies in emphasizing the importance of the dynamic time constraint for vehicle rerouting. Secondly, a framework for evaluating the performance of vehicle route planning algorithms is proposed along with a case study on the simulated scenario of Cologne city. Thirdly, based on the multi-agent architecture of SCATS, the next road rerouting (NRR) system is introduced. Each agent in NRR can use the locally available information to provide the most promising next road guidance in the face of the unexpected urban traffic congestion. In the last contribution of this thesis, further performance improvement of NRR is achieved by the provision of high-resolution, high update frequency traffic information using vehicular ad hoc networks. Moreover, NRR includes an adaptation mechanism to dynamically determine the algorithmic (i.e. factors in the heuristic routing cost function) and operational (i.e. group of agents which must be enabled) parameters.

The simulation results show that in the realistic urban scenario, compared to the existing solutions, NRR can significantly reduce the average travel time and improve the travel time reliability. The results also indicate that for both rerouted and non-rerouted vehicles, NRR does not bring any obvious unfairness issue where some vehicles overwhelmingly sacrifice their own travel time to obtain global benefits for other vehicles.

Chapter 1
Introduction

1.1. Research Motivation

Urbanization, a worldwide phenomenon describing a trend characterized by an increasing movement of the countryside's population into urban areas, has been ongoing for the last six decades. In the momentous year of 2007, city's populations exceeded for the first time rural area populations, as shown in Figure 1.1 from the 2014 United Nations' report [1]. This report also predicts that by the year 2050, there will be about two-thirds urbanized population in the world.

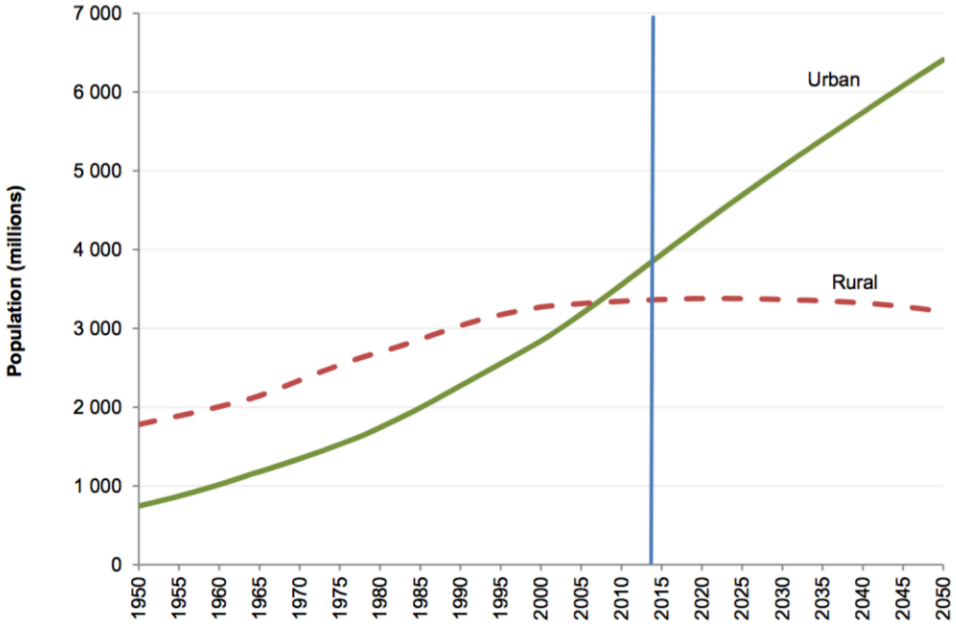


Figure 1.1: The world's urban and rural populations, 1950-2050 [1].

Urbanization has many positive impacts on human society. It creates an increasing number of better opportunities for jobs, education, and healthcare that more and more people are moving from the countryside to pursue. Urbanization makes the global distribution of population more concentrated in areas where less natural disasters occur, more food can be produced and more infrastructure can be built. Additionally, urbanization facilitates the requirement of modern industrialized society so that individuals cooperate with more people from diverse backgrounds. Consequently, the past 60 years of global urbanization have resulted in an enormous economic growth, and concentration of population in densely populated cities.

Urbanization leads to a series of unprecedented challenges including urban-rural inequality and environmental. Traffic congestion, one of the aforementioned new challenges, will be particularly studied in this thesis. A recent urban mobility report [2] states that in the year 2014 in U.S., the monetary loss due to traffic congestion is evaluated as $160 billion, representing 6.9 billion hours of extra travel time and 3.1 billion gallons of wasted fuel. This economic loss was only $42 billion back in 1982, and $114 billion in 2000. **Urban road traffic congestion is considered as the consequence of short supply in road capacity with respect to fast growing traffic demand.** The modification of road infrastructure is not as flexible as traffic demand; therefore, road traffic congestion often occurs during peak commuting hours when traffic can be built up by several times within only one hour, as indicated in Figure 1.2.

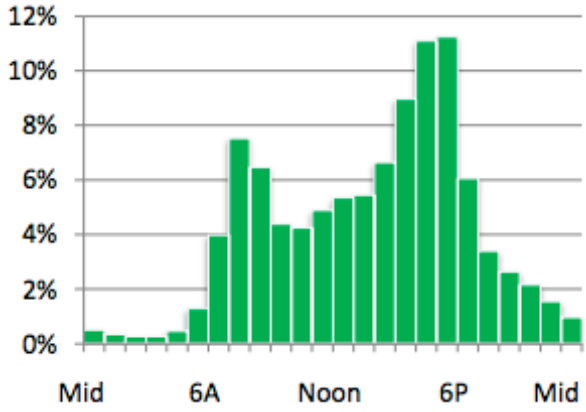


Figure 1.2: Percent of delay for hours of day [2].

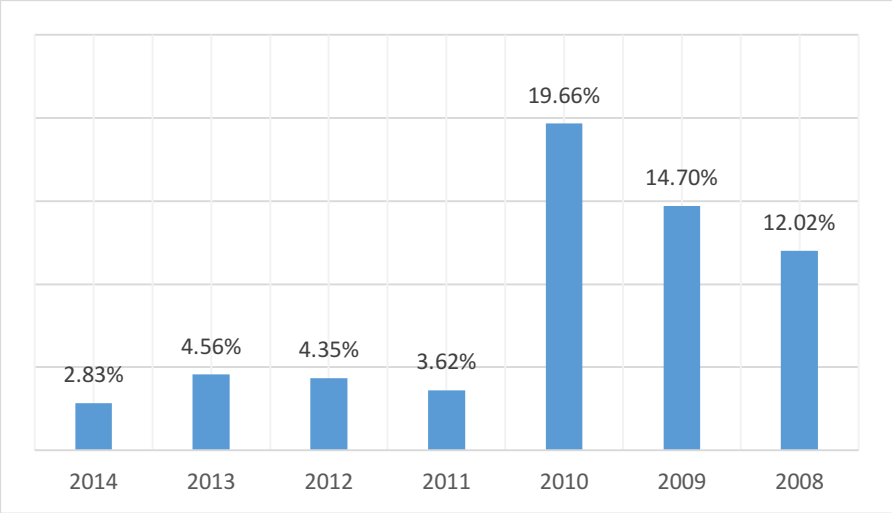


Figure 1.3: Growth rate of the number of vehicles in Beijing since 2008 [3].

One way to alleviate urban traffic congestion is the implementation of a public policy to restrict the growth of traffic demand. For instance, the local authority of Beijing has carried out two typical policies to control the total volume of vehicles on the roads. The first representative policy is called "End-number license plate policy". After a successful test during 2008 Beijing Olympic Games, all registered vehicles are classified into 5 groups by the last digit of their license plate numbers, 0 or 5, 1 or 6, 2 or 7, 3 or 8, and 4 or 9. Each group of vehicles is banned to be driven in the city center area during daytime of one of the 5 weekdays. For example, vehicles from the group "1 or 6" are not allowed to be on the road on Mondays from 7am to 8pm, those from the group "2 or 7" cannot appear in the central urban area on Tuesdays, and so on. This policy has resulted in a reduction of almost 40% of daily emission and nearly 20% of road traffic [4]. The other typical policy, "small passenger car purchase policy", is defined to limit the annual quota for newly registered vehicles since 2011. For example, according to the result of the latest lottery, for every 665 applicants for a new car registration, only one is permitted by random selection. As shown in Figure 1.3, this policy significantly decreases the growth rate of the number of vehicles in Beijing. Likewise, a more modest example is the application of road pricing policy in London. Instead of restricting the traffic demand and penalizing the violators, the local authority in

London charges drivers whose vehicles are running in the congested area during daytimes on weekdays. Although these policies are somewhat effective, due to the fact that the institution of legislation and law enforcement varies a lot in different countries, this type of solutions cannot be easily generalized to other countries to reduce the traffic congestion.

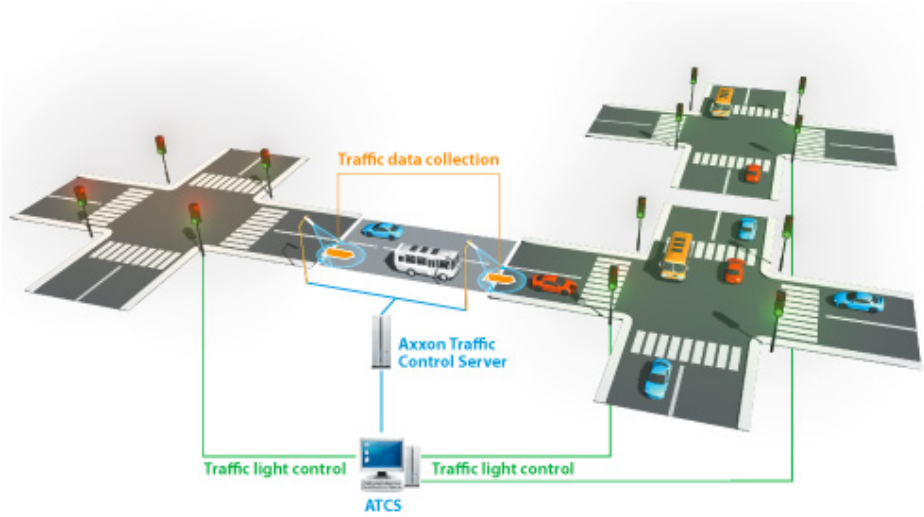


Figure 1.4: The schematic of adaptive traffic control systems[1]

Another way to relieve urban traffic congestion is to apply information and communication technology (ICT) to enhance the information exchange for all road transportation participants. For example, as shown in Figure 1.4, adaptive traffic signal control (ATSC), such as SCATS [45] and SCOOT [46], adjusts the offset, cycle and split of traffic signals to optimize the throughput of each main intersection. The basis of this automatic adjustment is the real-time traffic information collected during fixed time intervals by induction loops. The induction loops are generally installed under the ground of a certain location of major urban roads. They can detect the percentage of time that the detection area of the induction loop is occupied by vehicles. This system enhances the awareness of traffic conditions for the traffic operation center (TOC), thus improving the traffic light timings to control vehicles queuing in front of junctions. The second

[1] http://www.internetbillboards.net/wp-content/uploads/adaptive-traffic-control.jpg

ICT application for congestion avoidance is vehicle navigation systems (VNS), as shown in Figure 1.5, such as Google Maps, TomTom [100], etc. VNS collect real time traffic information by the widely used mobile devices. Drivers who use these VNS apps are able to check the current global traffic conditions easily to adjust their travel route plans. ICT may also be combined with administrative policies to reduce traffic. For example, automatic number plate recognition cameras are used in the aforementioned cases of Beijing and London to locate and confirm the violators and drivers who should be charged. In Singapore, they have introduced the real-time variable road pricing to charge drivers with different rates according to historical and real-time traffic conditions.

Figure 1.5: A use case of vehicle navigation systems

Nowadays, the aforementioned general urban road traffic congestion problem is not difficult to cope with using the state-of-the-art traffic prediction technology because this congestion is mostly recurrent, appears during the commuting peak hour or around the frequently used roads. Specifically, the daily traffic is predictable as the traffic variation within specific time periods (e.g. one hour, one day, one week) is known with good certainty. This is due to the fact that the generation of such type of traffic depends on the time and location of drivers' work, study, and home, which, in general, changes relatively infrequently. In

contrast, the non-recurrent traffic congestion, which is caused by **en-route events**, such as poor weather conditions, sports or concerts in stadiums, car incidents, road works, unplanned parades, etc., can deteriorate the global traffic conditions in critical areas within a very short time period. In order to ensure drivers' on-time arrival, according to the latest urban mobility report [2], they need to allow more than twice the anticipated travel time during peak hours to account for these unexpected incidents, as is shown in Figure 1.6.

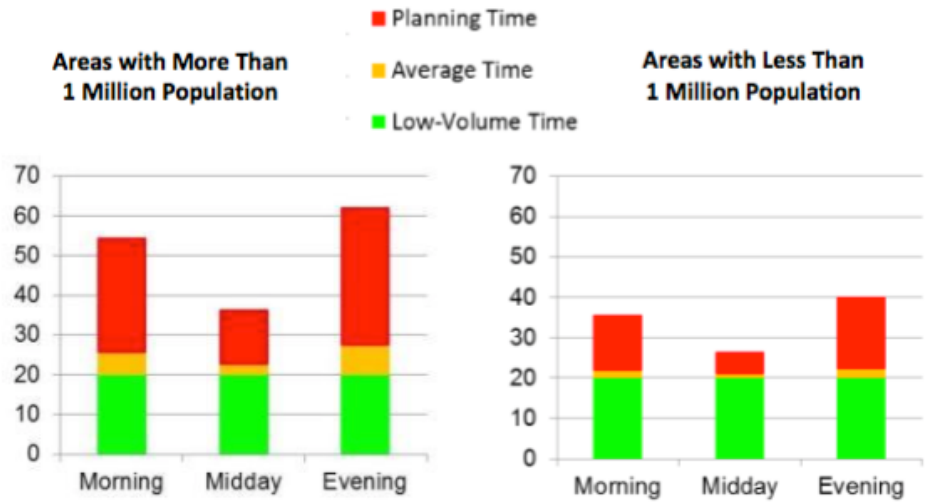


Figure 1.6: How much extra time should you allow to be 'on-time' [2].

As the fastest real-time traffic update for the two systems mentioned above (i.e. ATSC and VNS) is at least 2 minutes, and more commonly ranges from 5 minutes to 20 minutes, although they perform well when dealing with the recurrent congestion, they cannot deal with the non-recurrent congestion quickly enough. The response time is vital for the alleviation of en-route congestion using vehicle rerouting. If a rerouting decision is delayed, the vehicle may already have proceeded through one or more junctions, which reduces the rerouting opportunities. Specifically, under the stable traffic flow[2], a vehicle needs 20-35 seconds to traverse an urban road segment including its intersection. Thus, the 2-

[2] HCM. Highway Capacity Manual. Washington, DC: Transportation Research Board; 2010

20 minutes update frequency that the existing ITS have is not sufficient to provide rerouting suggestions under non-recurrent traffic congestions.

In practice, the existing solutions to deal with the en-route events are divided into two categories. The first category makes use of static or variable traffic signs. As shown in Figure 1.7, after the occurrence of unexpected en-route events, the local authority often uses the static or variable traffic signs to notify road users, mostly drivers, to take a suggested detour or seek alternative routes by themselves. The second category is based on the application of VNS. For instance, Waze, which was acquired by Google in June 2013, periodically scraps road events information published by the official social media (e.g. Twitter and Facebook), or reported by trustworthy users, and presents them on Google Maps. This allows the drivers to adjust their own travel plans but without considering the future impact on the global traffic around the event area.

Figure 1.7: Traffic signs used after the occurrence of events.

1.2. Problem Statement

Currently, there are no effective systems used in practice to deal with the unexpected urban road traffic congestion, and there is no theoretical work, to the

best of my knowledge, which investigates this non-recurrent congestion problem with practical assumptions and evaluations in realistic urban scenarios[3]. The problem studied in this thesis is stated as follows:

How to efficiently reduce non-recurrent traffic congestion in urban roads using advanced vehicle re-routing mechanisms?

There are several clarifications on the scope of this problem, stated as follows:

1. I focus on **urban road scenarios** rather than highway scenarios. In the latter, the number of junctions is lower and the road length is much longer compared to the former. Therefore, the designed solution for highways should be more focused on improving the real-time responsiveness for event information detection and notification, rather than rerouting the vehicles. Conversely, there are many possible solutions using vehicle rerouting to alleviate such congestion in urban scenario, due to its more sophisticated road network topologies. This increases the complexity to provide an effective approach, but also creates huge potential to deal with this problem with much better performance.

2. I focus on the **non-recurrent congestion** problem rather than recurrent congestion. Thus, my proposed system is expected to react to the event quickly and should be effective in short time periods, but does not have to be optimal. Moreover, this thesis studies the non-recurrent congestion problem caused by full link closures, rather than partial reduction of flow, single lane closure, etc.

3. To verify that the proposed system eventually **reduces the congestion**, many congestion indicators are used in this thesis to confirm that this objective is achieved indeed. Among these indicators, I use travel time index (TTI) for measuring average travel time and planning time index (PTI) for measuring travel time reliability.

4. The methodology used in the proposed system is to **re-route vehicles** to

[3] The detailed limitations will be illustrated in the next chapter "state-of-the-art"

achieve my goal. The recurrent congestions are caused by the increased traffic demand where the traffic distribution over each local area is more or less balanced as the roads are all highly occupied. While, the non-recurrent congestions are caused by the lack of event notification to drivers, which often leads to imbalanced traffic. Theoretically, rerouting vehicles leads to redistribution of the traffic on the fixed road infrastructure topology. This achieves more efficient use of road resources. An ideal re-routing approach should also let vehicle to reach its destination as soon as possible, which sometimes contradict the traffic balancing. Practically, no system is currently being used as it is complicated to find a route with the desired quality within a very short time, and it is hard to convince drivers to take the suggestions from a third party.

In addition, there are some practical constraints related to the studied research problem. The first constraint consists of keeping the number of rerouted vehicles as low as possible. In practice, drivers are reluctant to change their planned route unless they have to do so when a road closure occurs. Therefore, even though in most cases rerouting more vehicles will lead to higher traffic congestion reduction, I still need to keep the least number of rerouted vehicles to enhance the practicability of the proposed system. The second practical constraint is that the proposed system should not cause serious unfairness issue for other vehicles. That is, the reduction in average travel time should not come at the expense of a significant increase in travel time for a small number of individuals.

1.3. Contributions

The contributions of this thesis are summarized as follows:

- I investigated the possibility of extending the existing ITS infrastructure by adding a route guidance component. To this end, I have conducted comprehensive performance (i.e. execution and functionality) evaluations and comparison of four typical vehicle routing algorithms (i.e. static A*,

static Dijkstra, dynamic A*, dynamic Dijkstra) based on the architecture of the existing ITS. I found that the dynamic A* can consume a lot of system resources, in terms of both computation and storage, although it guarantees the best performance for the least trip time. If the deployed system has limited capability, then the static A* algorithm is a more suitable alternative.

- I proposed a next road rerouting (NRR) scheme to deal with the non-recurrent congestions based on existing systems. My scheme fits perfectly the real-time requirement of non-recurrent congestions problem because it only gives the most promising next road choice, rather than the full route, for vehicles whose planned routes include the closed road. After being routed in uncongested areas, it starts using VNS to get the full route to complete the rest of its journey. Thus, it reduces a significant burden for the servers in TOC, returns in useful timeframe, and still manages to ensure effective control of the global traffic conditions, as shown in my simulation results based on an urban scenario.

- I also proposed adaptation mechanisms for NRR to achieve better control of the current traffic by rerouting various vehicles. This adaptive feature enables NRR to calibrate its algorithmic and operational parameters. The algorithmic parameters in NRR are the weight values for four different factors used in the routing cost function. NRR uses coefficient of variation to evaluate the deviations for each of the four given factors measuring the status of each available next road choices. The more deviation it has, the higher weight value it should be given. The operational parameters of NRR are the group of agents that should be enabled to perform rerouting. Generally, the more agents enabled, the better performances tend to be achieved. Unfortunately, the number of vehicles involved in the rerouting process will be increased as well. The intuition proposed in NRR is to enable agents only when their traffic is not balanced. These agents are chosen by k-means algorithm. The adaptive feature is facilitated by

Vehicular Ad-hoc Networks (VANETs) technology, which can provide high resolution traffic information. In VANETs, a beacon is broadcasted by each vehicle at least every 0.1 seconds to share the real time status of the vehicle including its speed, headings, steering angle, etc. The simulation results show that a further improvement of global traffic conditions is achieved using NRR with adaptation mechanisms, compared to NRR without adaptations and two other state-of-the-art practical approaches.

1.4. Thesis Structure

The structure of this thesis is outlined in chapters as follows:

- *Chapter 2* describes the related work on the research problem of this thesis. I firstly introduce two categories of vehicle routing algorithms in the transportation domain: routing for one O/D (Origin/Destination) pair and multiple O/D pairs. Then, I focus on multi agent systems in urban traffic management, which is followed by the most recent work on reducing non-recurrent congestion. Finally, I summarize the limitations of the related work as oppose to the research problem.

- *Chapter 3* investigates the performance of the four most commonly used vehicle routing algorithms based on centralised ITS under various routing requests.

- *Chapter 4* overviews the architecture of my proposed next road rerouting system. This chapter provides deployment details of NRR as an extension of existing ATSC. Then, a typical rerouting process using NRR is introduced. Finally, this chapter defines the fundamental elements in multi agent design of NRR and explains the agent coordination mechanism in particular.

- *Chapter 5* emphases closely on my next road rerouting idea. Specifically, to justify the effectiveness of NRR, I provide a detailed introduction to the four factors used in measuring the cost of next road

choice. Besides, I also propose an efficient weight value allocation mechanism to identify the importance of the four factors in each different routing request.

- *Chapter 6* applies VANETs on NRR to study how the granularity of traffic information can affect the rerouting decision. An adaptive mechanism is also proposed for the selection of operational parameters to avoid unnecessary rerouting.

- *Chapter 7* draws the conclusion and discusses some future research directions that can achieve further reduction in non-recurrent urban traffic congestion.

Chapter 2
State-of-the-art

This chapter introduces the state-of-the-art approaches that are relevant to the research problem investigated in this thesis. Firstly, vehicle routing algorithms to find the least costly route for one Origin and Destination (O/D) pair are presented. Secondly, vehicle route assignment strategies seeking the lowest total cost routes for multiple O/D pairs are overviewed along with the discussion of two conventional traffic/route assignment objectives: user equilibrium (UE) and system optimum (SO). Thirdly, the up-to-date multi agent based traffic management systems for traffic light signal control and vehicle route optimization are discussed. Fourthly, the approaches particularly focusing on reducing non-recurrent urban traffic congestion are introduced. **Finally, the limitations of all the above approaches are highlighted for addressing the thesis research problem**.

2.1 Brief Introduction of Road Traffic Modelling

This section briefly introduces how road traffic is modelled and how the performance of a road traffic network is measured. Traditional traffic modelling approaches consider the road network as a directed acyclic graph, of which nodes represent junctions while edges represent roads and their corresponding driving directions. The key idea in the traditional approach is to model the traffic in the unit of "road" with solutions describing how many vehicles should be assigned to each road, rather than in the unit of "vehicle", with solutions describing what route should be assigned for each vehicle. Specifically, to bridge the perspective from

"vehicle" to "road", a volume-delay-function (VDF) is used to model how travel time varies with the number of "vehicles" on a specific "road". In macroscopic view, both computer network modelling and road network modelling have the same performance objective, namely, to reduce the end to end packet delay in data networks or corresponding trip time for vehicles. However, in microscopic view, road traffic modelling has no concept like packet arrival distribution which can be found in computer network modelling. This is because the majority of trip time on the road network is spent on each road (i.e. modelled as link delay), while the main part of end to end packet delay occurring on each router (i.e. modelled as node delay) over its route. More details can be found in the section 2.3.1.

Although the traditional road network modelling is based on each road segment, due to the instability of urban road traffic (i.e. short road length, various road conditions, and frequent disruption by signalized junction), there is a strong practical demand for assigning routes to each vehicle. Concretely, to reduce travel time (i.e. or travel distance) for a single trip, routing algorithms for one O/D pair can be used. For example, A* can compute the route with the least travel time for each vehicle then adapt to the road network with its heuristic function designed to find the lower bound of travel time between any given O/D pairs. These algorithms are elaborated in section 2.2. Additionally, routing algorithms for all O/D pairs can obtain the reduction of the total travel time for all trips. For instance, User Equilibrium traffic condition can be achieved by the iterative execution of Dijkstra's Algorithm, of which solutions are routes for each vehicle. More details of these algorithms are discussed in section 2.3.

The remaining sections (i.e. section 2.4 & 2.5) in this chapter are the descriptions of various system implementations, based on the fundamental concept of road traffic modelling introduced above.

2.2 Routing for One O/D Pair

This section introduces several important variants of shortest path finding algorithms used for route planning. In addition, their applications for vehicle route guidance in the transportation domain are also reviewed.

2.2.1 Dijkstra's Algorithm for Shortest Path Problem

The foundation of finding a personalized route for one O/D pair is the study of the classic shortest path problem. In a connected graph, the shortest path problem consists in finding the path (i.e. the set of consecutive edges) with the least defined cost from the source to the target vertex. In 1958, E. W. Dijkstra designed an algorithm [5], named after himself, which firstly solved the shortest path problem with its optimality guaranteed using Proof by Contradiction.

Algorithm 2.1: Dijkstra's Algorithm

```
1.  function Dijkstra(graph, source, target):
2.      create unvisited vertex set Q
3.      for each vertex v in graph:
4.          dist[v] ← ∞
5.          prev[v] ← Ø
6.          add v to Q
7.      dist[source] ← 0
8.      while Q is not empty:
9.          u ← vertex in Q with min dist[u]
10.         if u = target:
11.             return dist[], prev[]
12.         else:
13.             remove u from Q
14.         for each neighbor v of u:
15.             alt ← dist[u] + length(u, v)
16.             if alt < dist[v]:
17.                 dist[v] ← alt
18.                 prev[v] ← u
19.     return dist[], prev[]
```

As shown in Algorithm 2.1, Dijkstra's Algorithm (DA) takes 3 input elements: the *graph*, the *source* and *target* vertex. It returns 2 output elements: one array *dist[]* for retrieving the cost value for the shortest path. This array (*dist[]*) stores the minimum cost value from source vertex to one of the other vertices

indicated in the array index, while the array *prev[]* is used for retrieving the shortest path sequence. *prev[]* stores the predecessor vertex of a certain vertex given in the array index, according to the found shortest path. DA initializes the value of *dist[]* as infinity, the value of *prev[]* as empty for all vertices in the given graph, and adds all initialized vertices into a set Q, which is used for recording the unvisited vertices during the following execution process of DA. As the last step before the searching process of DA, the minimum cost value from source to destination, formalized as *dist[source]*, is set to 0. The algorithm starts from the source vertex searching each of its neighbours by updating their *dist[]*, then it moves on to one neighbouring vertex with the minimum *dist[]*. DA repeats this searching process iteratively until the *target* vertex is chosen as the current vertex or until all vertices are examined. Based on the framework of DA, many variants of this algorithm have been proposed in the following decades using techniques such as improving the data structure, introducing new heuristic functions, and reinterpreting the definition of the cost function.

2.2.2 Dijkstra's Algorithm using Heap or Bucket

The time complexity of DA depends on the implementation of Q, which is used for choosing the neighbouring vertex with minimum cost. For the given graph, the number of vertices and edges are denoted as n and m respectively. If DA does not use min-priority queue to implement Q, then its time complexity is $O(n^2)$. The min-priority queue is used to implement Q as shown in the two most notable [23] DA variants: DA using buckets (DIKB) [6] and DA using Fibonacci heap (DIKF) [7]. The bucket data structure is used in DIKB to reduce the time complexity of DA to $O(m + nC)$, where C denotes the maximum value of the edge cost in the given graph. The merit of DIKB is brought by the linear time complexity operation for insertion and deletion of Q. However, DIKB has huge storage requirements in creating maximally nC buckets. Moreover, DIKB needs to pre-process for the edge cost value to ensure C, the maximum value of the edge cost, is an integer and small

enough. DIKF also reduces DA time complexity to $O(m + nlog(n))$ without any extra storage requirement and pre-processing. Therefore, DIKF is the suggested algorithm for the most cases in the transportation domain, according to the results presented in [8], where DIKF is evaluated in realistic transportation networks and compared to DIKB and many other shortest path finding algorithms. There are many techniques for accelerating the shortest path computations as outlined in [24]. Some of them are still useful in large-scale scenarios.

2.2.3 Heuristic Shortest Path Finding and Re-planning

The most important variant of DA is A*[9]. A* is the foundation for a heuristic search on the framework of DA. Specifically, to compute *alt*, the potential cost value when choosing a candidate node for next step expansion, in line 15 in Algorithm 2.1, instead of *alt = dist[u] + length(u,v)*, A* uses *alt = dist[u] + length(u, v) + est(v)* by adding an output of a heuristic function *est(v)* to estimate the cost from one of *u*'s neighbours *v* to the given *target* vertex. In general, a heuristic is considered as a trade-off between computation time and optimality. Given a large-scale problem, heuristic-based methods are often used to provide a sub-optimal solution within acceptable time range. Unlike many other heuristic algorithms, A* with a well-designed, or more formally called admissible heuristic function can guarantee an optimal solution, but with a significantly reduced search space compared to DA. For example, in a road network scenario, the heuristic function in A* can be implemented using Euclidean distance. As the geographical length of any possible routes between any O/D pair cannot be less than its corresponding Euclidean distance, this heuristic implementation is called admissible, which means it never overestimates the cost in practice.

In dynamic environments, where the edge cost is changing over time, the optimal route needs to be updated accordingly. The intuition to do re-planning is to run A* from scratch once the graph is updated. However, re-planning from the scratch is a waste of computation when the changing environment does not or only

has minor effect on the previous optimal solution. D* Lite [10] is an efficient re-planning algorithm that only looks at certain areas that have their edge cost changed and repairs the previous route only if it is necessary. This process is achieved mainly by introducing a new heuristic function called "one-step look ahead cost", which is able to detect the changes in environment. Moreover, the whole search process of D* Lite is done in the reverse way from the target vertex to the current vertex, thus preventing a lot of computation on updating the estimated cost from the moving current vertex to the target. Compared to re-planning using A*, D* Lite is more efficient by nearly two orders of magnitude [10]. In addition to the dynamic environment, the typical A* algorithm is also not applicable if a route solution is needed quickly in a complex environment, where the number of vertices and edges in the given graph is excessively large. Anytime Repairing A* [11] (ARA*) solves this problem by using "inflation factor ϵ" to increase the output value of the admissible heuristic function in the typical A*. It is proven that maximally up to ϵ times computation cost could be saved when $\epsilon > 1$. The larger ϵ value is set, the faster the algorithm runs, and the worse the optimality of the route will be. ARA* trades off the speed and the optimality by decreasing the value of ϵ iteratively from a relatively large value, until it reaches the time threshold. Anytime Dynamic A* (AD*) [12] combines the advantages of the two algorithms to deal with the dynamic and complex environment in real-time.

2.2.4 Shortest Path Algorithm for Vehicle Road Guidance

Early stage research in applying shortest path algorithms for vehicle road guidance consists of three major directions: bi-directional search, sub-goal search, and hierarchical search. Due to the lack of powerful computation capability and efficient geographical data techniques, these three directions have the same objective: reducing the search spaces. To achieve this objective, as shown in Figure 2.1, bi-directional search [26] starts the process from both directions in parallel, one from origin to destination, the other from destination to origin, until both search processes meet at the same vertex somewhere between origin and destination.

However, this termination condition cannot be satisfied all the time. Sub-goal search [27] is performed by manually selecting some points in the middle of the given O/D pair, before running bi-directional search. Thus, sub-goal search will always terminate in finite time. Its downside is that the principle to choose the appropriate "sub-goal" ensuring the best efficiency is subject to the map topology, which cannot be determined for general case. The hierarchical search method [28] has a pre-processing stage which splits the map data into multiple levels according to the priority of road infrastructure. It turns out to be very efficient for routing long trips, as the hierarchical search avoids a lot of computation for areas that are neither close to the origin nor the destination.

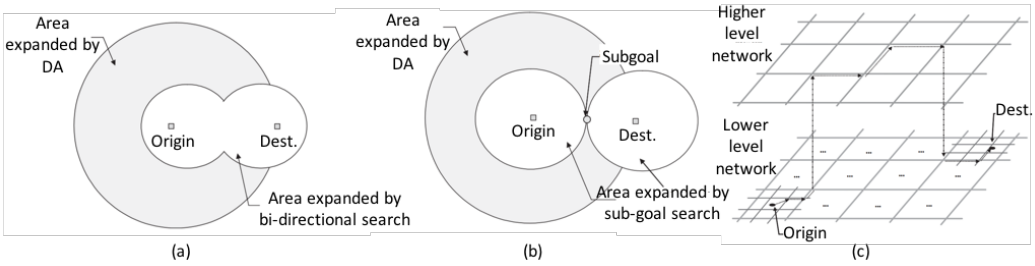


Figure 2.1: Bi-directional search (a), sub-goal search (b), and hierarchical search (c) from shortest path in road network [25].

Most recent research activities on vehicle route guidance are focusing on proposing various extensions of the cost function, which is *length(u, v)*, as shown at line 15 of Algorithm 2.1. In the transportation domain, the cost can be interpreted using metrics such as travel time, travel length, travel time reliability, fuel consumption, number of turns, or a combination of some of them. For example, Kanoh [13] uses a virus genetic algorithm to obtain multi-objective optimal routes considering various route types including number of junctions, type of turns, number of lanes, width of roads and so on. In [14], in a wireless sensor environment, the multi-attribute decision-making (MADM) method [18] is used for computing the best route in real-time in terms of lowest travel time, shortest travel distance, and the best lane status (largest sum of road width). Ronan and Gabriel [15] proposed an algorithm called EcoTrec that leverages the vehicular ad-hoc network (VANET) to collect and exchange information in order to obtain the optimal eco-

route. EcoTrec applies Handbook Emission Factors for Road Transport (HBEFA) formula [19] to estimate a vehicle's emission and achieves a good trade-off among travel distance, travel time, and vehicle emission. Instead of extending the number of relevant factors for the routing cost, some work focused on a particular cost factor and provided deep investigation. For instance, in [16], the travel time factor in the road network is modelled as a discrete time-dependent network where the weight value on each particular edge varies over discrete time dimension. This A* variant can provide a route for a given O/D pair with guaranteed least amount of travel time. However, due to the difficulty of traffic data acquisition and storage [17], this algorithm is seldom used in practice.

2.3 Routing for Multiple O/D Pairs

Routing vehicles for multiple O/D pairs is also called traffic/route assignment in the transport modelling domain. One category of assignment technology consists of directly applying single O/D pair routing for all given O/D pairs. However, this method usually does not result in reasonable traffic as it assumes that the road network has unlimited capacity, which means the travel time on each road is only in proportion to the road length without considering the number of vehicles running on this road. In a more realistic model, the route assignment problem should be considered as: given a set of O/D pairs, how to minimize a certain travel cost, usually the total travel time, constrained by a fixed road capacity.

2.3.1 User Equilibrium and System Optimum

There are two network traffic states that an ideal route assignment strategy can achieve, user equilibrium (UE) and system optimum (SO) [29]. According to the statement of Wardrop's first principle [20]:

"The journey times on all routes actually used are equal, and less than those which would be experienced by a single vehicle on any unused route."

This traffic state is regarded as UE, which is similar to Nash equilibrium [21] in game theory. In this state, the traffic equilibrium is reached following an approach in which vehicles optimize their routes autonomously, until no faster route can be found. Another traffic state SO is described in Wardrop's second principle [20] as:

"The average journey time is a minimum."

SO implies that the best road network performance that a route assignment strategy can achieve is enabled through the vehicles' cooperation. The objective functions of traffic assignment for UE and SO can be formalized as follows [30]:

$$UE\text{: } \underset{x_a}{\operatorname{argmin}} \sum_a \int_0^{x_a} t_a(x)dx$$

$$SO\text{: } \underset{x_a}{\operatorname{argmin}} \sum_a x_a * t_a(x_a)$$

subject to:

$$\textstyle\sum_p f_p^{od} = q_{od} \text{: } \forall o, d$$

$$x_a = \sum_o \sum_d \sum_p \delta_{a,p}^{od} * f_p^{od} \text{ : } \forall a$$

$$f_p^{od} \geq 0 \text{: } \forall p, o, d$$

$$x_a \geq 0 \text{: } a \in A$$

Equation 2.1: Formulation of traffic assignment problem to achieve SO and UE.

where t_a is travel time on the road a, x_a is the equilibrium flow assigned on the road a, A is the set of all roads in the given map, p is the route, q_{od} is the trip rate / total traffic flow between o and d, f_p^{od} is the traffic flow on route p connecting O/D pair od, $\delta_{a,p}^{od}$ equals to 1 when road a is on the route p connecting O/D pair od, otherwise it equals to 0.

The above equations and inequalities describing the constraints are the principle of flow conservation and non-negativity. Both objective functions for UE and SO can be solved by an appropriate optimization technique, usually by Frank-Wolfe algorithm [22], if the following four assumptions are satisfied:

1. **The volume-delay-function (VDF): $t_a(x_a)$ should be differentiable and non-decreasing.** The purpose of this assumption is to ensure that the objective functions are convex, thus a global optimum can be found. Specifically, Bureau of Public Roads (BPR) [31] function is commonly chosen as the representative implementation of VDF for the traffic planner to estimate the traffic pattern. As shown below:

$$T_f = T_o * (1 + \alpha(\frac{V}{C})^{\beta})$$

Equation 2.2: BPR function.

where:

T_f is actual travel time

T_o is free-flow travel time

V is current traffic flow

C is road capacity

α, β need to be tuned, often their suggested values are 0.15 and 4 respectively [31].

Compared to other types of VDFs [32], BPR has much fewer compulsory parameters and is still proven to be efficient for traffic planning. It is worth noting that its assumption is not quite realistic, especially in urban road scenarios where the road travel time is unstable due to the signalized junction and short road length.

2. **The travel time on a given road is independent of other roads.** This assumption simplifies VDF for all roads by using one variable only, which means that the travel time of a given road is a function of the traffic on this

road only. In practice, the travel time on different roads, especially neighbouring roads, are actually correlated to each other as the traffic on a given road will propagate to its downstream roads.

3. **The network states are perceived as the same for all drivers.** Concretely, this assumption means all drivers have complete knowledge about the global traffic states, when they are making routing decisions. So the case where a faster route is not chosen due to the lack of information for a driver will never occur.

4. **All drivers choose the route to minimize their travel cost.** This assumes that all drivers are making their decisions to achieve one common goal.

The aforementioned traffic/route assignment strategies to achieve UE and SO are called static traffic assignment (STA). STA plays a significant role in the early stage of traffic pattern estimation as it lays the foundation for the subsequent traffic assignment techniques. There are two main research works that investigated STA. The first one is the identification of the theoretical gap between UE and SO. In [33], the gap is quantified in two separate cases: when VDF is linear, the total travel time in UE is at most $\frac{4}{3}$ times more than the one in SO; when VDF is continuous and non-decreasing, then the total travel time in UE is at most twice as the one in SO. The second notable STA research work is the SO traffic assignment constrained by fairness conditions. The work shown in [34] proposed a new approach to achieve the constrained SO, so that its performance still advantages the one of UE, and at the same time ensures no one contributes the global benefit by choosing a much longer and slower route. Additionally, there is an interesting phenomenon called "Braess Paradox" in traffic assignment research. In general, this paradox means under some circumstances, when adding more roads, the congestion level will be counter-intuitively increased. The original statement of Braess Paradox translated into English can be found in [37], along with the mathematical formation, proof, and a case study.

"Braess Paradox" has been studied in [35] under a series of reasonable assumptions, the paradox is likely to occur more frequently than as an anomaly case. In [36], "Braess Paradox" is also tested in realistic scenarios repeatedly, and then the sufficient condition of this paradox is refined as experienced drivers with reasonable decision making behaviour. The Braess Paradox implies that the existing approaches dealing with traffic congestions are risky. Specifically, increasing the road capacity and using selfish routing (i.e. VNS) for all vehicles can sometimes (and in practice, not infrequently) increase the congestion level. This implication highlights the importance of using vehicle rerouting to alleviate non-recurrent traffic congestion, which is the research methodology used in this thesis.

2.3.2 Dynamic Traffic Assignment

The last few years have witnessed the development of computation capability and big data technology. Due to the numerous unrealistic assumptions made by STA, another traffic assignment methodology named dynamic traffic assignment (DTA) is more frequently used by city planners [38]. Compared to STA, DTA [39] models the traffic using a discrete time-dependent network, which means the VDF is also a function of a particular entry time of the vehicle into a certain road, rather than a function of the current traffic volume only. In other words, given the same OD pairs with different departure times, the results are still the same using STA, but different using DTA. A typical DTA for traffic simulation used by city planners is Gawron's dynamic user equilibrium [40], which is used as the default traffic assignment algorithm in the well-known open source urban road traffic simulator, Simulation of Urban Mobility (SUMO) [41]. This algorithm firstly assigns routes for all O/D pairs using shortest path algorithm DA by considering the road length as edge cost. Then it runs the simulation, records the actual travel time on each road, then uses DA to re-assign the routes using DA treating travel time as edge cost. This step is done iteratively until the edge cost for all roads is relatively converged. Unlike the traditional path-based and link-based algorithms, Bar-Gera

[56] designed another "origin-based" algorithm to achieve DTA. For each origin, the algorithm creates a special "bush" data structure which is an acyclic connected sub-graph from the given origin to all possible destinations. For each bush, the algorithm applies the reduced Newton method to iteratively optimize the proportional approach to the flow on each node. Nie [59] improved it by adopting the second-order derivative of the objective function to achieve a more accurate approximation. For the same "origin-based" DTA, B Algorithm [60] balances the flow between the maximum flow path and the minimum flow path. Shin-ichi [57] further improved the B Algorithm for the operations on "bushes" and was evaluated [58] as the best DTA algorithm in terms of the efficiency in computation, memory, and convergence.

2.3.3 Stochastic Traffic Assignment

Another way to increase the practicability of traditional traffic assignment is to use stochastic models [42] to enable the relaxation of the last two aforementioned assumptions of STA. The stochastic traffic assignment assumes that the drivers do not have perfect information of the global traffic conditions of road network, and do not always make their routing decisions reasonably. Specifically, both drivers' traffic knowledge and routing decisions follow a certain probability distribution. The closer the road is to the driver, or the lower cost the route is, then the higher probability a driver will know the traffic or choose the route. There are generally three types of stochastic models in the literature, multinomial probit [63], nested logit [61], and generalized nested logit [62]. The detailed description of these models will not be included in this thesis as stochastic traffic assignment models are computationally expensive (i.e. need to know all/most of the possible routes and the distribution of their usage frequencies) and the benefits these models have for traffic planning still need further investigation.

2.4 Multi-Agent Traffic Management Systems

Agent technology is the key concept for implementing distributed artificial intelligence [43]. Specifically, the paradigm of multi agent systems is well suited for the management of road traffic [44], as the road traffic network can be treated as a collective set of geographically distributed local areas, the traffic state is changing over time in each local area, and this change is sensitive to behaviors from any road network participants (i.e. drivers, pedestrians, traffic regulators, etc.). This section divides multi-agent traffic management systems into two categories: traffic light signal control and vehicle routing optimization, then reviews the state-of-the-art technologies and research in each category.

2.4.1 Multi Agent System for Traffic Signal Control

Traffic light signal control is considered the most typical application of the multi-agent concept in road traffic management. The most widely deployed systems are Sydney Coordinated Adaptive Traffic System (SCATS) [45] and Split Cycle Offset Optimization Technique (SCOOT) [46]. They have been successfully applied world-widely in over 27 [48] countries and over 200 [49] cities, respectively. Both SCATS and SCOOT have a similar 3-tier hierarchy. Take SCATS for example, as shown in Figure 2.2. The basic agent in the bottom layer is each intersection, which is controlled and coordinated by a regional computer according to the real-time traffic information. All the regional computers are then organized by a central server for high level configuration and optimization in a particular city. The agents here are regional computers controlling tens of intersections. The main differences between them are the mechanism of reaction to the real-time traffic information. When the traffic states are updated, SCATS chooses the best traffic light signal plan from several candidates that are configured manually in advance. On the contrary, SCOOT can adjust all the related parameters

(i.e. slip, cycle, offset, etc.) and provide an on-line optimized traffic signal plan. This difference is mainly due to the additional types of traffic data collectors (i.e. sensors and cameras) SCOOT has, while SCATS mainly relies on induction loops. More specifically, the different deployments of loop detectors, for example, have led to the aforementioned difference as well. SCATS installs one induction loop at the downstream for each lane to get the traffic information: occupancy, while SCOOT deploys two loop detectors on each lane, one in downstream, the other in upstream. So that it can retrieve traffic information like, queue length, speed, and occupancy. Therefore, more information allows SCOOT to tune the parameters in a finer granularity. Although SCOOT has more flexibility and advanced control mechanism, SCATS has less deployment cost and proven to have comparable effectiveness. The two systems have dominated the global market in urban traffic control during the last 4 decades.

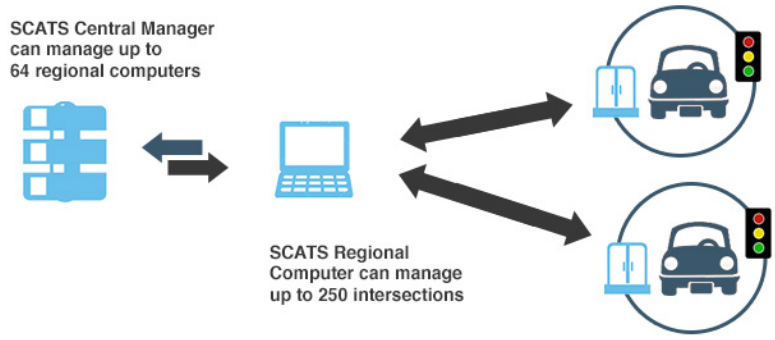


Figure 2.2: 3-tier system architecture of SCATS [47].

Traffic prediction technology frequently appears in the recent research on enhancing the multi-agent traffic signal control system. This prediction technology is driven by the increased number of types of collected traffic information from various deployed sensors. One typical example is InSync [52], which has been applied in 31 states and 2300 intersections in the U.S. up to November 2015. InSync ranked the top in terms of waiting time reduction in several U.S. cities, as evaluated and compared in a survey [50] with four other popular systems. The traffic information collection of InSync is mostly done by Internet Protocol (IP) video cameras. This leads to a huge advantage as many useful microscopic

information can be extracted such as the exact number of vehicles, speed for each particular vehicle, and even vehicle types. By taking advantages of this rich information, InSync can predict short-term traffic conditions to create so call "green tunnels" minimizing the number of stops for the longest platoon. Another way of collecting rich traffic information for predictive control is to use vehicular ad-hoc networks (VANETs), where vehicles are connected and periodically broadcast their states. VANETs are used in the approach proposed by K. Pandit [53] in which an online scheduling algorithm called "the oldest arrival first" is used. It is shown in the presented simulation results that approximately equal-sized platoons can be achieved with significantly reduced intersection delays, as compared to the state-of-the-art algorithm. VANETs are used in a predictive control method proposed by B. Asadi and A. Vahidi [54] that help to achieve minimum use of braking to improve fuel efficiency accordingly. Some pioneering work in this area like [51] have tried to apply multi-agent reinforcement learning for adaptive traffic signal control. Its effectiveness has been proven by the experiments on 59 intersections in the city center of Toronto, Canada. The results show that 27% travel time reduction can be achieved even when all agents are working independently.

2.4.2 Multi-Agent System for Vehicle Route Guidance

Similar to the robotics research in the artificial intelligence area, most multi-agent systems for vehicle route guidance consider each vehicle as an agent, then use different proposed coordination mechanisms to achieve a reduction of total travel cost (i.e. travel time, travel distance, fuel consumptions, etc.). For example, a decentralized delegate multi-agent system [64] is proposed to reduce the traffic congestion using anticipatory vehicle routing. The word "delegate" comes from the pheromones in the ant colony algorithm used for agents to exchange information. CARAVAN [66] puts vehicle agents into VANETs environment, and applies "virtual negotiation" to exchange route allocation cooperatively to achieve the reduction of total travel delay and communication overhead. Sejoon Lim [55] built

a probabilistic path choice model based on a realistic dataset. In this model, each driver's route decision is regarded as a fractional flow. All vehicle agents in the same local area can exchange their route choice to achieve UE or SO. Relying on a central server, participatory routing planning [67][68][69] uses the previously planned routes to estimate future traffic conditions for the incoming routing requests. This routing collaboration among vehicle agents is done by the communication between the cloud server and in-vehicle mobile devices (i.e. smartphone). Last but not least, BeeJamA [65] considers each junction-controlled region as an agent for traffic congestion problems. The agent in BeeJamA plays a role like a router in a computer network by keeping an updated routing table and assigns routes for vehicles. The coordination of agents mimics the process of bees foraging.

2.5 Study of Non-Recurrent Traffic Congestion

To the best of my knowledge, only a few applied research works can be found addressing the reduction of urban traffic congestions caused by en-route events. The most relevant work [70] appears for improving the rate of on-time delivery in logistics[4]. It applies a Markov decision process to achieve a significant increase in delivery performance especially when non-recurrent congestions occur. Some theoretical works like [71] and [72] have proposed dynamic route choice models to maximize the on-time arrival expectations according to the current observed traffic. Many recent related works focus on the study of the impact of this non-recurrent traffic congestion. In [77], Alexander et.al. quantified the non-recurrent congestion impact that the incident-related delay contributes to 30% of total wasted travel time, by conducting statistical studies on the traffic data from loop detectors in California. Osogami [76] concluded from a simulation-based study on the traffic impact caused by various types of road closure including single lane, multiple lanes for single direction and all lanes for both directions. There is

[4] A variant of the typical vehicle routing problem, in which an ideal solution can minimize the travel cost of a certain vehicle when it returns its origin after traversed all the required places.

also research on detecting traffic incidents either from social media [74], or from the massive real-time trajectory data [75].

2.6 Summary of Limitations

As the investigated research problem in this thesis is how to efficiently reroute vehicles in order to significantly reduce non-recurrent congestions in urban areas, the limitations of the discussed related works, with regard to this problem, are summarized as follows:

- **Limitations of one O/D pair routing**: Generally, in a congested road network, one O/D pair routing cannot guarantee or lead to any types of global benefit, meaning that in the face of non-recurrent congestion, the travel time and the road network uncertainty will deteriorate, as all vehicles choose their routes selfishly, according to the Braess Paradox.
- **Limitations of multiple O/D pairs routing**: As the information access and rerouting feedback process should be completed in a very short time, the computation intensive multiple O/D pairs routing, especially its requirement of global traffic and route choice information, is not suitable due to the limited capability of existing techniques.
- **Limitations of existing multi agent traffic management systems**: In general, ATSC focuses on reducing the waiting time at intersections, which is not directly correlated to minimizing the total travel time. Vehicle-based MAS requires the exchange of route choices among vehicles either locally or globally. However, in practice, the route choice information is private to an individual user. Moreover, the route choice for the whole trip is not always available while driving, especially when driving on a long trip or in unfamiliar areas. Additionally, vehicle-to-vehicle communication is not reliable when exchanging relatively long messages such as route choice information in real-time. Region-based MAS needs a lot of infrastructure replacement and upgrade. It also has complex and inefficient hierarchy for collecting traffic and optimizing route assignments.

- **Limitations of studies on non-recurrent traffic congestions**: There is much less work in the literature devoted to determining the appropriate reaction to reduce congestion due to unexpected en-route events. From the theoretical research, [73] reveals that many unrealistic assumptions and the lack of computation/memory performance analysis made theoretical models inapplicable in practice.

The aforementioned limitations are addressed in the next chapters by the proposed solution named "Next Road Rerouting (NRR)". In general, NRR tends to reroute vehicles to the less congested road, thus Braess Paradox would be much less likely to occur. Moreover, NRR only provides vehicles with the best next road direction, hence it fits the rigorous real-time requirement of reducing non-recurrent congestions. It also avoids complex and error-prone coordination mechanisms among vehicles by considering each junction and its controlled roads as an agent. Another worth noting feature of NRR is that it only needs locally accessible information for the rerouting decision without considering the route choices of all relevant vehicles and global traffic conditions. Finally, NRR increases the practicability of research in reducing non-recurrent urban traffic congestions by relying on the widely deployed ATSC and popular VNS.

Chapter 3
Vehicle Routing on Centralised Traffic Management System: A Performance Evaluation Study

This chapter presents an investigation of the effectiveness of adding a vehicle routing component to an existing centralised Intelligent Transportation product (e.g. IBM Intelligent Transportation [101]) in response to en-route events. I firstly discuss the motivation of this work, followed by the evaluation methodology including the compared algorithms, evaluation metrics, and the background of simulation settings. At the end, the evaluation results are presented along with the discussion on how this work inspires my proposed next road rerouting system.

3.1 Motivation

Although the optimal or quasi-optimal vehicle route assignment can significantly reduce the traffic congestion as shown in a lot of recent research, most of the existing Traffic Management Systems (TMS) such as ATCS [50] and IBM Intelligent Transportation [101] lack vehicle routing functionality to reduce the non-recurrent congestion. The most probable reason for this is that unlike other products such as Google Maps, the TMS is mainly designed for traffic managers rather than drivers. Moreover, traditional TMSs lack communication facilities to

push or disseminate messages directly to each road user. A compromised way to implement this communication, namely, a radio station for broadcasting road traffic information has been used for decades. With the fast development of communication technology, the efficient bi-directional communication link between TOC and road user is expected to be realized very soon. In the face of en-route events, drivers are more likely to take route suggestions from trustworthy agencies (e.g. TMS) as these drivers are missing the required information for making decisions in real-time. Therefore, to successfully plug a vehicle routing component into the existing centralized TMSs for non-recurrent congestion reduction, a performance comparison study is highly needed to reveal the effectiveness of popular vehicle routing algorithms under various use cases.

3.2 Evaluation Framework

3.2.1 Overview

The evaluation framework contains a set of metrics (i.e. travel time, travel distance, travel time variability, number of selected nodes, computation time and data storage requirement) and scalability levels (i.e. length and location of O/D pair, e.g. whether O/D pair is in city center, suburban, or remote area), as shown in red and dark blue blocks respectively in Figure 3.1. Besides, the execution process of this evaluation framework is presented on the left, while the required data and its types are illustrated on the right.

Simulation of Urban MObility (SUMO)[5] is used to conduct experiments in this thesis as it is the most widely used open-source microscopic (i.e. at the vehicle level instead of the traffic level) simulator for urban mobility (i.e. mainly the mobility of vehicles). This discrete event simulator is developed in C++, and can be used on Windows, Linux, and Mac OS in the format of a Graphic User Interface or command line console. There are 3 fundamental inputs (i.e. XML files) for any SUMO simulation: map, representing an urban road network (i.e. roads, junctions,

[5] Official website: http://sumo.dlr.de/

and their connections); traffic, indicating the departure time, O/D pair, route, and type for each vehicle; and configuration, containing the required information to control the simulation (e.g. specify input/output files, start and end time). TraCI (Traffic Control Interface)[6] is a set of APIs for retrieving information (e.g. current number of vehicles) and change behaviors (e.g. reroute vehicles) of the running road traffic simulation. All the vehicle routing algorithms in this study are implemented in Python and their solutions replace the original route for each vehicle as soon as it enters the simulation using TraCI. More samples of using SUMO can be found in the official tutorial[7].

The evaluation process starts from the preprocessing of the simulation dataset TAPASCologne [78]. As TAPASCologne has fully covered the greater area of Cologne, and contains lots of information (i.e. road types, shapes of buildings, etc.) that is irrelevant to this study, I firstly cut this dataset into 3 typical scenarios: city center, suburban, and remote areas including both geographic and traffic information. Then, I construct discrete time-dependent road network data by extracting the basic network structure and recording periodically the travel time on each road. Based on the preprocessed data structure, a Python script is applied for randomly generating O/D pairs in different areas with various trip lengths. After the data preprocessing stage, the data storage for each algorithm can be measured. While the routing algorithm runs, it takes two types of input: network data (i.e. connectivity and link cost) and O/D pairs, and after each iteration, it updates the four measurements: travel time, travel distance, travel time variability, and number of selected nodes. At the end of the algorithm execution, the computation time is recorded.

[6] http://sumo.dlr.de/wiki/TraCI
[7] http://sumo.dlr.de/wiki/Tutorials

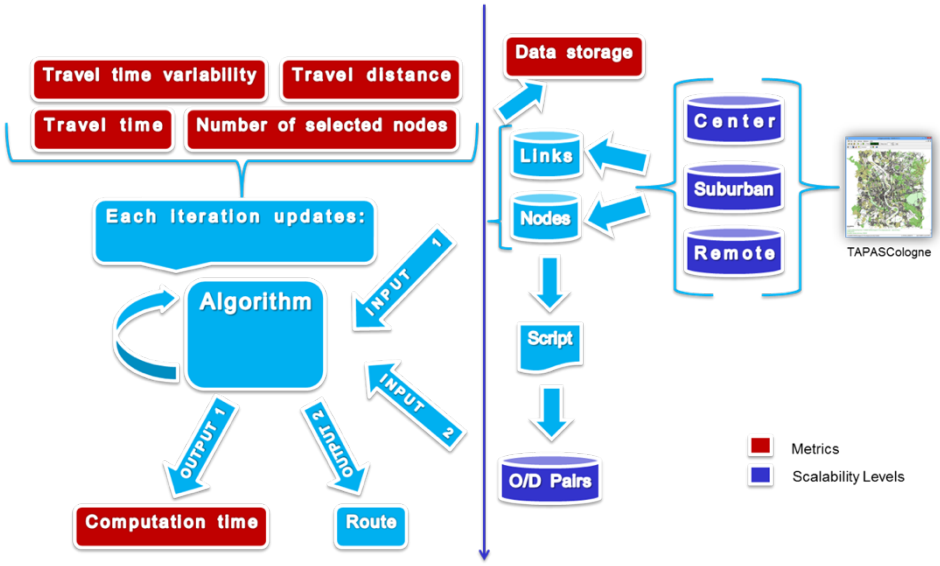


Figure 3.1: Performance evaluation framework for vehicle routing algorithms based on centralized TMS.

In order to compare vehicle routing algorithms, I consider travel distance and travel time to be the most two important factors for drivers' route choice. I firstly choose two data structures to model the road network: a static network for finding shortest distance route, and a discrete time-dependent network for finding the shortest time (i.e. fastest) route. In the static road network, the edge cost is mapped as road length, while in the dynamic road network, the edge cost is defined as travel time that varies under different road entry time. Then, for each road network, I applied Dijkstra's Algorithm (DA) and A*[8] respectively. Thus, the four compared algorithms in this study are denoted as static DA, static A*, dynamic DA, and dynamic A*. It worth to mention as well that Dijkstra's Algorithm is implemented using binary heap for priority queue, while the chosen Heuristic function of A* is the Euclidean distance in the static network and the mixed lower bound [16] in the dynamic network.

[8] Details can be found in section 2.2.1 for DA, and section 2.2.3 for A*

3.2.2 Data Pre-processing

As shown in Figure 3.2, three scenarios are extracted from three different areas in the original larger Cologne road network: city centre, suburban and remote area. Thus, the different simulation scenarios are named centre, suburban and remote, represent a different scalability level at the same time. Although these 3 sub-maps have the same size: 5.350(width) * 9.350(length) = 50.0225km^2, they can still ensure varying scalability levels in terms of the number of nodes and links in the graph representing the road map as well as the traffic load (i.e. the number of cars in a certain time period), as depicted in Table 3.1 and Figure 3.3 respectively.

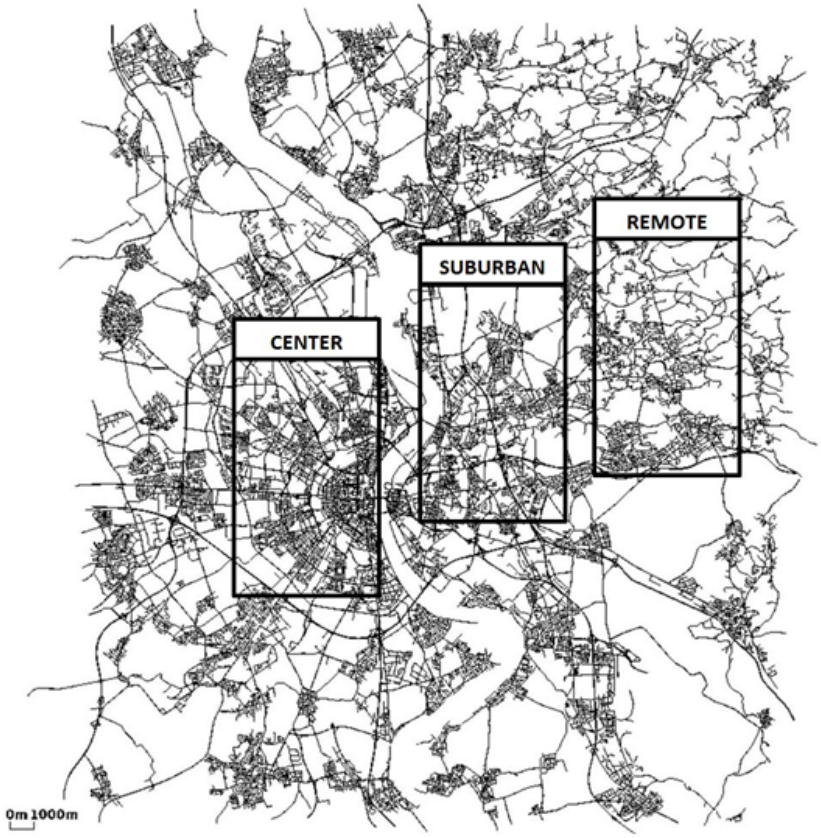


Figure 3.2: The three scenarios as shown in TAPASCologne.

Table 3.1: Number of nodes and links in three scenarios.

	Number of nodes (Junctions)	Number of links (Road segments)
Centre area	4025	8496
Suburban area	2597	5711
Remote area	1810	4170

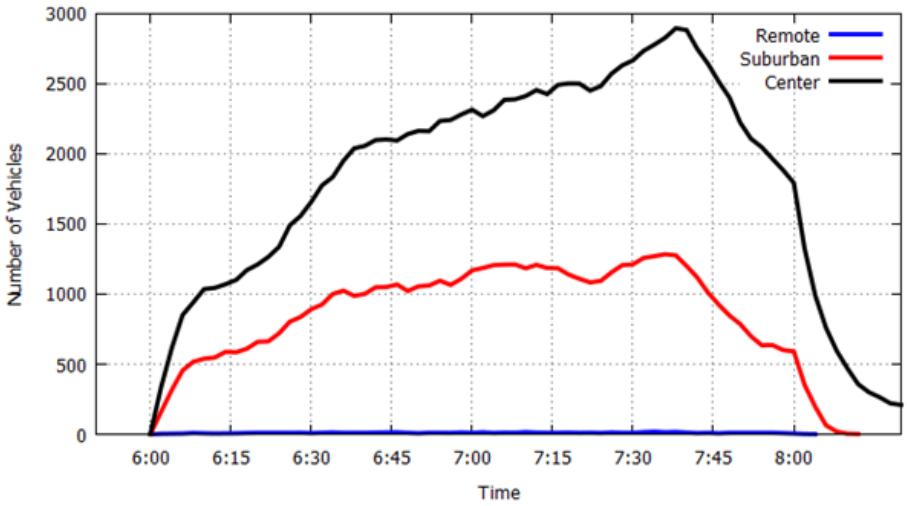


Figure 3.3: Traffic load in the three scenarios.

As the test scenarios have already been set with different scalability levels, for routing algorithms searching a path for each OD pair, I need to find out how the performance of these algorithms varies with the trip length. In practice, the exact trip length can only be known when the car reaches its destination. In order to use trip length as another scalability parameter in my experiments before each trip begins, the Euclidean Distance between the origin and the destination nodes is applied to measure the trip length. Usually, the longest trip distance in an urban area is around 10km, so if a driver plans a trip longer than 10km, the hierarchical routing algorithm [28] is more suitable in this case. Consequently, in my experiments, the testing groups of OD pair are organized into 5 trip length scales, 2km, 4km, 6km, 8km, 10km as depicted in Figure 3.4. It is worth noting also that two OD pairs with similar Euclidean trip lengths may have quite different real trip distance due to the difference in the topology of the area between the origin and destination nodes. To mitigate the potential negative impact caused by this fact, 4 different OD pairs are selected for one trip length group in one specific simulation and the average of their results is calculated. Hence, I have 60 sets of testing results for each routing algorithm.

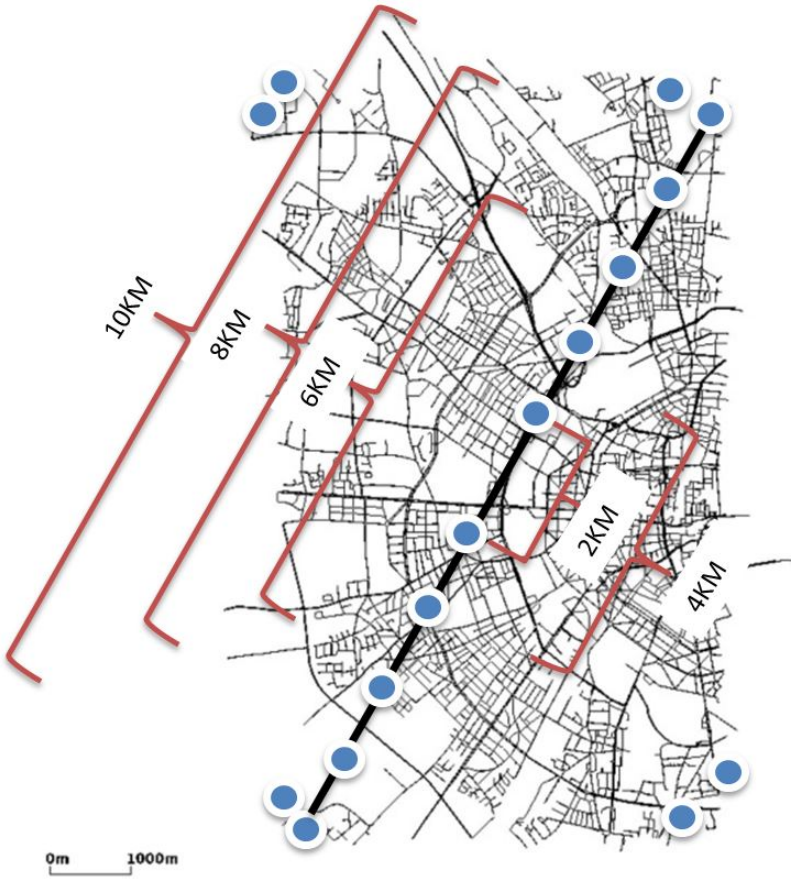


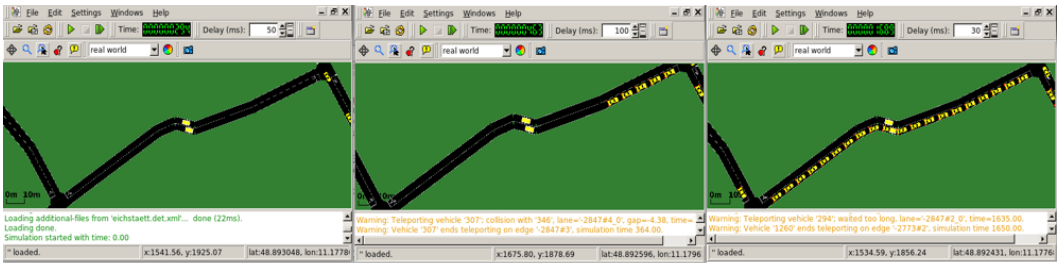
Figure 3.4: Illustrative example of Origin-Destination pairs selection.

3.2.3 Improved Travel Time Calculation

Simulation of Urban Mobility (SUMO) [41] is the most commonly used open-source simulator for urban transportation simulation. Based on SUMO, Traffic Control Interface (TraCI) [79] is one of its official plugins. It can enable the functionality for retrieving information from the running traffic scenario and perform behaviours of vehicles and traffic lights during the simulation run time. Although SUMO and TraCI can provide a powerful and high quality simulation for researchers, there are still some issues on design and implementation that need to be further refined. In this section, a new method is proposed to rectify the problems (i.e. unsuitability for urban scenario and infinity travel time) happening when the SUMO API implementation or well-known equation – BPR (Bureau of

Public Roads) [31] is used for travel time calculation. The improvement is highlighted through comparative evaluation of the proposed method against the original SUMO API under two error-prone cases.

The two existing solutions (SUMO API and BPR) and their disadvantages are presented based on three cases when using SUMO. In this experiment, the road network of German city Eichstätt in SUMO format[9] is used, then I generate the random traffic flow by using the Python script tools in SUMO, and finally use TraCI API to calculate the average travel time every second for each road on the map while the simulation is running.



(a) First case: empty road (b) Second case: few cars (c) Third case: full of cars (100% road occupancy)

Figure 3.5: Three typical cases showing the limitations of the travel time calculation using SUMO API and BPR.

To study the problem of the first travel time calculation, SUMO API : "traci.lane.getTravelTime()", the lane "-2847#2_0" is selected for tracking. As shown in Figure 3.5(a), this lane is empty, so the return value of this API is "18.6962785114s", as the default free flow travel time. As shown in Figure 3.5 (b), the problem is when this car approaches the end of this lane, its speed as well as that of the cars following it is decreasing to zero. Therefore, the travel time is increasing sharply to infinity. I need to prevent this unrealistic value as in practice it should depend on the duration of the red traffic light, which is usually a few minutes but not infinity. I assume that the implementation of the SUMO API for calculating travel time is just to divide the road length by the average vehicle speed.

[9] http://sumo.dlr.de/wiki/Tutorials/OSMActivityGen/eichstaett.net.xml

And to compute the latter, it simply sums the speed of all vehicles running on the road and then makes it averaged. This assumption is confirmed by the source code of this API located at "SUMO_HOME/src/traci-server/TraCIServerAPI_Lane.cpp" where "SUMO_HOME" is the home directory where user stores the SUMO source code.

I track the same lane "-2847#2_0" to investigate the problem of using BPR as shown in Equation 2.2 to calculate the travel time in the urban scenario. For the implementation of BPR, a set of induction loops are deployed on the middle of all road segments to record the number of cars passed through during a time interval to calculate the current traffic flow. The problem is shown in the comparison of two scenarios: first, as in Figure 3.5(a), there is no car running on this lane, thus the volume is zero, the current travel time is the free-flow travel time; this scenario is reasonable. Second, in Figure 3.5(c), when the lane is almost full of cars and they just stand still for at least one time interval, thus there is no car running through the induction loop, therefore the current volume is zero, similar to the first case. This means the travel time is incorrectly computed as the free flow travel time. However, in this case the current travel time should be much slower than the free-flow travel time.

To overcome the aforementioned problems, a simple solution is proposed and implemented that ensures more accurate calculation of travel time in urban scenarios. In this solution, each road segment is considered as two separate parts. One is occupied with vehicles, while the rest is unoccupied. So for the unoccupied part, the maximum allowed speed is used to calculate the travel time while the average vehicle speed is used for the calculation of the occupied part. Particularly, I introduce the minimum vehicle speed for the occupied part and set it to[10] 0.1 m/s for the case where all the vehicles on the road are standing still. This is because those vehicles will not stop forever, they are just waiting for the chance (i.e. green

[10] Inspired from http://sumo.dlr.de/userdoc/Simulation/Output/TripInfo.html as it defines the "waitSteps" as the number of steps in which the vehicle speed was below 0.1m/s

traffic light or congestion mitigation in the road ahead) to go. Finally, the temporal results of the above two parts are summed as the travel time for each road. As each simulation step in SUMO lasts 1 second, the average travel time is calculated every 30 seconds. the experiment results are outlined in Table 3.2 . Note that the first case lasts from the 30^{th} second to 60^{th} second; the second case lasts from the 480^{th} second to 540^{th} second; the third case lasts from the 1440^{th} second to 1500^{th} second. From these results it can be seen that the improved travel time computation shows more stable outputs. It avoids the occurrence of infinity values when the average speed is zero, and distinguishes the two cases where the current traffic flows all equal to zero.

Table 3.2: Travel time calculation results (unit: second)

Simulation Time Stamp	SUMO API	BPR	Improved Calculation
30	18.70	18.70	18.70
60	18.70	18.70	18.70
480	92.53	18.70	43.59
510	1000000.0	18.70	611.49
540	1000000.0	18.70	611.49
1440	531.90	20.54	512.98
1470	1000000.0	20.54	1500.69
1500	1000000.0	20.54	1500.69

3.2.4 Evaluation Metrics

The metrics used in the performance evaluation of vehicle routing algorithms are the number of selected nodes, computation time, data storage requirement, travel distance, travel time, and travel time variability.

- N**umber of selected nodes.** The number of selected nodes is a widely used metric in artificial intelligence research to measure the magnitude of search space of a certain shortest path algorithm. It is an indicator to show the theoretical efficiency of a shortest path algorithm, the less is the better.
- **Computation time.** The computation time is measured in seconds to show how fast a shortest path algorithm runs. It is different from the previous metric "number of selected nodes" because sometimes the algorithm can decrease the number of selected nodes, but at the same time it may bring too many time-consuming computations such as square or square root operations (e.g. compute Euclidean distance). Hence, the computation time is an indicator to assess the practical efficiency of a shortest path algorithm.
- **Data storage requirement**. The dynamic memory usage is not easy to be monitored during the algorithm's execution. Therefore, I measure the data storage requirements as it is proportional to the memory cost. Some algorithms show the best performance in terms of computation time but this advantage may cost large memory space usage. Although the storage space is not as big an issue as it used to be due to recent developments in data storage technology, it is still one of the key indicators from an engineering perspective, especially when deploying or optimizing the operations of the existing large scale ITSs.
- **Travel distance** and **travel time**. The travel distance and travel time represent the realistic length of the route travelled by the vehicle and the time spent over the trip. These two metrics are very important to drivers as the cost of a route. Many more meaningful costs depend on them, such as, fuel consumption and emissions. Generally[11], the longer

[11] The fuel consumption and emissions are also highly depend on the type of vehicle's engine, road conditions, and weather[102].

time or distance a vehicle travels, the more fuel it consumes, and more emission it produces.

- **Travel time variability.** The travel time variability ($TT_{variability}$) indicates the uncertainty of the travel time for a route. Specifically, according to the historical traffic data, it describes how travel time varies given a certain route. In this study, the travel time variability is calculated based on Polus's study [80], as shown in Equation 3.1. In the simulation of this evaluation study, for each road, 240 average travel time samples have been collected over the period from 6:00am to 8:00am with 30 seconds sampling interval. Subsequently, these samples are used to calculate the standard deviation and AVG_TT (i.e. average travel time). Then, the travel time variability for the same road can be calculated. Finally, the travel time variability for a certain route is the summation of the travel time variability for all the individual roads along this route, under the assumption that the travel time variations on road segments are independent from each other.

$$TT_{variability} = \frac{StandardDeviation}{AVG\_TT}$$

Equation 3.1: The calculation of travel time variability using Polus's method.

3.3 Evaluation Results

The results shown in Figure 3.6 highlight the theoretical performance for different vehicle routing algorithms. Generally, for all algorithms, the number of selected nodes decreases [16] gradually with the decrease of scalability level (i.e. the size of the road network that varies from centre, suburban and remote areas) as well as the trip length. These results lead to some interesting conclusions. First, the dynamic and static versions of DA exhibit similar performance and are much less effective in the magnitude of search space (i.e. number of selected nodes) compared to A*, which means that DA confirms its lack of advantage from a design point of

view. However, due to the ease of its implementation, as shown in the following test, DA is still useful under many circumstances. Second, due to the advanced design of its lower bound, dynamic A* always performs the best and left the other three algorithms far behind even compared with static A*. The only exception is when the trip is planned in the centre area with a length of 2km, where both dynamic A* and static A* show the same theoretical performance. In this case, static A* is recommended for the sake of implementation simplicity. Third, it is found that in the remote area scenario, the theoretical performance of static A* shows clear degradation when the trip length gets longer (i.e. $\geq$ 6km), especially when the trip length is about 10km.

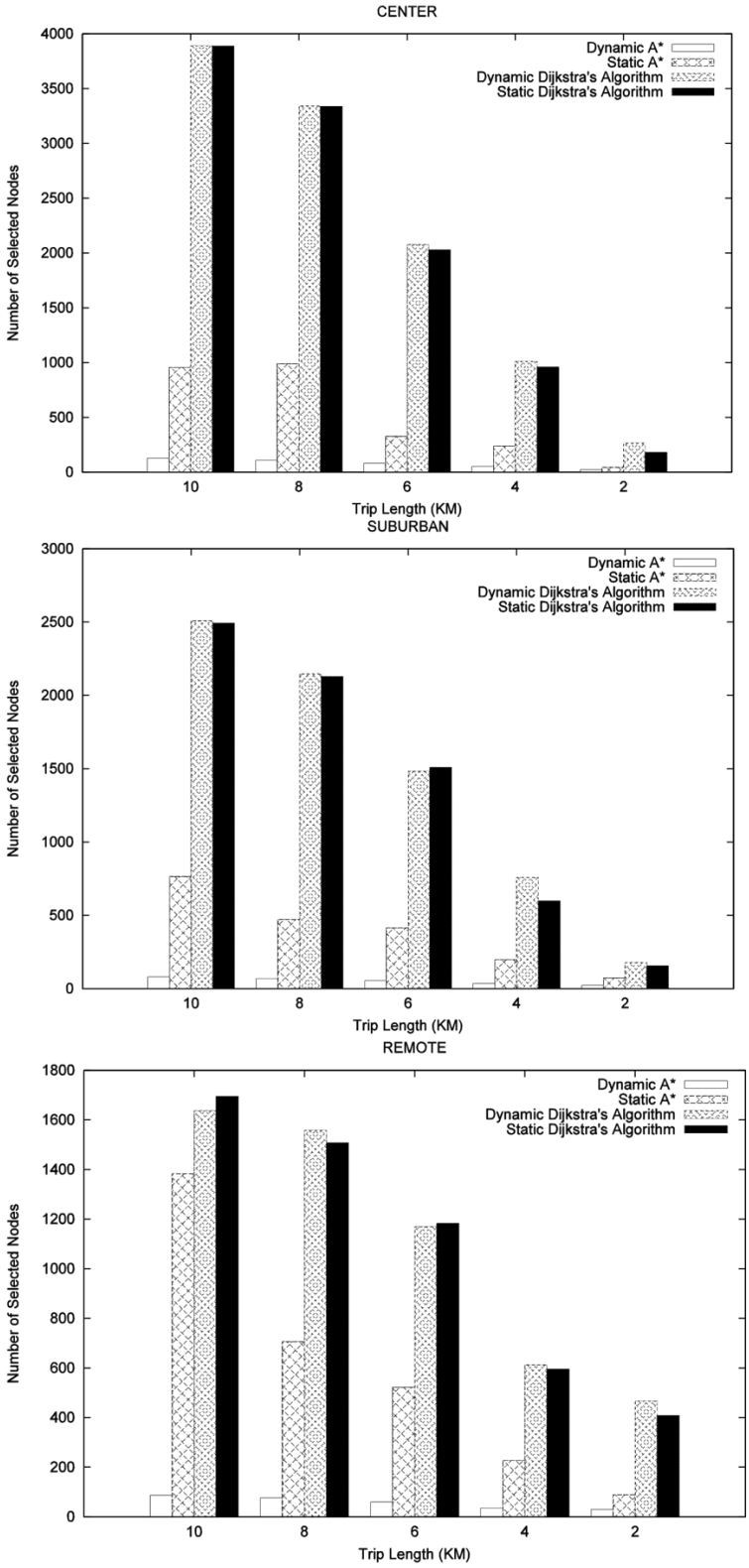


Figure 3.6: Number of selected nodes under various trip lengths

The computation time reflects the practical performance of an algorithm based on its execution time and is calculated after the prerequisite data (i.e. map and lower bounds) have been loaded into memory. As depicted in Figure 3.8, the computation time for all the algorithms is proportional to the scenario scalability level as well as the trip length. In the remote area scenario, the performance of static A* decreases sharply when the trip length is equal to or greater than 6 km. These results are mainly in line with the theoretical performance results (i.e. number of selected nodes) discussed above.

Additionally, there are three observations worth noting. First, dynamic A* outperforms other algorithms under almost all tested scenarios. It performs even better than static A* as the latter needs to calculate the lower bound, which needs time consuming operations like square and square root, during its execution, while dynamic A* loads the lower bound it needs into the memory, and just spends memory access time for the heuristic function. Second, dynamic DA always shows the worst performance and is much less effective when compared with the other three algorithms because it has no heuristic function as A* to estimate the cost, thus it has to check the travel time information from the hard disk whenever a new node is selected. Last but not least, static A* achieves the best practical performance when the trip length is less than 6km in the centre area, 4km and 2km in the suburban area, and 2km in the remote area.

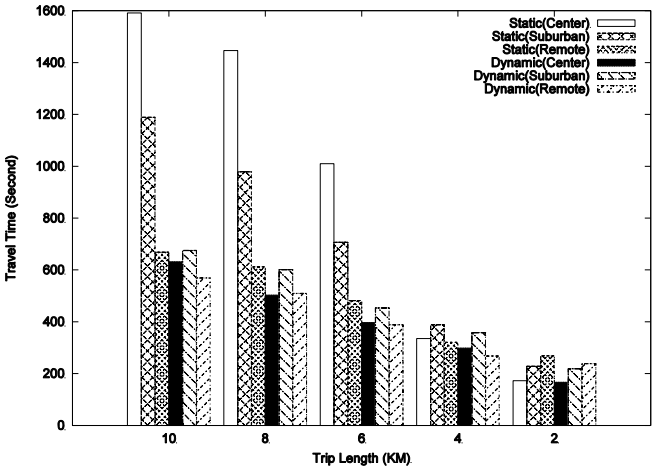


Figure 3.7: Impact of various trip lengths and urban scenarios on the efficiency of vehicle routing algorithm in terms of travel time.

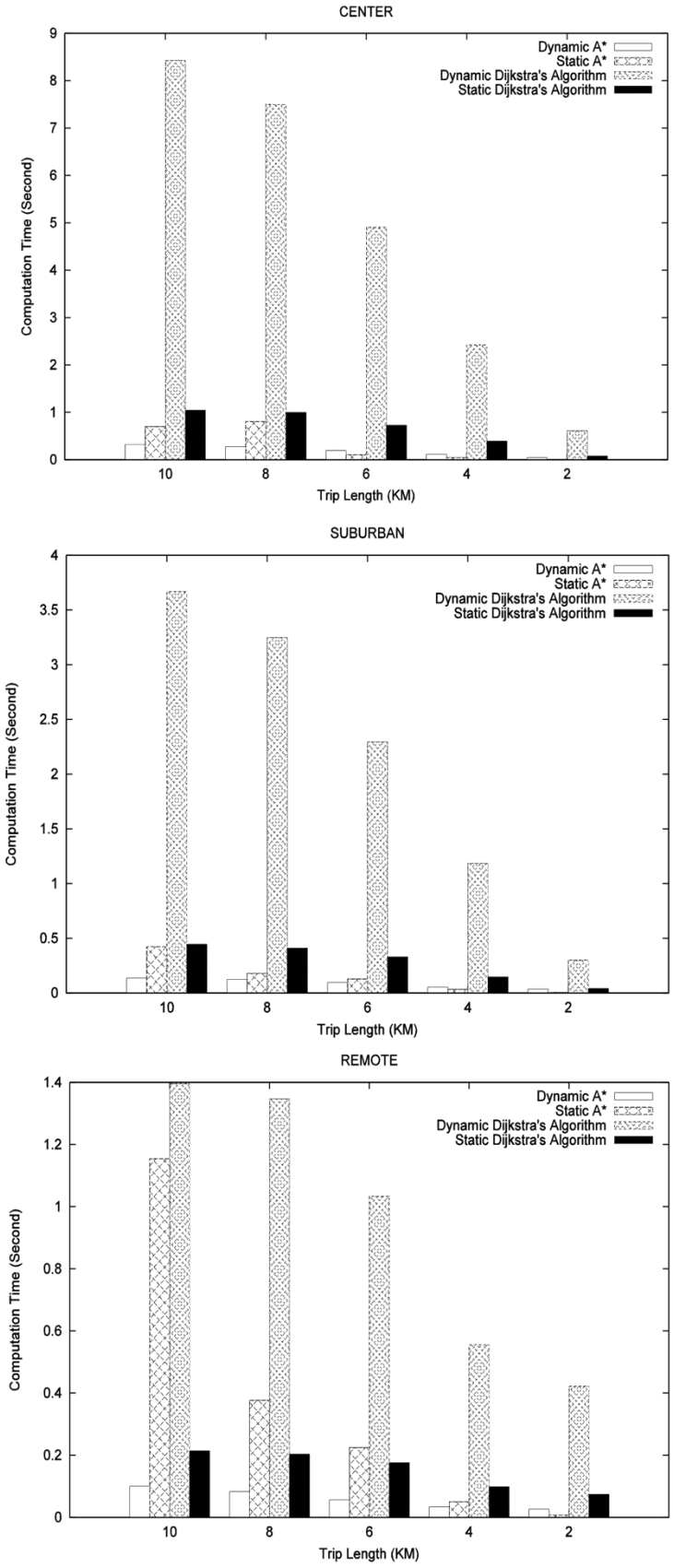


Figure 3.8: Computation time under various trip lengths.

Note that Figure 3.7 and Figure 3.9 apply for either A* or DA, as it represents a lower bound on the cost which will be estimated by A*. Therefore, in the discussion of the evaluation results for travel time and travel distance, both algorithms are considered as a whole and I just make a comparison between their static and dynamic versions.

As shown in the histogram of travel time in Figure 3.7, the results are clear for the cases when trip length is 10km, 8km, and 6km. Then, it can be concluded that for the same trip length the dynamic algorithms ensure a faster route in the remote scenario compared to suburban and centre areas. Notice that in the city centre scenario the calculated route is the slowest. On the contrary, for shorter trips length (i.e. 2km and 4km) the results are unclear for the static algorithms, as in this case the travel time of the route would be highly dependent on the road topology between the OD pairs. Although the results for the dynamic algorithms more or less have the same pattern for the trip lengths greater than or equal to 4km, the order is not as normal as I expected because they provide better routes in suburban scenario compared to the centre area scenario. Moreover, the calculated route in the remote scenario is faster than that calculated in centre scenario for trip lengths of 10km and 4km only, while very similar routes, in terms of travel time, are calculated for trip lengths of 8km and 6km. From these results it can be concluded that the dynamic algorithms can provide more stable routes, in terms of travel time, compared to the static counterpart. Finally, for short trips of 2km and 4km, all the algorithms provide very similar quality of route. Hence, in this case the simplest algorithm is suggested.

Looking at the graph of travel distance depicted in Figure 3.9, it can be seen that the static algorithms can always give the shortest route compared with the dynamic ones. However, this advantage is limited to trips of the same lengths in one specific scenario. Consequently, if the travel distance is the only metric considered for vehicle routing then any of four algorithms can satisfy the drivers' requirements. The only exception for this metric is the case of trip length of 2 km where the travel distance planned in the remote area is almost 3 times, much longer

than the other two scenarios. This is mainly due to the characteristics of the road network topology in the remote area. To overcome this issue, it is suggested that the vehicle's navigation system might recommend alternative metrics when the computed travel distance exceeds some thresholds.

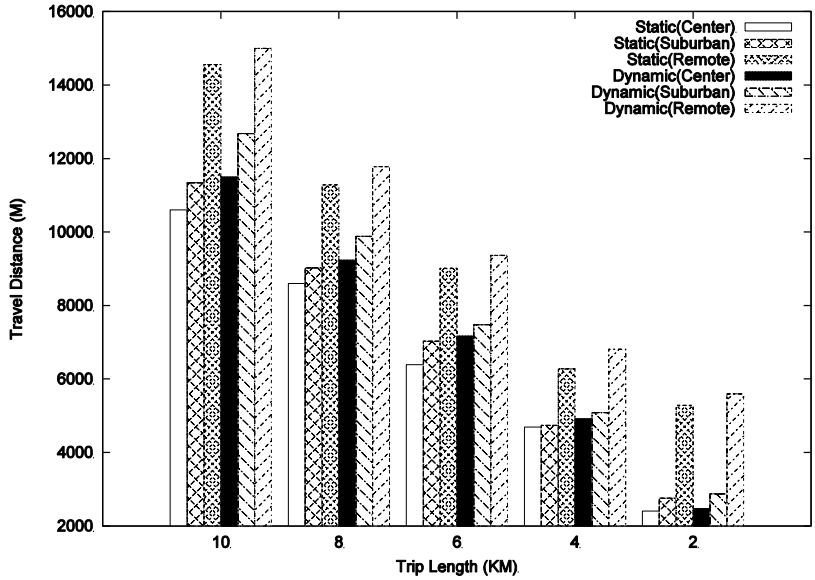


Figure 3.9: Impact of various trip lengths and urban scenarios on the efficiency of vehicle routing algorithm in terms of travel distance.

The results plotted in Figure 3.10 divulge, as expected, that the $TT_{variability}$ differs significantly in the three scenarios. For the suburban scenario, the travel time variability of the routes provided by both static and dynamic algorithms is lower than that of the routes calculated in the centre area. However, this supremacy decreases gradually when the trip length gets shorter. When the trip length drops to 2km the four algorithms show roughly the same performance.

On the other hand, for the remote scenario, the travel time variability of the routes calculated by the four algorithms is much lower (around 2500 times) than the previous case. This is due to the fact that during the period from 6:00am to 8:00am there is almost no change for the traffic flow in the remote area, as depicted in Figure 3.3. Last, for algorithm comparison, in the centre scenario, static algorithms perform slightly better than the dynamic ones, in the suburban case they

show roughly the same performance, while in remote area, dynamic algorithms are better.

It can be seen that one abnormal point exists in the remote scenario trip length of 4km. The reason is probably the extremely low change of traffic flow in the remote scenario, so the result of travel time variability would be very sensitive to the various topologies in the area between the different O/D pairs. To conclude, there is no obvious difference among the four algorithms in terms of travel time variability. Therefore, an improvement would require a new algorithm to be devised.

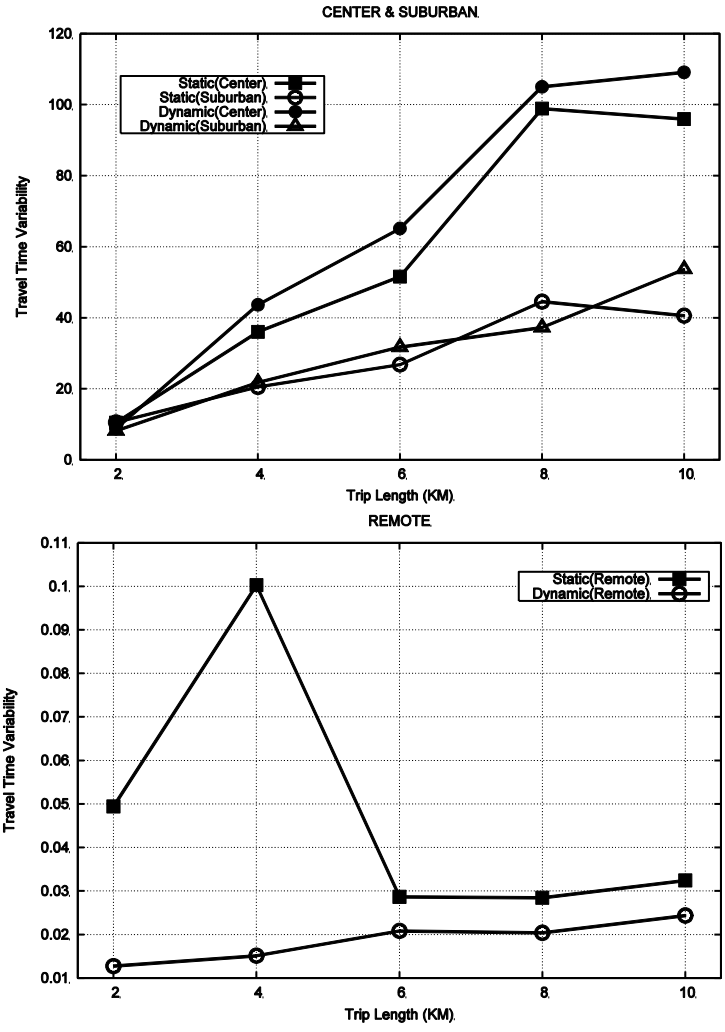


Figure 3.10: Impact of various trip lengths and urban scenarios on the efficiency of vehicle routing algorithms in terms of travel time variability

In the Table 3.3, the memory space needed by each algorithm to perform the route calculation under different scalability levels is presented. Basically, static algorithms need only to load the map data into the memory, and in my implementation this data consists of a static map "StaticMap" data in SUMO format.

In contrast, for the dynamic algorithms more data need to be loaded such as link status data "DynMap_Links", node data "DynMap_Nodes" and lower bound data "DynMap_LBs". The link data shows the different travel times on different time intervals for every link, in addition to the transportation topology data which includes node data and basic link information. The link status in the dynamic context is thus a huge volume of data where its size is “the number of time intervals” times larger than the corresponding size in the static context. In my simulation, the travel time update frequency is 30 seconds, and the simulation duration lasts 2 hours, which means that the dynamic link status data is 240 times larger than the static links data.

For dynamic A*, its advanced lower bounds need to be pre-calculated by static all-to-all DA and the results should be stored for each scenario. Afterwards, these results will be loaded to the memory to enable the execution of A* algorithm.

Table 3.3 indicates that dynamic A* needs 55.56 times more memory space than its static counterpart when the execution being performed in centre area, and even for the remote area, it still needs 126,713,008 bytes, which is 31.47 times more that static A*. This is the only one obvious disadvantage of dynamic A*.

Table 3.3: Data storage requirement for each algorithm under different urban scenarios.

	Centre area			Suburban area			Remote area		
	Static	**DynDA**	**DynA***	**Static**	**DynDA**	**DynA***	**Static**	**DynDA**	**DynA***
Static Map	7,884,103	null	null	5,455,490	null	null	4,039,884	null	null
DynMap Nodes	null	217,250	217,250	null	144,669	144,669	null	101,628	101,628
DynMap Links	null	144,139,906	144,139,906	null	96,389,386	96,389,386	null	69,029,812	69,029,812
DynMap LBs	null	null	293,667,771	null	null	120,703,021	null	null	57,581,568
Total	7,884,103	144,357,156	438,024,927	5,455,490	96,534,055	217,237,076	4,039,884	69,131,440	126,713,008

Besides the five metrics that have been discussed above, the algorithm implementation cost is another important aspect that should be taken into account to ensure a more informed decision about which algorithm to use in a centralized ITS. Since sometimes the algorithms are implemented at the hardware level which is highly dependent on the number and type of statements for the algorithm execution, a simple implementation can not only reduce the computation time but also decrease the energy consumption. Since A* has similar implementation to DA with one more heuristic function, the ranking of the implementation cost of the four algorithms studied in this work can be defined as follows:

$$Dynamic\ A^* > Dynamic\ DA > Static\ A^* > Static\ DA$$

Finally, the suggestions on the most suitable algorithm to apply in different scenarios are presented in Table 3.4. These suggestions are based on the centralized ITS architecture, in which the ITS server receives a large number of driver's requests of fastest and shortest routes. For example, when a vehicle is driving in the suburban area, and it requests the fastest route with a trip length of about 6km, then the centralised ITS will choose to run dynamic A* to response in the most efficient way.

Table 3.4: Suggestions on the most efficient vehicle routing algorithm urban different urban scenarios

	Trip length				
Fastest / Shortest	**10km**	**8km**	**6km**	**4km**	**2km**
Center area	Dynamic A*/ Static A*	Dynamic A*/ Static A*	Dynamic A*/ Static A*	Static A*/ Static DA	Static A*/ Static DA
Suburban area	Dynamic A*/ Static A*	Dynamic A*/ Static A*	Dynamic A*/ Static A*	Static A*/ Static DA	Static A*/ Static DA
Remote area	Dynamic A*/ Static A*	Dynamic A*/ Static DA	Static A*/ Static DA	Static A*/ Static DA	Static A*/ Static DA

3.4 Summary

This chapter presents a thorough performance evaluation of four vehicle routing algorithms followed by deep analysis and comparison of the obtained results. This evaluation work for both static and dynamic routing algorithms is carried out based on a realistic transportation network and highly realistic traffic load. To the best of my knowledge, such valuable performance assessment under different scalability levels and trip lengths has never been done in the literature. Moreover, the implementation cost of these algorithms and the suggested most suitable algorithm to apply in several scenarios are also discussed.

Dynamic DA has never been suggested for any scenario of practical use due to its enormous computation time. If the driver needs the shortest route, static DA is recommended for centralized ITS use in remote area due to its low complexity and good performance in terms of computation time. In the centre and suburban scenarios, static A* is a good choice for long trips (i.e. $\geq$ 6km) whereas static DA is a better alternative for short trips (i.e. $\leq$4km). For fastest route queries, dynamic A* would be highly recommended due to its low computation time and high quality of the calculated route, especially for long trips. For shorter trips, static A* is preferred as it can also provide routes with good travel time and its memory usage cost is low.

The following observations about the evaluated routing algorithms motivate my Next Road Rerouting approach to routing in the presence of en-route events. Firstly, after this evaluation study, it can be inferred that the centralized TMS cannot handle lots of routing requests within an acceptable time frame, when a non-recurrent urban congestion occurs. Secondly, this evaluation work reveals that the performance of vehicle routing algorithms varies from region to region, and is sensitive to the length of the trip. This conclusion highlights the need of designing an adaptive routing algorithm in order to achieve the high level system efficiency. Moreover, the system architecture should remain based on the typical 3-tier architecture including traffic operation centre, regional computers, and junction / road side unit controllers.

Chapter 4
Next Road Rerouting: System Architecture

This chapter is an introduction of the proposed Next Road Rerouting from a system architecture perspective. Firstly, the deployment of NRR on a typical ATCS: SCATS is overviewed. The design of the multi-agent architecture in NRR is presented in detail including the agent definition and the coordination mechanism description. The explanation of why NRR can achieve the global benefit by making use of the locally accessible information follows. In addition to the deployment details and multi-agent design, I elaborate a typical rerouting process using NRR in the face of non-recurrent traffic congestion. Finally, the concepts of centralised and distributed system design used in NRR are highlighted.

4.1 Motivation for Next Road Rerouting

Generally, traffic rerouting decisions may be classified as altruistic, where vehicle routing decisions are made to benefit the overall system, or selfish, where individual vehicles make decisions to try to optimize their own performance. While in theory global rerouting would offer the best system wide benefits, the lack of practical implementations and fairness issues make it unlikely to be adopted by users. Selfish solutions are already in use in the form of vehicle navigation systems (VNS), but these solutions suffer in terms of performance as penetration rates rise. My solution heuristically tries to balance the benefits of selfish and altruistic

solutions while mitigating the drawbacks of these solutions, that is, it is implementable, has benefits for individual users, but also seeks to balance traffic to obtain global benefits.

Altruistic routing works under the assumption that urban traffic congestion is a result of unevenly assigned traffic with respect to the capacity of existing road infrastructure [29] and hence seeks to balance the traffic load throughout the road network. Working cooperatively [81] by exchanging route choices (i.e., altruistic routing) among vehicles can lead to system optimum, in which the minimum ATT is obtained, as stated in Wardrop's second principle [82]. Although the fairness issue of system optimum solutions is addressed in [29], there are two limitations which hinder their application in the real world. Firstly, the route choice information is not always available for exchange due to privacy issues and drivers' unawareness of their full routes. Secondly, the dynamic traffic assignment for system optimum is practically intractable due to its huge complexity [40] which cannot provide real-time response to en-route events.

By contrast to altruistic routing, selfish routing is relatively easily implemented via the use of VNS. However, according to Wardrop's first principle [20], if every vehicle chooses the fastest route for itself, then a user equilibrium will eventually be reached wherein no one can unilaterally choose a faster route. This represents a local rather than global optimum, even if the user equilibrium can now be achieved in both travel time and travel time reliability [83]. Additionally, in the context of en-route events, the VNS response time might not be sufficiently responsive to allow the vehicle to avoid the impacted area.

To address the aforementioned issues with selfish and altruistic rerouting, NRR proposes a heuristically inspired two step rerouting process.

At an NRR enabled junction NRR seeks as a first step to divert vehicles around en-route events. Depending on the area of junctions enabled near the event, this will have the effect of routing the vehicle over a small number of road segments around the event. These immediate rerouting decisions are based on both global and vehicle-centric considerations, taking into account both the balancing of traffic exiting the junction (altruistic rerouting) and the impact of the diversion on the

individual vehicle's optimal route (selfish rerouting). These decisions are based on quickly calculable factors, and can be made in time to avoid the en-route event.

As a second step, while being diverted to an area beyond the influence of the en-route event, a VNS is used to propose a route from the end of the diversion to the destination. The static optimal route suggested by VNS is usually very close to the exact fastest route computed by dynamic A∗ [16] with considerable computational and storage cost [17], but still easily achieved within the time frame of traversing one or more road segment.

4.2 Deployment and Architecture of NRR

As the most widely deployed ATSC shares a similar 3-tier architecture, I take SCATS as an example to discuss the deployment details of NRR. As depicted on the left side of Figure 4.1, in the top of SCATS 3-tier architecture is the central manager located at the Traffic Operation Center (TOC). It can manage up to 64 regional computers residing in the middle tier. At the bottom tier, up to 250 intersections, where traffic lights and in-ground loop detectors are deployed, can be controlled by each regional computer. The regional computer is responsible for adjusting the scheduling and synchronization of various traffic lights' phase it controls, based on the real-time traffic information gathered from loop detectors it connects.

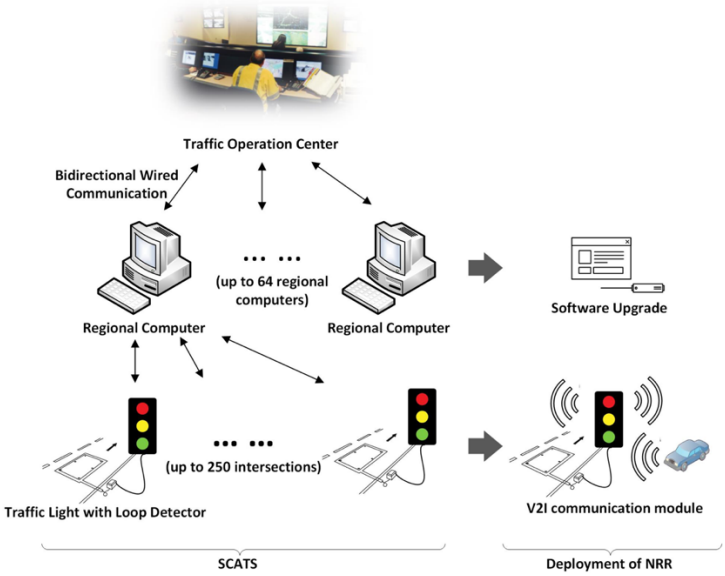


Figure 4.1: Architecture and deployment of NRR based on the existing SCATS.

As shown on the right of Figure 4.1, NRR needs only one hardware upgrade to the existing SCATS architecture (i.e., V2I communication module) at the bottom tier to enable the exchange of the information required for the rerouting process between traffic light and vehicle. As opposed to V2V communication, V2I is much less likely to suffer from non-line-of-sight communication problems, meaning that almost full communication coverage can be achieved around each intersection by avoiding signal blockage due to buildings and other obstacles. Moreover, in unexpected congestion scenarios, V2I can ensure high rate of timely and successful transmissions in the range of all the roads that each traffic light controls. Secondly, instead of deploying high-cost hardware such as a powerful road side unit, an additional feature of NRR is the low-cost software upgrade for all regional computers in order to enable the re-routing calculation and its corresponding local data management.

In practice, at each intersection the traffic lights, loop detectors combined with the regional computer controlling them are all connected with cable. This bidirectional wired communication has prompt transmission rate and fairly low loss rate. As a result, in the rest of this thesis, I consider regional computers, traffic lights and loop detectors together as one entity called *intelligent Traffic Light (iTL)*.

4.3 Overview of Rerouting Using NRR

The proposed vehicle rerouting process using NRR is presented in this sub-section along with the corresponding UML sequence diagram. As shown in Figure 4.2, when an en-route event occurs, (1) the Traffic Operation Center (TOC) verifies it and (2) notifies the iTL located at the upstream of the road where the event occurred to activate NRR by sending an emergency message. (3) This iTL broadcasts the rerouting alarm to all the vehicles in the incoming roads that it controls. (4) Those vehicles which, first, confirm that the blocked road is included in their ongoing route, then send a re-routing request which contains their destination locations, rather than the full route information which is usually inaccessible, to respond to the iTL. (5) For each rerouting request, the iTL uses the latest local traffic information gathered from induction loops, along with the local map (all outgoing roads that it controls) to compute the routing cost for each of its possible next road choices. (6) Subsequently, it suggests the one with the least cost value by sending back the rerouting result. (7) The vehicle then enters the NRR suggested optimal next road and re-computes the route for the rest of its journey with the help of its online VNS. Finally, when the event is cleared the TOC sends event dismiss to the iTL to disable NRR as described in steps (8), (9), and (10) shown in Figure 4.2.

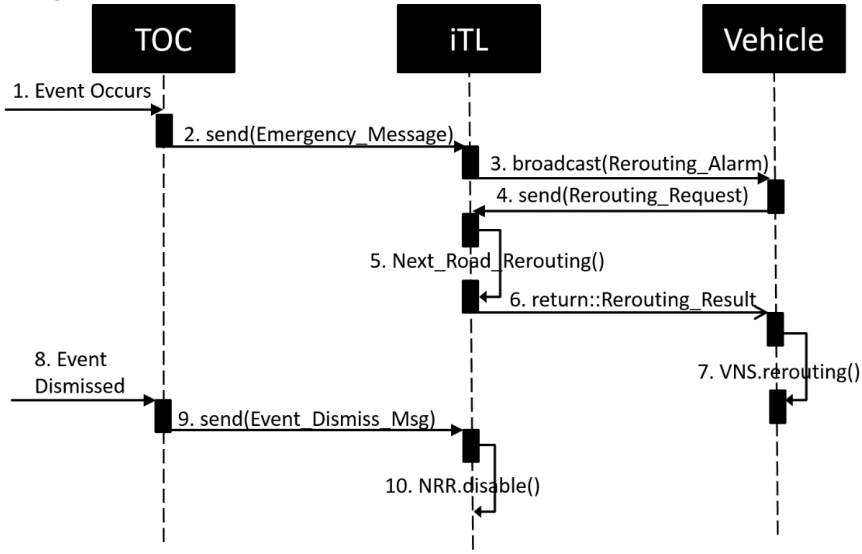


Figure 4.2: Sequence diagram of a typical re-routing process using NRR.

There are some clarifications for VNS. Firstly, similar to the regional computer in iTL that can run the computational unit of NRR, the VNS is the specific device of the vehicle that can compute the full route from a vehicle's origin to its destination. This assumes that every vehicle is equipped with VNS that are used for computing the original route when this vehicle enters the network, and rerouting in the step 7 as shown in Figure 4.2. This assumption also implies that even the drivers who are not using VNS would choose 'rational' routes that are very similarly to what would be calculated by VNS. Secondly, in step 7, the location of a closed road already known to the vehicle from the rerouting alarm received in step 3, thus the closed road would not appear in the solution provided by VNS.

In general, adapting the route of vehicles which are only one junction away from the blocked road is not enough to avoid congestion. In addition to the general ten steps mentioned above, my scalable NRR can also work in different operating levels involving more iTLs to alleviate the congestion in a wider area around the blocked road segment. As shown in Figure 4.3, I define Level 0 NRR as the NRR system with the closest iTL enabled only. Without loss of generality, Level i+1 NRR means I enable all of Level i NRR's neighboring iTLs additionally with the iTLs that are already enabled in Level i. By enabling Level i, I have access to additional road segments for the rerouting process, allowing traffic to be more evenly spread around the en-route event. To enhance the description of the NRR rerouting process, all use cases of the key actors are visualized in Figure 4.4 and the messages exchanged among them are presented in Table 4.1.

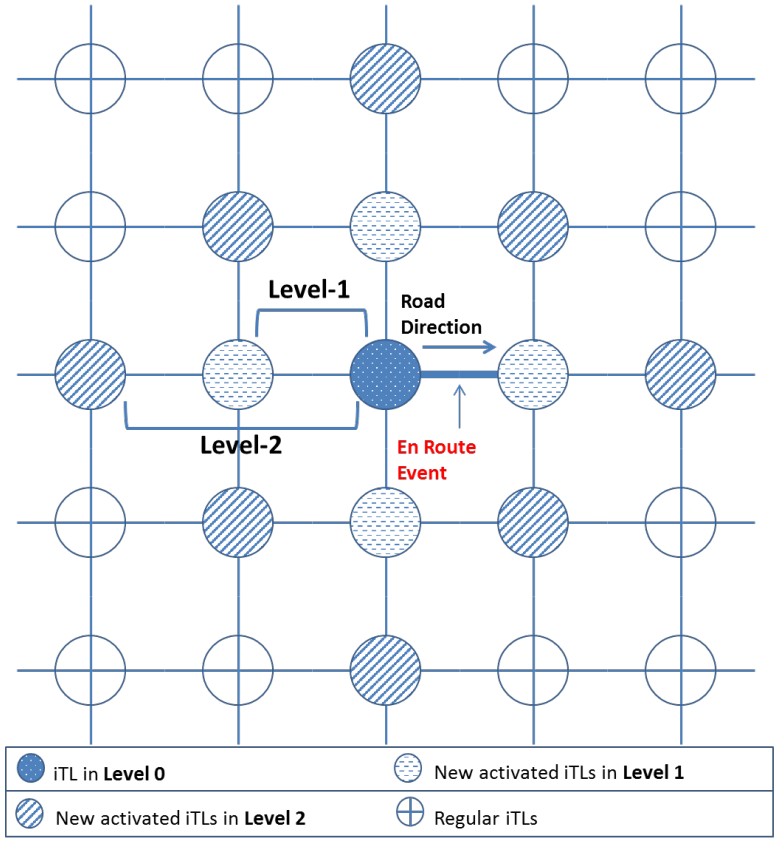


Figure 4.3: Activated iTLs in different NRR levels.

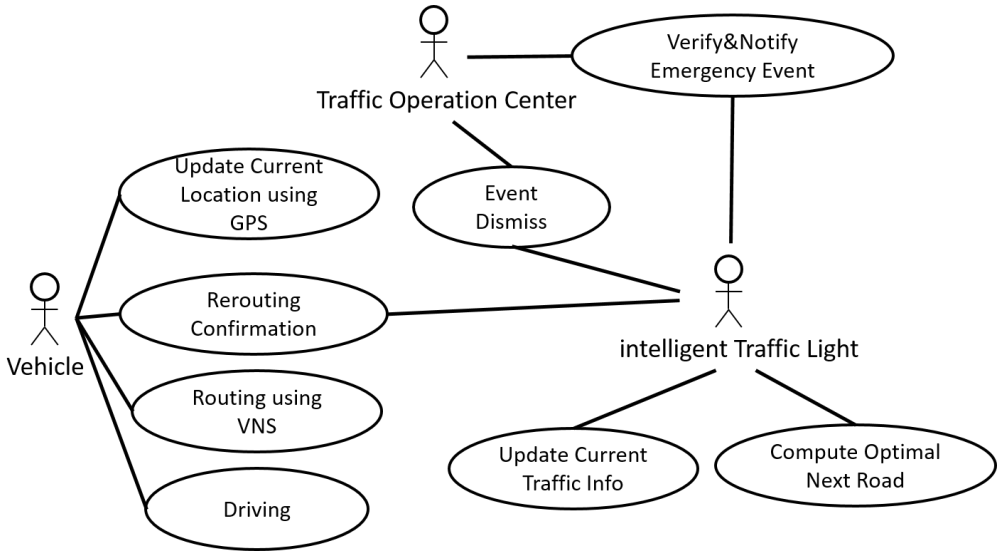


Figure 4.4: Use case diagram of all key actors in NRR.

Specifically, in Figure 4.4, the Traffic Operation Center is in charge of verifying and notifying the occurrence and completion of en-route events. It is a centralized system design as the information related to events must be broadcasted by a trustworthy third party. iTL is an important distributed intermediary between TOC and the vehicle. In general, iTL updates current traffic conditions by retrieving information from induction loops. When events occur, iTL accepts rerouting request from the vehicle and replies the suggested next road direction computed by itself. As a basic component, besides driving on the road, each vehicle is assumed to be aware its location by GPS equipment, to get a route decision for its full trip by equipped VNS, and to interact with iTL for making the next road direction choice in the face of en-route events.

Table 4.1: Summary of all messages used in NRR.

Message name	**Message content**	**Transmission Mode**	**Transmission direction**
Emergency Event	Closed Road ID, Level of NRR	Wired	TOC → iTL
Rerouting Alarm	Blocked Road ID, iTL ID	Wireless Unicasting / IEEE 802.11p	iTL → Vehicles
Rerouting Request	Destination Location, Current Location, Vehicle ID	Wireless Unicasting / IEEE 802.11p	Vehicle → iTL
Rerouting Result	Suggested Road ID, Vehicle ID	Wireless Unicasting / IEEE 802.11p	iTL → Vehicle
Event Dismiss	Released Road ID, Level of NRR	Wired	TOC → iTL

4.4 Multi-Agent System Architecture

The design of the multi-agent system in the proposed NRR makes it possible to improve the global road traffic by making decisions using locally available information. In my MAS architecture of NRR, I define an ***agent*** as a iTL deployed on each junction. The ***environment*** consists of the traffic states on all outgoing roads that the agent controls, as well as all vehicles on the incoming roads the agent monitors. The ***interactions*** between each agent and the environment in the proposed system appear in two ways as follows: the agent accesses traffic information of its outgoing roads from its neighboring agents; the agent receives and responds to rerouting requests from vehicles driving on its incoming roads. Specifically, the ***action*** that an agent can take to change the environment is to send rerouting suggestions to vehicles which are going to be stuck in the closed road. Thus, the states of the current surrounding traffic will be changed subsequently.

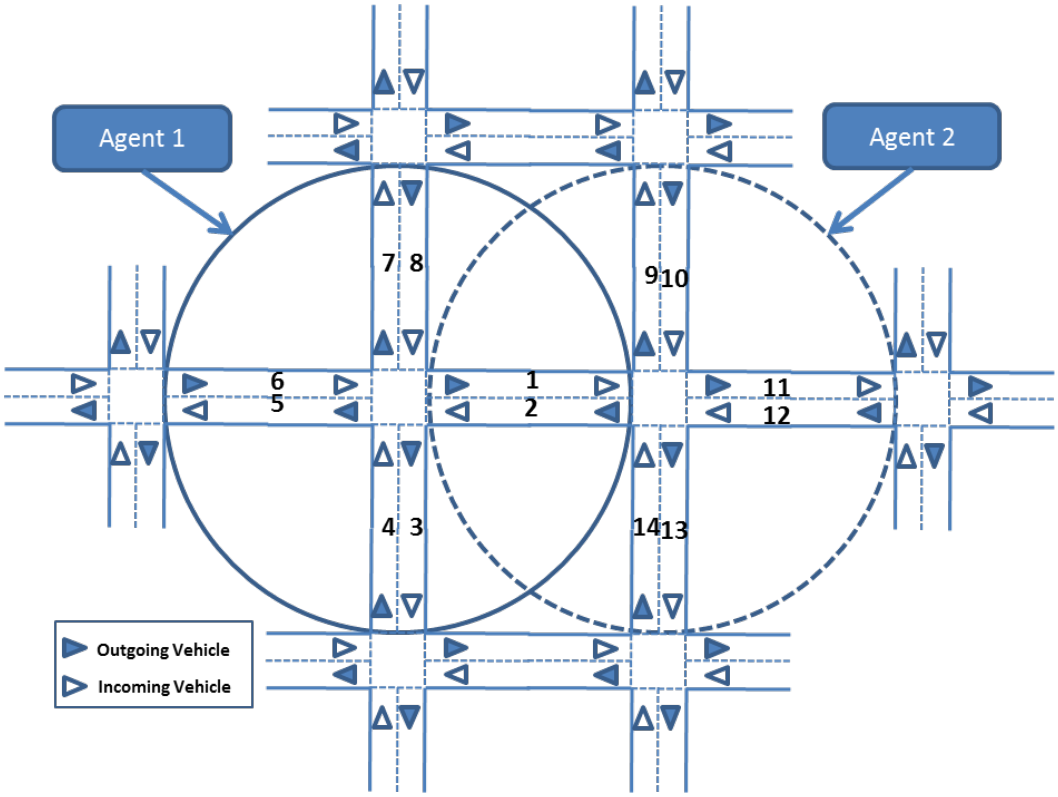


Figure 4.5: MAS architecture in NRR.

The agent ***coordination*** mechanism of NRR relies on the natural propagation of traffic. In the transportation modelling study, for the sake of simplicity, the volume-delay function or travel time function of a certain road is often assumed to be independent from the traffic on its neighboring roads. However, this is not the case in practice. Specifically, for a congested long route, if the congestion on the downstream road is released, the following roads on the upstream along the same congested route will be mitigated gradually.

As depicted in Figure 4.5, the outgoing roads of agent 1 are the lanes 1, 3, 5, and 7 which are the available options of a vehicle to be rerouted (i.e., agent's actions). This decision should be taken by collecting the current traffic information of these outgoing roads with the vehicles' re-routing requests that are received by the iTL from the incoming roads (e.g., roads 2, 4, 6, and 8 in the case of agent 1). The purpose of balancing the traffic load is to maximize the utility of the existing road infrastructure. In general, balancing the local traffic load only does not guarantee that the global traffic load will be balanced as well. NRR starts to balance

the local traffic load from the area where the stability of traffic load decreases most (i.e., where an en-route event occurred), then takes advantage of the agent's connectivity in urban road networks to propagate this mitigation effect. For instance, in Figure 4.5, when the road 3 is blocked the traffic load of all other three outgoing roads will be suddenly increased due to 1/4 loss of output under the same traffic input. NRR starts to guide the vehicles requesting re-routing to different road directions to stabilize the local traffic distribution. The key point is that each outgoing road in this agent is also an incoming road for another agent. In this case, lane 1 is an outgoing road in agent 1 but also incoming road in agent 2, thus the en-route event will soon affect the status of agent 1 and the other agents follow because the heavy traffic in lane 1 will quickly increase the traffic on lanes 9, 11 and 13 as well. If NRR is enabled for a suitable amount of surrounding agents, the traffic load will be more widely balanced, leading to an increased probability to reduce travel time for more vehicles.

4.5 Summary

In this chapter, the proposed Next Road Rerouting is discussed from the system architecture perspective. NRR implements the centralized system design to ensure the reliability of the dissemination of en-route events information. NRR also implements the responsive distributed system design using a multi-agent model based on a highly practical 3-tier architecture commonly used in the existing ACTS. Specifically, the architecture of SCATS is taken as an example to introduce the deployment of NRR and its potential cost accordingly. A rerouting process using NRR is also overviewed using a sequence diagram, use case diagram, and a table summarizing all types of required messages. More importantly, the multi-agent system architecture in NRR is discussed with the definition of each fundamental component and elaboration of the agents' coordination mechanism to reduce the non-recurrent congestion.

Chapter 5
Next Road Rerouting: Heuristic Approach

This chapter presents Next Road Rerouting mechanism by detailing its decision making process using a heuristic approach. Specifically, the four factors considered in the proposed heuristic routing cost function, when making the next road rerouting choice, are described; namely, road occupancy, estimated travel time, geographic distance to destination, and geographic closeness of congestion. The definition, motivation, and calculation of these four factors are presented respectively along with a proposed weight assignment algorithm to identify the importance of each factor in each different rerouting request. Finally, the evaluation methodology and results are presented under synthetic and quasi-realistic simulation scenarios.

5.1 Heuristic Routing Cost Function

The basic idea of proposed Next Road Rerouting in the face of en-route events is its 2-step rerouting: firstly, NRR gets a quick decision for vehicles to its next road; then, the vehicle uses the slower solution, VNS, to get a route choice decision for the rest of its trip. In step 5 of NRR rerouting process shown in Figure 4.2, the iTL will suggest the next road with the least cost for each rerouting request.

Particularly, after receiving a rerouting request from a specific vehicle ve, iTL retrieves the current location of this vehicle ($ve.curLoc$) as well as its intended destination location ($ve.destLoc$) (see Table 5.1 for key abbreviations).

Table 5.1: Key abbreviations

ve	Vehicle which sends rerouting request to iTL
$ve.curLoc$	The current location of ve
$ve.destLoc$	The destination location of ve
$ve.nrs$	The set of all available next roads for ve
$ve.nr$	The NRR suggested next road for ve
e	A certain road in $ve.nrs$
e_{cls}	The closed road
RO	Road occupancy
TT	Estimated travel time
GD	Geographic distance to destination
GC	Geographic closeness of congestion
x	A certain factor in $\{RO, TT, GD, GC\}$
$e.x$	A certain factor of e. E.g. $e.RO$ represents the road occupancy of e
CV_x	The coefficient of variation for x of $ve.nrs$
CV_{sum}	The summation of all CV_x
w_x	The weight value of x. E.g. w_{RO} represents the weight value of road occupancy
$\boldsymbol{w}$	The weight value of all factors, $\boldsymbol{w} = (w_{RO}, w_{TT}, w_{GD}, w_{GC})^{\boldsymbol{T}}$
$\boldsymbol{c_e}$	The cost of all factors for e. $\boldsymbol{c_e} = (e.RO, e.TT, e.GD, e.GC)$

Firstly, iTL uses $ve.curLoc$ and its map data to retrieve all available next roads $ve.nrs = \{e_1, e_2, \cdots, e_{N_e},\}$ (N_e: the total number of available next roads). If $N_e > 1$, then iTL should select the most suitable next road ($ve.nr$) for ve to follow.

Then, iTL measures the routing cost of each road e in $ve.nrs$ considering the weighted linear combination of the following four factors: a measure of

occupancy the new road, estimated travel time for the new road, distance to destination using the new road, and geographic closeness to the congestion using the new road. In the following, I describe how to assess each of the cost factors: *e.RO*, *e.TT*, *e.GD*, and *e.GC*.

5.1.1 Road Occupancy

Road Occupancy ($e.RO$) is measured as the percentage of time that a loop detector is occupied by a vehicle during a fixed time interval, which is commonly known as degree of saturation in SCATS [45]. It is a significant indicator showing the real time traffic load of a certain road, thus it can be used for balancing the local traffic. In this thesis, I assume that $e.RO$ can be directly retrieved by the loop detector.

5.1.2 Estimated travel time

Travel Time ($e.TT$) is the estimated mean travel time over the road e. It is the ratio of the road length ($e.len$) to the mean travel speed on this road ($e.u$). Greenshield's Model [84] is used to estimate $e.u$ because the induction loop in SCATS can only provide $e.RO$. Let us denote by $e.k$ the current traffic density (i.e., number of vehicles per km) of e and by $e.k_j$ the traffic density when traffic jam occurs on e, then basically, $\frac{e.k}{e.k_j} = \frac{\text{current number of vehicles on } e}{\text{maximum number of vehicles on } e}$) [85]. In this particular problem, only e with the minimum cost is suggested, rather than getting its accurate cost value, as $e.RO$ is proportional to the number of vehicles on e, thus $\frac{e.k}{e.k_j} \approx e.RO$, then.

$$e.TT = \frac{e.len}{e.u} = \frac{e.len}{e.u_f(1 - \frac{e.k}{e.k_j})} \approx \frac{e.len}{e.u_f(1 - e.RO)}$$

where $e.u_f$ is the free flow speed or maximum permitted speed of e. It is worth noting that $e.u_f$ and $e.len$ are static values that can be retrieved from the digital map data stored in iTL

5.1.3 Geographic Distance to Destination

Geographic Distance to destination ($e.GD$) shows how close a road e can lead ve to $ve.destLoc$. Considering the facts that the size of city center map that NRR needs to mitigate an unexpected congestion is not large (i.e., usually less than 1000 nodes, refer to Table 5.2) and its topology is almost static (i.e., rarely changes), NRR precomputes the shortest distance in kilometers for all possible origin and destination pairs using one-to-all Dijkstra's Algorithm, and loads this data to the server's memory. Thus, $e.GD$ can be accurately retrieved in a much faster way (i.e., memory access time only without any on-line computation) than applying on-line estimation using Euclidean distance. Note that the origin and destination of all trips are within the range of the road network scenario used in the simulation of this thesis. The technical details on getting the subset of a simulation scenario can be found in Appendix A.1.

5.1.4 Geographic Closeness of Congestion

Geographic Closeness of congestion ($e.GC$) shows how far one of the next road choices e can deviate ve from the closed road e_{cls}. In general, when a road is closed, the congestion level of other roads around it is increased, and the closer a road is to the blocked road, the higher its congestion level will be. This factor is expressed, as shown in Equation 5.1, by the similarity of the vector $\boldsymbol{v_e} = (e.sLoc, e.eLoc)$ from the start junction location to the end junction location of e, and the vector $\boldsymbol{v_{e_{cls}}} = (e_{cls}.sLoc, e_{cls}.eLoc)$ from the start junction location to the end junction location of e_{cls}. Notice that $\boldsymbol{v_e}$ can be obtained when iTL receives the rerouting request while $\boldsymbol{v_{e_{cls}}}$ can be retrieved when iTL verifies the reported

event in the rerouting step 2. The law of cosine [86] is used for calculating the similarity of the two vectors.

$$e.GC = similarity(\boldsymbol{v_e}, \boldsymbol{v_{e_{cls}}}) = \frac{\boldsymbol{v_e} \cdot \boldsymbol{v_{e_{cls}}}}{\|\boldsymbol{v_e}\| \|\boldsymbol{v_{e_{cls}}}\|}$$

Equation 5.1: Cosines similarity to calculate geographic closeness of congestion.

So far, NRR can construct the cost vector $\boldsymbol{c_e} = (e.RO, e.TT, e.GD, e.GC)$ for each possible next road e. It is worth to mention that lower values of the above four factors lead to a better rerouting for ve. Given a specific weight assignment vector for the aforementioned four factors $\boldsymbol{w} = (w_{RO}, w_{TT}, w_{GD}, w_{GC})^T$, the NRR suggested next road for ve is the one with the least value of cost function $\hat{\boldsymbol{c}}_{\boldsymbol{e}} \cdot \boldsymbol{w}$ as shown in Equation 5.2

$$ve.nr = \underset{e}{argmin}\, \hat{\boldsymbol{c}}_{\boldsymbol{e}} \cdot \boldsymbol{w}$$

Equation 5.2: Heuristic routing cost function.

where $\hat{\boldsymbol{c}}_{\boldsymbol{e}}$ is the normalized $\boldsymbol{c_e}$ with each of its element $e.x$ scaled in the range [0,1] using Equation 5.3

$$e.\hat{x} = \frac{e.x - e.x_{min}}{e.x_{max} - e.x_{min}}$$

Equation 5.3: Normalization for each factor.

where $e.x_{min} = \min(\{e.x, e \in ve.nrs\})$, $e.x_{max} = \max(\{e.x, e \in ve.nrs\})$

5.1.5 Adaptive Weight Assignment Approach

Through identifying the importance of each of these four factors, the system will be able to assign the most suitable weight value to each of them to compute the final routing decision. In NRR, the values of the factors used in the next road cost function vary depending on the different time stamp (i.e., $e.RO$, $e.TT$) and different current/destination location of the vehicle to be rerouted (i.e., $e.GD$, $e.GC$). Therefore, a suitable weight value allocation $\boldsymbol{w}$ should be variable for different rerouting requests [87]. In the next road selection, for a particular factor over all next road choices, the greater the variation of its value is, the more importance is given to it in the computation of the rerouting decision. Indeed, the

next road choice with the least cost value of this factor represents a substantial gain compared to other road choices which have higher values of this factor. Since all factors represent different measurements, the coefficient of variation (CV) is used instead of standard deviation to compute the variability for each factor. Specifically, as shown in the following equations. iTL calculates CV for each factor $x \in \{RO, TT, GD, GC\}$ over all available next roads, then, it gets summation of all factors. Finally, the weight value of x is its corresponding proportion to CV_{sum}.

$$CV_x = CV(e_1.x, e_2.x, \cdots, e_{N_e}.x)$$
$$CV_{sum} = \sum CV_x$$
$$w_x = \frac{CV_x}{CV_{sum}}$$

Equation 5.4: Weight allocation using coefficient of variation.

In the example shown in Figure 5.1, when a vehicle is approaching a junction, it has three road choices to follow: r_1, r_2, and r_3. To calculate the road occupancy RO factor, I assume that all vehicles have the same length (4.5 meters) and the same minimum gap with each other (2.0 meters). By knowing the actual length of those three roads, RO for all roads is calculated as

$$RO_1 = \frac{1 \times 6.5}{80.0} = 8.125\%$$
$$RO_2 = \frac{2 \times 6.5}{30.0} = 43.33\%$$
$$RO_3 = \frac{4 \times 6.5}{80.0} = 32.5\%$$

In this example, I simplify the calculation of the second factor (i.e. estimated travel time TT) as the ratio of the road length to its average instantaneous travelling speed. When there is no vehicle running on this road, the average speed is replaced by the maximum allowed speed in this calculation. In this case, the calculations are as follows:

$$TT_1 = \frac{80.0}{11.0} = 7.27s$$

$$TT_2 = \frac{30.0}{(10.1 + 9.3)/2} = 3.09s$$

$$TT_3 = \frac{80.0}{(3.7 + 3.7 + 3.9 + 3.5)/4} = 21.62s$$

The third factor is the geographic distance to destination GD. As described ealier, the value of this factor is directly retrieved from the pre-loaded memory. Thus, I just give these three values as:

$$GD_1 = 1300m, GD_2 = 900m, GD_3 = 600m$$

The coefficient of variation CV is the ratio of standard deviation to the mean value. In this case, I get the following CV s for all three factors[12]:

$$CV(RO) = 0.53, CV(TT) = 0.74, CV(GD) = 0.31,$$

Their summation is 1.58. Then I get the following weight allocation:

$$w_{RO} = \frac{CV(RO)}{1.58} = 0.333$$

$$w_{TT} = \frac{CV(TT)}{1.58} = 0.472$$

$$w_{TD} = \frac{CV(TD)}{1.58} = 0.195$$

Notice that the summation of these weight values should equal to 1.

[12] For the sake of simplicity in this example, I avoid the calculation of geographic closeness to congestion as it is relatively complicate.

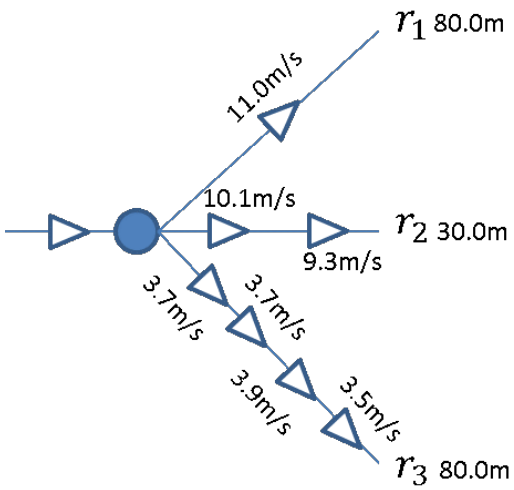


Figure 5.1: An example of weight values allocation calculation in NRR.

5.2 Evaluation Methodology

5.2.1 Simulation Settings

The version (0.24.0) of Simulation of Urban Mobility (SUMO) [41] combined with the Traffic Control Interface (TraCI) [79] is the simulation platform used to carry out the performance evaluation of NRR.

The evaluation of NRR is carried out in both realistic and synthetic scenarios. A sub-set of TAPASCologne 0.17.0 [78] is chosen as a realistic evaluation scenario for NRR. TAPASCologne is an open source project providing a large-scale dataset with the highest realism for urban vehicular simulation based on SUMO. It uses a realistic map of Cologne extracted from OpenStreetMap and generates traffic demand from 6:00 am to 8:00 am using Travel and Activity PAtterns Simulation (TAPAS) methodology [89] and Gawron's [40] traffic assignment algorithm .

A subset only (i.e. different from three subset maps shown in Chapter 3) of this map is used in my evaluation because the original size of TAPASCologne is

too large (1129.71 km2) to investigate the impact of a single closed road. The chosen sub map is a 3.69 km2 large area located on the west of the river in the Cologne city center. The first 30 min of original traffic of this sub-map, ranging from 6:00 am to 6:30 am is used for NRR evaluation.

Table 5.2: Simulation scenarios statistics.

	Cologne_center	**8 x 7**
#Juncitons	389	86
#Roads	737	254
#Roads / #Junctions	1.89	2.95
Average Road Length (m)	93.20	115.80
Covered Area (km^2)	3.69	1.22
Total Lane Length (km)	95.15	58.83
Traffic (#vehicles)	7665	2942
Traffic Density (#vehicles/km/lane/hour)	96.86	100

Even though a realistic map can provide trustworthy evaluation results, the great diversity of urban road network topologies may lead to a significant difference in the corresponding NRR evaluation results. In order to mitigate this impact, in my evaluation, I generated grid maps. Due to the limited rerouting choices of small grid maps and the large observation area for studying the impact of closing one road in a big grid map, the 8 × 7 map (i.e., 8 intersections in the horizontal axis and 7 intersections in the vertical axis) is chosen as a representative grid map for the following evaluations. Apart from the number of junctions, they share all the rest of settings, e.g., all road segments in this grid map set have equal length of about 120 meters. Each road segment comprises of two roads each of which has two lanes (i.e., mimic main urban roads) in the same direction.

For the 8 × 7 grid map testbed, 30 minutes traffic demand is generated evenly according to the road length and the number of lanes for each road. Three key parameters in this random generation process are chosen to ensure that the synthetic scenario can still simulate the city center scenario in peak hours traffic. First, the

repetition rate is the amount of time in seconds between vehicles insertion over the whole network. Its value varies across all grid map scales to maintain the consistency of the traffic density with that of the city center of Cologne, which is about 100 vehicles per km per lane per hour (see Table 5.2). Second, the minimum trip distance is set to twice the average road length because a meaningful route in this study should have at least two consecutive roads. Last but not least, the fringe factor is set to 10, which means edges that have no successor or predecessor will be 10 times more likely to be chosen as start or endpoint of a trip. This allows us to model through-traffic which starts and ends outside of the simulated area. The setting of traffic lights is also set to static, meaning that every traffic light has a fixed phase duration regardless of the changes in traffic conditions.

It is worth emphasizing that to make these synthesis maps capable of simulating a realistic urban road network, the three configuration parameters (i.e., the ratio between number of roads to junctions (#R/#J), the average road length, as well as the traffic density outlined in Table 5.2) should be in line with their corresponding values in the city center of Cologne.

For both scenarios, grid map and city center of Cologne, the whole simulation keeps running until all the vehicles finish their trips. Therefore, the full simulation time is longer than the predefined 30 mins trip generation time.

5.2.2 Evaluation Metrics

- **Travel Time:** Also called trip time in this thesis, is the amount of time a specific vehicle needs to finish its trip. It is calculated as the sum of the travel time this vehicle spends on each individual road along its route.
- **Free-Flow Travel Time**: Free-flow travel time for a specific road is the amount of time a vehicle needs to traverse it at the maximum-allowed speed on this road.
- **Average Travel Time (ATT)**: Average travel time is a mean value of the travel time of all vehicles' trips. It indicates the overall status of traffic for the whole observed road network.

- **Travel Time Index (TTI)**: Also called congestion index, is a commonly used metric for measuring urban traffic congestion level [2]. It is calculated as the ratio of the sum of the travel time to the sum of the free-flow travel time for all vehicles. This metric is more meaningful than the average travel time because it gives a measure of the proportional increase over the ideal.
- **Travel Time Reliability**: This concept refers to the unpredictability of travel time. For drivers it can give some measure of likely worst case delay [90]. The focus of this thesis is on the travel time reliability for the whole set of trips instead of a single trip only.
- **Planning Time Index (PTI)**: In practice, travel time reliability is measured by the planning time index [90]. In order to keep consistency with TTI, for all trips as a whole, PTI is calculated as the ratio of the 95th percentile travel time (i.e., which is shorter than 5% of all trips) to the average free-flow travel time.
- **System Instability (SI)**: System instability is a metric that I introduce to describe the variation of traffic load distribution over the whole simulation duration and road network. Given the set of discrete time intervals of a simulation duration $T = \{T_1, T_2, \cdots, T_n\}$ and the set of all roads in the simulated road network $E = \{e_1, e_2, \cdots, e_n\}$.

$$SI = \sigma(\sigma(e.RO_t, e \in E), t \in T)$$

where σ means the computation of standard deviation, $e.RO_t$ means the occupancy of road e at the time interval t. When the value of SI is low, the system is described as stable which represents that the traffic load is more or less evenly distributed on all roads. Note that both non-congested and fully congested road networks will result in low SI. In these cases, further rerouting is not necessary or helpful, as the existing road capacity is already well used. A high value for SI indicates that further rerouting may be of benefit, as the traffic is unevenly distributed.

5.3 Evaluation Results and Analysis

In the following I first explore the impact of purely altruistic and selfish routing strategies on traffic performance in the presence of en-route events. The benefits and disadvantages of these strategies are illustrated through simulations using a grid map. It should be noted, however, that implementations of altruistic strategies do not exist in practice. Thus when evaluating the performance of my NRR routing policy I compare it to two commonly used selfish rerouting strategies. These comparisons are made both for a grid map and a subset of the city centre of Cologne.

5.3.1 Impact of Selfish and Altruistic Rerouting on Traffic Conditions

I have evaluated 4 scenarios, as described below, and compared their results against each other:

- Original (**ORG**): The original scenario with the initial 30 minutes traffic demand, as described previously in the simulation setting section, without any closed road or any particular dynamic routing strategies applied. The routes for all vehicles are generated before the simulation using Gawron's traffic assignment algorithm.
- En Route Event (**ERE**): The ORG scenario with two roads of one road segment in the center of the map (as shown in Figure 5.2) closed for 20 minutes (from the 5th min to the 25th min). I set the maximum allowed speed for the closed road to 0.1 m/s to mimic the road closure, which is a commonly suggested technique in the SUMO community. The closed road also lies in the center of geographical traffic distribution in the map.
- Constant Rerouting (**ConRe**): This scenario represents selfish rerouting. Here, upon encountering an en-route event, vehicles update their fastest route according to up to date traffic information.

- Load Balance Rerouting (**LoaRe**): I choose this scenario to represent altruistic rerouting which focuses on balancing local traffic without considering the destinations of individual vehicles. In this scenario, when encountering an en-route event, vehicles update their next road choice according to current local traffic, choosing the road with the lowest occupancy level. The sacrifice is that it is highly likely for an individual vehicle to be diverted further and further away from their ideal route.

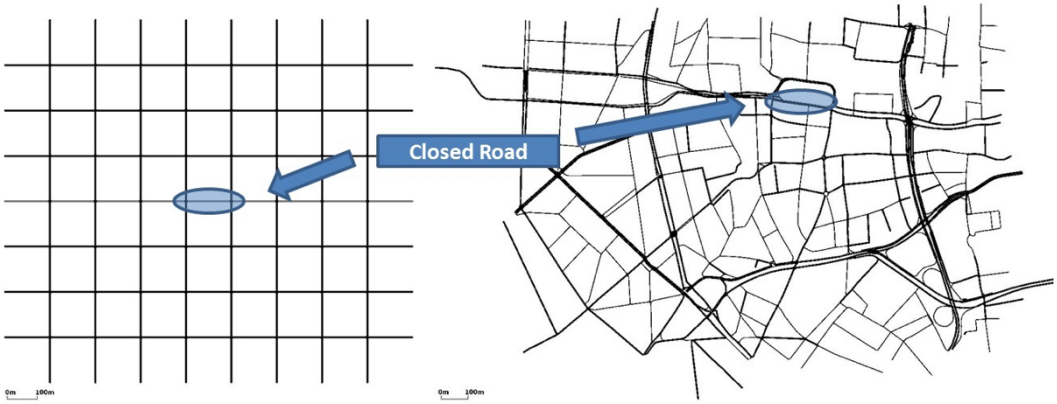


Figure 5.2: Location of the closed road in grid map (left, 8X7) and realistic map (right, city center of Cologne).

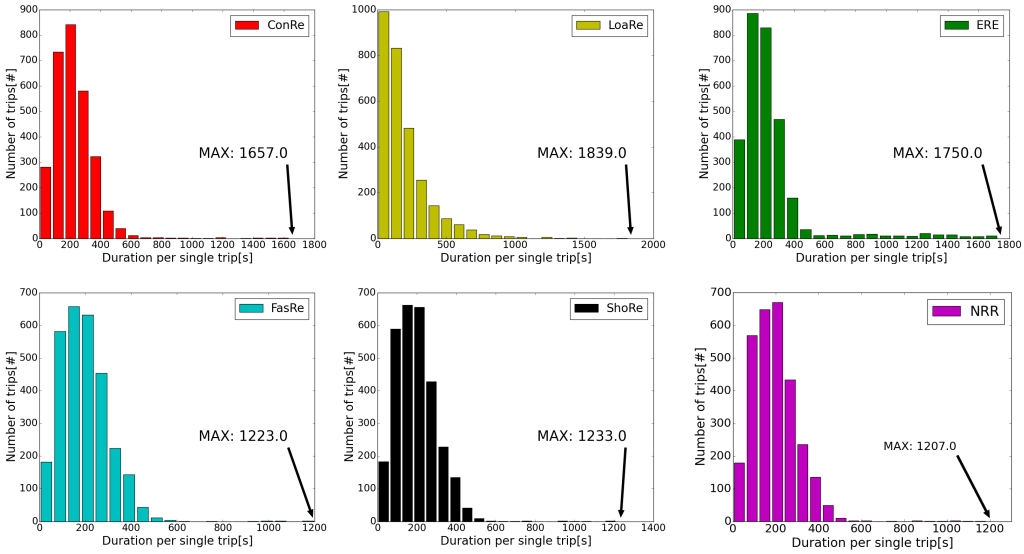
Table 5.3 summarizes the performance metrics (Average Travel Time, Travel Time Index, 95th Percentile Travel Time, Planning Time Index, and System Instability) for each of the four above scenarios. I observe that in ERE scenario, compared to ORG, 2 closed roads only, representing 0.79% of the total road capacity in the map, can bring a significant negative impact even on those vehicles running through the other 252 open roads. This table reveals as well that the Average Travel Time (ATT) has increased by 28.94%, in addition to an 80.99% rise in Planning Time Index (PTI), which means that the trip time becomes extremely unreliable. Moreover, the considerable growth of system instability up to 123.21% is also in line with the degradation of travel time reliability.

Table 5.3: Performance comparison of ConRe and LoaRe against ORG and ERE in 8x7 grid map.

	Average Travel Time (sec)	**Travel Time Index (TTI)**	**95th Percentile Travel Time (sec)**	**Planning Time Index (PTI)**	**System Instability**
ORG	207.55	2.79	375.95	5.05	0.56
ERE	267.61	3.40	719.75	9.14	1.25
ConRe	246.42	2.96	446.95	5.37	0.61
LoaRe	212.99	2.82	573.0	7.59	0.45

Compared to ERE scenario, both ConRe and LoaRe can mitigate the unexpected traffic congestion in terms of the achieved ATT and trip time reliability. However, the 7.92% reduction of ATT that ConRe brings is much less than 20.41% that LoaRe does. This is due to the exceptionally good system stability achieved by the latter, which is even 19.64% better than the original scenario, whereas the former is 8.93% worse than the ORG case in terms of the achieved stability.

(a) 8 × 7 grid map



(b) city center of Cologne

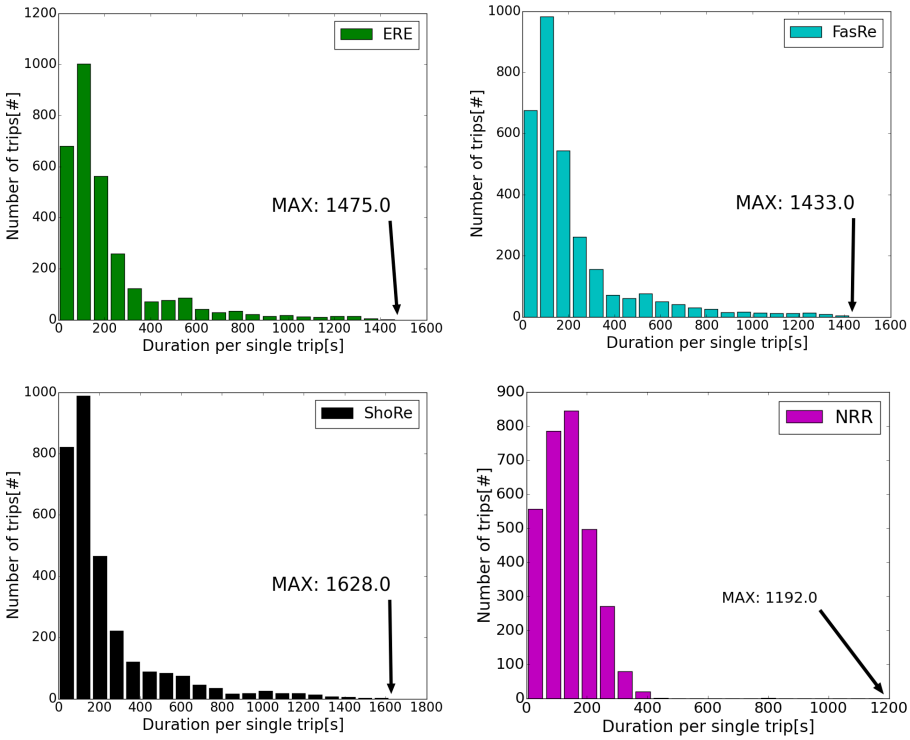


Figure 5.3: Trip duration distribution of the evaluated scenarios in both 8 × 7 grid map (a) and city center of Cologne (b).

On the other hand, as a consequence of omitting the vehicle's destination location, when LoaRe is applied, there are a few vehicles which have much longer

travel time than the average. Correspondingly, the trip duration distribution shown in Figure 5.3 reveals that LoaRe has a significantly longer right tail than ConRe. Thus LoaRe shows a much lower trip time reliability performance improvement (i.e., 16.96% only, as compared to ConRe's 41.25% of improvement) and causes serious fairness issues for a certain number of vehicles.

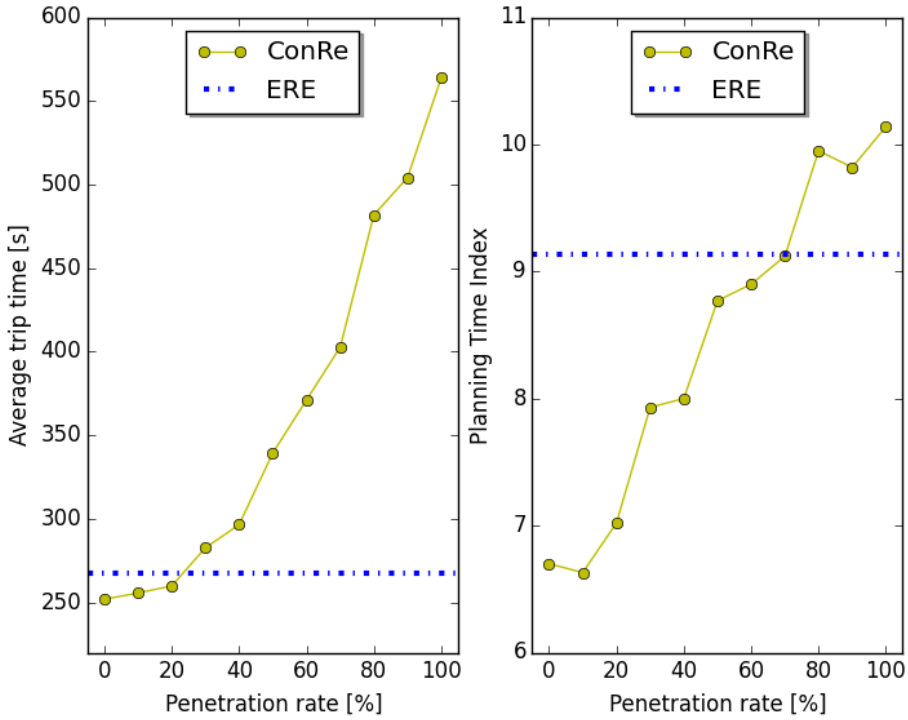


Figure 5.4: Impact of the penetration rate on the performance of ConRe.

In these tests, the routing algorithm is only invoked upon encountering an en-route event. Thus, only a small number of cars use the algorithm. In the final test, I explore the consequence of increased use of the ConRe algorithm. In particular, I modify the ORG scenario so that a certain percentage of cars recalculate their route once every second. Figure 5.4 indicates the impact of penetration rate (percentage of cars employing the strategy) on Average Trip Time and Planning Time. Clearly increasing the number of vehicles using selfish rerouting has a very negative impact on performance. This is consistent with the results in [85] and in line with Braess's paradox [37].

In summary, even a small portion of roads closed in the center of a road network, can cause a substantial degradation of traffic conditions. However, neither selfish rerouting nor altruistic rerouting is suitable for improving both average trip time and trip time reliability when such events occur, especially under higher penetration rates. In the following I will demonstrate the benefits of my proposed NRR policy vs. commonly available selfish solutions.

5.3.2 Investigating NRR's Scalability

As discussed in Chapter 4, NRR has multi-level options, i.e., the higher the level the traffic manager chooses, the more junctions with NRR-enabled iTLs around the closed road will be activated to run NRR. To find the best scalability level of NRR, I have evaluated its performance using 8 × 7 grid maps from Level 0 to Level 4. Compared to Level 0 NRR, the reduction of ATT and 95th percentile trip time (expressed in percentage) achieved by NRR in all other higher levels are shown in Table 5.4.

Table 5.4: Performance of NRR under different scalability levels in 8x7 grid map.

NRR level	**L0**	**L1**	**L2**	**L3**	**L4**
# enabled iTL	2	8	18	32	44
ATT	218.60	216.09	216.26	213.53	212.88
Percentage of ATT reduction to L0 (%)	0	1.15	1.07	2.32	2.62
95th Percentile Travel Time (PTT)	403.0	396.0	397.0	387.95	380.0
Percentage of 95th PTT reduction to L0 (%)	0	1.74	1.49	3.73	5.71

One important conclusion that can be drawn from this table is that the upgrade from Level 0 to Level 1 brings enough performance enhancement while upgrades to Level 2, Level 3 and even Level 4 bring only minor additional improvements. In order to minimize operational costs (i.e., the number of NRR enabled iTLs), I suggest implementation of Level 1 NRR only.

5.3.3 NRR vs. The Existing Solutions

To show the performance gain when applying NRR, the two most commonly used solutions in practice, namely Fastest Rerouting and Shortest Rerouting, are implemented in this evaluation.

- Fastest rerouting (**FasRe**): During the road closure time period in ERE scenario, all vehicles that have the closed road included in their ongoing routes, reroute once according to global traffic information. This scenario aims to mimic the fastest route that existing VNS can provide. When a driver is notified about an event ahead, this common solution uses the on-vehicle navigation system again based on the latest global traffic information, excluding the closed road from the rerouting result since it will appear as a bottleneck.
- Shortest rerouting (**ShoRe**): During the road closure time period in ERE scenario, all vehicles that have the closed road included in their ongoing routes, reroute once only based on the length of roads. This scenario mimics the shortest route that existing VNS can provide. In practice, the drivers are usually notified about an en-route event only one junction away from the location where it has occurred. This notification can be either through temporary road signs, or the observations of the drivers of deteriorating road conditions. Therefore, in my simulation, FasRe and ShoRe are implemented as Level 0 rerouting strategies. Notice that when the traffic congestion propagates back further than one link from the closed link, I assume that the drivers are waiting in the congestion queue rather than rerouting themselves. This is because drivers usually tend to follow their pre-selected route with more patience, and tend to act only when they become aware of the en-route event typically one junction ahead.
- **NRR**: During the road closure time period in ERE scenario, my proposed Level 1 NRR is enabled for congestion avoidance.

Table 5.5 compares the performance of the algorithms for the grid topology and city center of Cologne respectively. I discuss the performance according to the performance parameters of travel time index, 95th percentile travel time, planning time index and system instability.

Table 5.5: Performance comparison of NRR, ShoRe, and FasRe with ORG and ERE scenarios (Cologne Center / 8x7).

	Average Travel Time (sec)	**Travel Time Index (TTI)**	**95th Percentile Travel Time (sec)**	**Planning Time Index (PTI)**	**System Instability**	**Total Travel Length (km)**
ORG	140.09 / 207.55	1.40 / 2.79	269.75 / 375.95	2.70 / 5.05	0.74 / 0.56	4483.79 / 2719.31
ERE	214.88 / 267.61	2.11 / 3.40	705.50 / 719.75	6.91 / 9.14	3.32 / 1.25	4483.79 / 2719.31
FasRe	216.10 / 218.39	2.12 / 2.83	711.75 / 403.95	6.97 / 5.24	3.34 / 0.69	4486.05 / 2741.18
ShoRe	227.69 / 218.15	2.25 / 2.83	746.75 / 400.95	7.37 / 5.21	3.43 / 0.66	4485.55 / 2735.48
NRR	145.98 / 216.09	1.42 / 2.79	292.0 / 396.0	2.85 / 5.12	0.81 / 0.63	4571.11 / 2873.25

Travel time: In terms of the reduction of the ATT, according to the evaluation results shown in Table 5.5, Level 1 NRR shows the best performance compared to ShoRe and FasRe. More precisely, in 8 × 7 grid map, NRR decreases the ATT by 19.25% compared to ERE, while this improvement is limited to 18.48% for ShoRe and 18.39% for FasRe. Although the advantage NRR brings is relatively marginal, less than 1% compared to ShoRe and FasRe, in realistic scenario (i.e.,

city center of Cologne) this advantage becomes a much more significant 32.06%, with ShoRe and FasRe perform even worse than ERE by 5.96% and 0.57% respectively. Similar conclusions can be drawn regarding the achieved TTI.

According to the trip distribution statistics plotted in Figure 5.3, in both grid and city center of Cologne maps, NRR still has a long right tail similar to that of ERE, ShoRe and FasRe, due to the fact that there have been always a few vehicles already stuck in the closed road before any rerouting strategy is applied. Thus, their trip time will be severely affected but for most of the other vehicles NRR successfully moves the trip time distribution to the left in Figure 5.3, saving more time for more trips compared to other rerouting strategies.

Travel time reliability: In terms of PTI reduction for both maps, Level 1 NRR performs the best among ShoRe and FasRe, and shows higher gain compared to that shown by ATT evaluation metric. Specifically, in 8 × 7 grid map, NRR performs 43.98% better than ERE, while ShoRe and FasRe outperform the latter by 43.00% and 42.67% respectively. In realistic scenario, NRR maintains this advantage by 58.76% compared to ERE, while, similar to ATT, ShoRe and FasRe even perform 6.66% and 0.87% worse than ERE.

All solutions perform worse in city center of Cologne than in the grid map. A reasonable explanation is that compared to 8 × 7 grid map, city center of Cologne scenario has almost 3 times more vehicles and larger areas, and there is only one road segment closed for both scenarios. Hence, as opposed to 8 × 7 grid map scenario, there are a lot more vehicles in city center of Cologne which are not or only slightly affected by the en-route event but still being counted in the overall simulation results.

Due to many direction-changing restrictions in realistic urban roads (i.e., one-way road, prohibited left/right turn), as well as the limited scalability of the two compared solutions (i.e., Level 0), ShoRe and FasRe always have much less rerouting choices than NRR, therefore, they tend to give the same rerouting direction to a higher percentage of vehicles, leading to more congested roads. This is the reason why ShoRe and FasRe performs even worse than ERE in which no

rerouting strategy is applied, apart from the previously discussed limitations of selfish rerouting.

Other evaluation metrics: From the evaluation results of system instability I observe that NRR can also balance the traffic load on the roads better than FasRe and ShoRe. Additionally, the notable traffic improvement NRR brings is not a result of diverting event-affected vehicles to a much longer route which is usually not preferred by the drivers. There are only marginal differences among NRR, FasRe and ShoRe in terms of total travel length, maximally 5.04% in grid map and 1.91% in realistic map, nevertheless, the considerable variations of performance gain among them compared to ERE can go up to 32.06% in ATT gain and 58.76% in PTI gain in realistic scenario (see Figure 5.5).

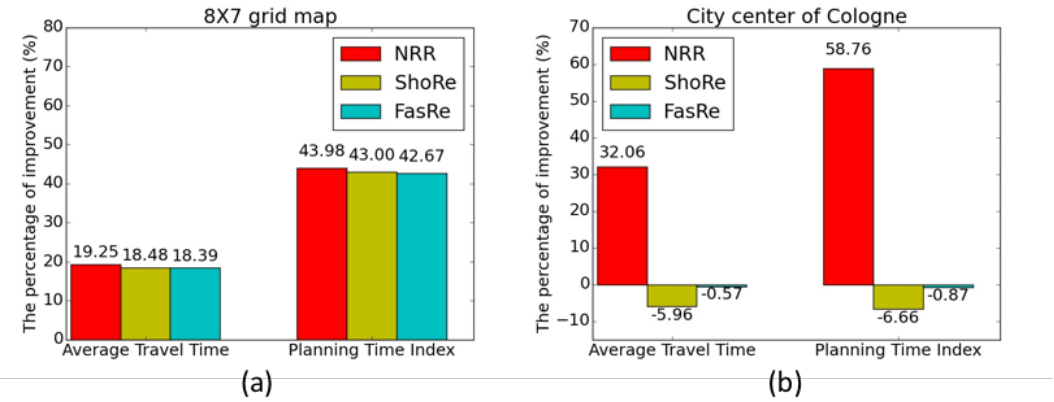


Figure 5.5: Comparison of the percentage of improvement achieved by NRR, ShoRe and FasRe over ERE in terms of ATT and PTI.

5.3.4 Study of the Impact of NRR on both Rerouted and Non-Rerouted Vehicles

The previous results assess the impact of the strategies on all vehicles, whether they are directly impacted by having the en-route event as part of their original route, or only indirectly by potential increased traffic due to rerouted vehicles. I have further examined the rerouted and non- rerouted vehicles separately.

As shown in the Table 5.6 and Table 5.7, there is a common advantage among FasRe, ShoRe and NRR which consists in the small portion of vehicles chosen to be rerouted in both grid and realistic scenario, which means that the three rerouting strategies would not affect the travel experience of the most drivers by repetitive rerouting requests. Although in grid map, they can all reduce the trip time considerably for rerouted vehicles, only NRR maintains this advantage in the city center of Cologne, while FasRe and ShoRe increase more trip times even for the rerouted vehicles. Therefore, in spite of the fact that NRR is designed for mitigating traffic congestion mainly from the global point of view, it still can provide attractive incentive for each individual driver to encourage them to accept rerouting instructions given by NRR.

If the driver does not accept the rerouting decision given by NRR, surprisingly, the results also indicate that in both maps, NRR is the only rerouting strategy that can reduce more trip time for more non-rerouted vehicles, in comparison to the number of non-rerouted vehicles which have their trip time increased. However, drivers are still being strongly encouraged to accept NRR's decision, because on average they would save up to at least 10 times more trip time than when not doing so.

Based on all the findings illustrated above, and one extra fact that even for non-rerouted vehicles the average wasted trip time is much less than the average saved trip time, the conclusion can be drawn that NRR is the only rerouting strategy that can not only bring significant benefit for rerouted vehicles, but also improve traffic which consists of non-rerouted vehicles and cause nearly no serious fairness issue.

Table 5.6: Impact of NRR, ShoRe, and FasRe on rerouted vehicles (Cologne Center / 8x7).

	NRR	**ShoRe**	**FasRe**
# vehicles having SAME trip time	0 / 1	0 / 0	0 / 0
# vehicles WASTED trip time	1 / 3	3 / 0	5 / 1
Average WASTED trip time (sec)	40.0 / 121.0	152.67 / 0	64.6 / 88.0
# vehicles SAVED trip time	136 / 124	3 / 117	3 / 118
Average SAVED trip time (sec)	558.69 / 854.35	81.33 / 896.91	47.33 / 886.94

Table 5.7: Impact of NRR, ShoRe, and FasRe on non-rerouted vehicles (Cologne Center / 8x7).

	NRR	ShoRe	FasRe
# vehicles having SAME trip time	865 / 926	854 / 855	910 / 862
# vehicles WASTED trip time	855 / 880	1284 / 1015	1205 / 997
Average WASTED trip time (sec)	4.25 / 24.05	39.45 / 22.15	13.67 / 23.33
# vehicles SAVED trip time	1218 / 1008	922 / 955	943 / 964
Average SAVED trip time (sec)	85.65 / 62.85	12.61 / 66.04	13.72 / 65.78

5.3.5 Impact of Varying Weight Allocation Strategies on NRR

In this subsection, I analyze the results of multiple NRR versions with varying weight allocations. I have compared 6 typical weight allocation strategies for NRR: one (NRR_ada) of them uses the adaptive process described with Equation 5.4; NRR_even is another strategy which evenly assigns weight values for all four factors of the cost function; the other four strategies assign full weight value for each of the four factors as shown in Table 5.8.

Table 5.8: Comparison of varying weight allocations strategies' impact on NRR (Cologne Center / 8x7).

	w	TTI	PTI	SI
NRR_ada	adaptive	1.42 / 2.79	2.85 / 5.12	0.81 / 0.63
NRR_even	$(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4})^T$	1.44 / 2.82	2.85 / 5.22	1.04 / 0.67
NRR_oc	$(1,0,0,0)^T$	1.43 / 2.82	2.85 / 5.25	0.90 / 0.67
NRR_tt	$(0,1,0,0)^T$	2.04 / 2.83	6.53 / 5.26	2.70 / 0.67
NRR_gd	$(0,0,1,0)^T$	1.67 / 2.79	3.89 / 5.15	2.06 / 0.67
NRR_gc	$(0,0,0,1)^T$	1.44 / 2.86	2.91 / 5.31	0.88 / 0.73

Table 5.8 validates that in both 8 × 7 and center of Cologne testbeds, NRR using adaptive weight allocation achieves the lowest congestion level (TTI) and

system instability (SI) while ensuring the highest travel time reliability (PTI). The NRR using evenly assigned weight values can also achieve good results in both testbeds, but it still performs a bit worse than NRR_ada. Moreover, as there is no justification for evenly assigned weights in the general case, it is risky (i.e. the performance could be much worse) when applied to other urban scenarios. Except for the strategy which assigns full weight to the road occupancy factor (NRR_oc), the other three weight allocation strategies do not show consistent performance in both testbeds.

5.4 Summary

In this chapter, the proposed Next Road Rerouting (NRR) based on the widely used adaptive traffic light control system and vehicle navigation system (VNS) is introduced. NRR diverts each vehicle affected by an en-route event to its optimal next road considering four factors measured in real time, namely the road occupancy, the travel time, the geographic distance to its intended destination and the geographic closeness to the closed road. The obtained evaluation results highlight that in comparison to the commonly used existing solutions, NRR can achieve a reduction of average trip time and an improvement of travel time reliability up to 38.02% and 65.42% respectively in a realistic map. Moreover, NRR can even improve the traffic conditions for more than half of non-rerouted vehicles. Besides, my evaluation results reveal also the devastating impact on traffic when overusing selfish rerouting (i.e., VNS) and highlight the benefit of the smart altruistic rerouting strategy (i.e., NRR).

Chapter 6
Next Road Rerouting with High Resolution Traffic Information

In this chapter, I firstly illustrate two problems incurred by using traffic information with low update frequency and limited coverage for rerouting vehicles. Then, I introduce the high resolution traffic information provider: VANETs and describe the motivation of applying it to improve the performance of NRR. Based on VANETs environment, an adaptive mechanism using k-means algorithm is proposed to select the most suitable group of iTLs to perform NRR more efficiently. Finally, the performance gains brought by the aforementioned improvements are shown by comparing NRR with existing induction loops.

6.1 Problems of Low Resolution Traffic Information

Compared to the existing solution using traditional shortest path finding algorithms in VNS, NRR reroutes vehicles efficiently by only calculating the best "next road" rather than the entire route. Moreover, all the required information to perform this rerouting is locally available, which saves huge potential cost for obtaining global information. However, the provider of information on road traffic conditions in NRR (i.e. induction loop) has limited update frequency (i.e. no less than 1 min) and coarse granularity (i.e. only arterial roads are covered), preventing it from supplying sufficient information to allow making better rerouting decision.

Besides, this problem can also be found in another popular existing method to collect traffic information, floating car data (FCD), which is usually applied in VNS with 2-5 mins update frequency and arterial roads coverage only.

For example, rerouting system in Figure 6.1 aims to balance traffic load locally, when vehicles are approaching the junction from road r_1 two cases can be distinguished as follows: in the first case, as shown in Figure 6.1 (a), the system will reroute the vehicles from r_1 to r_2, after a time duration t_1, as shown in Figure 6.1 (b), the system should reroute the vehicles from r_1 to r_3 but due to its slow update frequency ($T > t_1 + t_2$), the system's view stays at Figure 6.1 (a). Thus, the system keeps incorrectly rerouting the vehicles from r_1 to r_2. In the second case, as shown in Figure 6.1 (d), the system should reroute the vehicles from r_1 to r_3 because the latter has more capacity than r_2. However, due to its limited coverage for major roads, the system considers that r_2 has no traffic as in Figure 6.1 (e), and incorrectly reroute the vehicles into it. The results shown in both Figure 6.1 (c) and Figure 6.1 (f) reveal that the system has created another bottleneck without actually balancing the traffic.

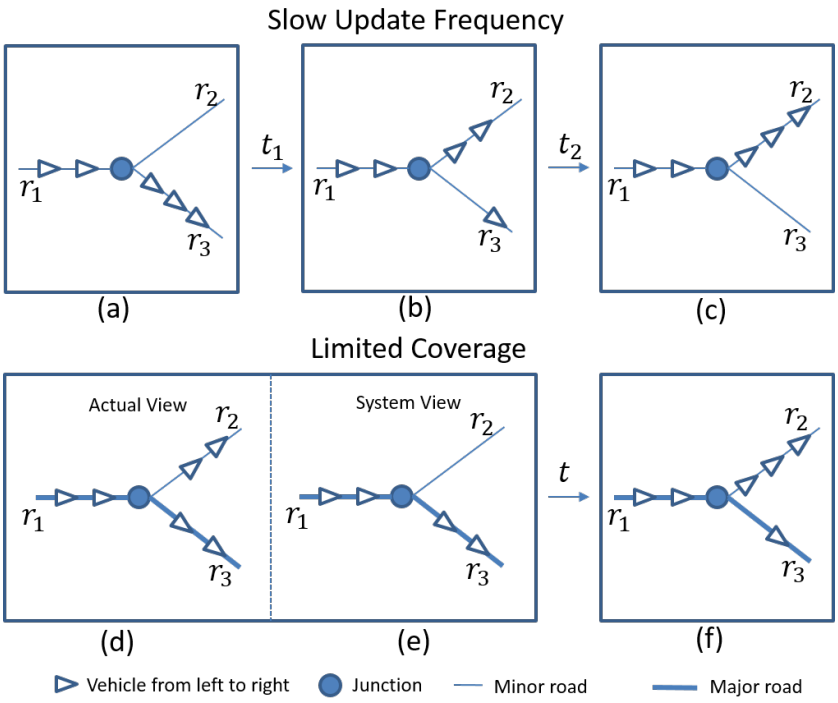


Figure 6.1: The impact of slow update frequency and limited traffic information coverage on the rerouting decision.

6.2 Motivation of VANETs for Vehicle Rerouting

One of the pre-requisite technologies to address the aforementioned problems is the emerging VANETs. VANETs can provide real-time traffic information with full coverage, high resolution and high update frequency to make timely adaptation possible in face of unexpected congestions. In a typical VANETs scenario, each vehicle broadcasts and receives "beacon" messages periodically to enable better awareness of the local traffic situation within its transmission range. In the two most widely recognized VANETs standards, this "beacon" message is called Cooperative Awareness Message (CAM) and is part of the "Facilities Layer" of ETSI ITS [92], or defined as "Basic Safety Message" in IEEE WAVE protocol stack [93]. A beacon message contains information about the speed, acceleration, position, heading, etc. of a certain vehicle. It is broadcast at least 10 times per second. VANETs can also cover all areas where roads have vehicles. Most importantly, VANETs technology fits perfectly into local urban scenarios, especially in NRR system where a whole city map is processed separately and simultaneously in different agents. This is because I surprisingly found that the length of up to about 90% of urban roads is within VANET's one-hop transmission area which is typically 300 m [94] (could be up to 1000 m [93]).

Note that the beacons suffer from high loss rate in practice, as shown in [103] where in some scenarios the effective transmission range can be shrank by up to 90%. NRR based on one-hop transmission can avoid this issue to some extent, compared to the extensive applications of multi-hop transmission in the literature to get the global traffic conditions. Moreover, some state-of-the-art technologies [104] can help to achieve more reliable transmission for one-hop used in NRR by implementing reliable beacons congestion control mechanisms such as the works done by [105]. The sampling rate can be accelerated to overcome the beacon loss as well. As can be found in Table 6.2 in the later evaluation this chapter, the best traffic condition can be achieved when sampling rate of traffic information is every 10s. While the normal frequency of beacons in VANETs is every 0.1s. This leaves huge potential for accelerating sampling rate.

Table 6.1: Selected map statistics.

	#roads<300m / #all roads	**Average road length (m)**	**Area (km^2)**	**World Congestion Ranking**
Beijing (Asia)	89.31%	138.04	1235.51	15th
Cape Town (Africa)	95.84%	94.51	646.03	55th
London (Europe)	98.27%	57.90	865.24	16th
Los Angeles (North America)	94.21%	115.08	864.68	10th
Rio de Janeiro (South America)	96.14%	96.80	572.37	3rd
Sydney (Australia)	97.43%	78.01	404.41	21st

This interesting conclusion is made from the statistics of various city maps from OpenStreetMap [88]. As shown in Table 6.1, I select a representative city from each continent (excluding Antarctica, because it has no big city with serious congestion problems) where citizens often experience heavy congestion in peak hours according to a well-recognized worldwide congestion report released by TomTom [91].

However, compared to driving safety and infotainment [96], the VANETs' research community has devoted little effort to improving traffic management. Most of related solutions in the literature are not practical enough as they usually require the exchange of vehicles' routing decisions, which violates the drivers' privacy. Moreover, the full route information can even be unknown for drivers traveling in new areas. Additionally, these solutions often need global traffic conditions information, relying on the error-prone coordination mechanism, based on ad hoc communication, among various moving vehicles. This mechanism also suffers from the non-line-of-sight problem [96] around intersections.

6.3 NRR with VANETs

6.3.1 Architecture

The deployment of NRR in VANETs maintains the major part of previous NRR's architecture (i.e. TOC connects multiple regional computers (RC) which are in charge of one or more NRR agents) with only one replacement of induction loops by VANETs. As shown in Figure 6.2, in the NRR shown in the previous chapters, the communication between the RSU and vehicles is in one-to-one manner only for rerouting confirmation. The same communication is now extended in a-NRR by adding one (RSU) to all (vehicles on one particular road) manner for traffic information collection. Another tip for on-site deployment is that the RSU should be placed in a higher location to avoid non-line-of-sight problem.

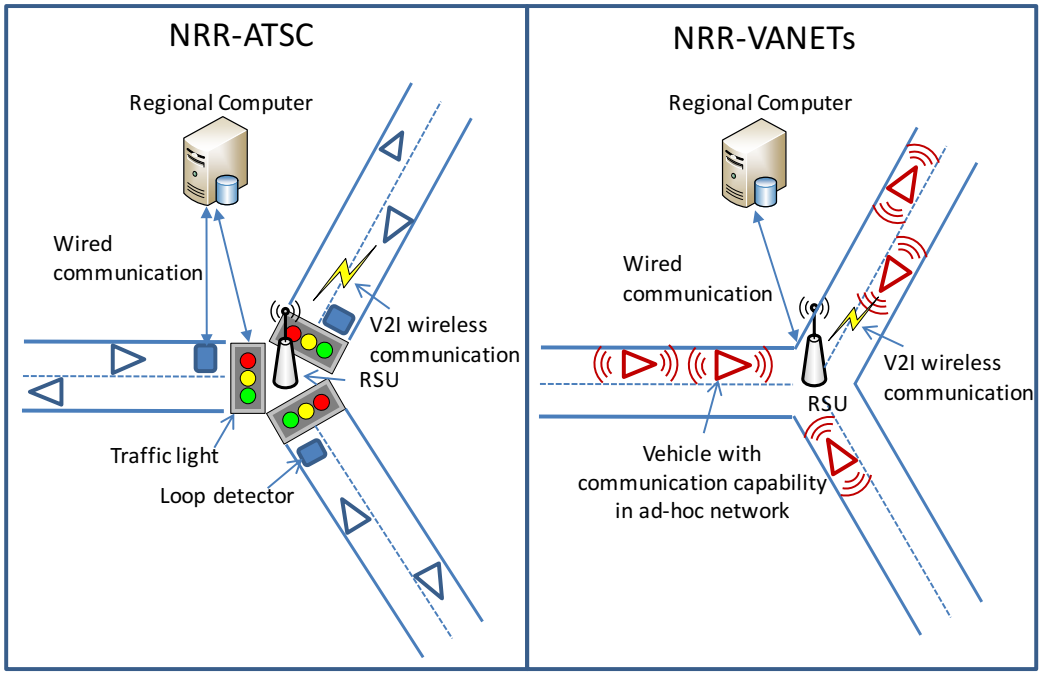


Figure 6.2: The comparison of architecture of NRR deployed in ATCS and VANETs.

6.3.2 Adaptive Selection for Operational Parameter

The NRR-enabled local areas (agents), which are the operational parameters of NRR, aim at achieving the best trade-off between system cost and traffic improvement. In my previous work [99], the evaluation results on grid maps reveal that with the increase of the number of NRR-enabled local areas, the improvement of traffic performance increases sharply up to the peak value when 5 agents are enabled, then slowly decreases. Therefore, the traffic improvements are not rigorously proportional to the system cost (i.e. amount of activated NRR agents).

However, enabling 5 local areas/agents cannot guarantee that the peak value will be reached under any set of traffic conditions and urban scenario. In order to find the most suitable agents to be enabled, traffic managers need to tune NRR with several trials manually according to various traffic and closed road locations. The extra cost raised from this process could potentially prevent NRR to be applied in future smart cites.

The version of NRR in the previous chapters chooses the number of activated agents based on the closed road location. It assumed that the closed road is the centre of en-route congestion distribution. It then chose the agent that contained the closed road first, if the achieved traffic improvements were not sufficient it enabled all its neighboring agents until reaching a satisfactory traffic improvement. However, its underlying assumption is not always correct and the number of agents will increase exponentially as the level value grows [99], making the selection more inaccurate. To solve these two problems, based on the philosophy of NRR, I believe that if a certain local area does not have a roughly balanced traffic load, then NRR needs to be executed. Thus, this selection process deals directly with the cause of congestion rather than the weak assumption of closed road.

Therefore, the key question is how NRR can determine whether a given local area has a balanced traffic load or not. To solve this problem, by making advantages of high-resolution traffic information provided by VANETs, k-means [95] algorithm is applied in NRR. First, for each local area with at least two outgoing

roads, the TOC of NRR calculates and updates the standard deviation STD of RO of all outgoing roads periodically, then k-means is applied to generate two (k=2) clusters in each time interval. One cluster has relatively smaller STD while the other has larger STD. The latter cluster of local areas needs to enable rerouting because larger STD means such agent does not have roughly balanced traffic load. Let us consider the example shown in Figure 6.3 where in 1800 secs simulation test, I close the central road of 8X7 grid map (in Figure 6.3.a) from 300th to 1500th sec. From in Figure 6.3.b it can be observed that all agents have similar small STD before the road closure. 600 secs after the occurrence of an event, it can be seen from in Figure 6.3.c that only few agents located in the vicinity of the closed road have much larger STD. It is, therefore, difficult to accurately determine a threshold separating the small number of agents with large STD from the majority of agents with small STD, but k-means algorithm, as a typical clustering algorithm, can easily achieve this with low computation cost because in this case k=2 and only one-dimension input data is used. As the STD of the most of agents still remains a small value when the event occurs, the initial centroids of k-means algorithm used in NRR is chosen as the maximum value, and median value of STD for all agents.

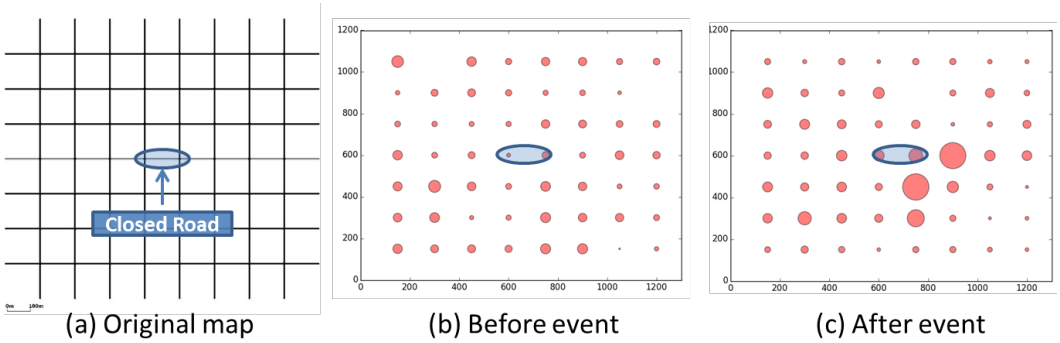


Figure 6.3: The geographical distribution of standard deviations (STD) of a set of RO values from all outgoing roads in each agent: before and after the occurrence of an event on the central road in 8X7 grid map scenario. The larger the circle is, the larger value of STD a certain agent has.

6.3.3 Evaluation Results and Analysis

To check if the adaptive mechanism for operational parameters selection can still provide competitive results, and the high resolution traffic information provided by VANETs can further reduce the congestion, I conducted a performance evaluation study by comparing NRR with both ATSC (N-A) and NRR with VANETs (N-V). In this study, I kept the same simulation settings as in the previous chapter for this evaluation, and only focused on the city center of Cologne scenario.

The first part of this evaluation consists in investigating the best update interval of traffic information for N-V. As previously introduced, each vehicle in VANETs environment periodically broadcasts its status every 0.1 second. However, if I choose this value as the update interval of traffic information, then the incurred overhead for the system to store and process the data would be excessively large. Moreover, this update interval is very sensitive to the traffic change, meaning that the rerouting decision will more likely lead to an increase in congestion, according to the instability of urban road traffic and Braess's Paradox. Therefore, in this evaluation I chose six update interval candidates: 1s, 10s, 30s, 60s, 120s, and 300s. The best one should present the lowest value in terms of both TTI and PTI.

From the results shown in Table 6.2, 10 seconds update interval for NRR with VANETs can provide the lowest value of TTI and PTI. These results are in line with my previous assumption (i.e. made in the second last paragraph in Chapter 1.1) stating that the ideal update frequency of traffic information for routing should be less than 30 seconds. N-V with 1 s has achieved a little increase in terms of TTI and PTI. This also confirms the assumption I made in the previous paragraph in this section and which states that unnecessarily short interval can even increase the congestion. Starting from 30 seconds, the effectiveness of congestion reduction decreases as the interval length grows.

The second part of this evaluation is to compare N-A and N-V. Here, I put N-A into a more practical setting. As all selected 8 agents (level 1) in the previous

evaluation for N-A are not controlled with traffic lights indeed, I enable all traffic light controlled agents in N-A to ensure the highest effectiveness of congestion reduction. Additionally, the update interval is set as 120 seconds, which is the shortest [50] that the common ATSC system can provide.

Table 6.2: Results of key congestion measurements for NRR with VANETs (N-V), and NRR with ATSC (N-A) under various traffic information update intervals.

	TTI	**PTI**
N-V (1s)	1.46	2.91
N-V (10s)	1.45	2.90
N-V (30s)	1.47	2.96
N-V (60s)	1.48	2.96
N-V (120s)	1.48	2.96
N-V (300s)	1.49	3.04
N-A (120s)	1.53	3.20

It is encouraging to see from Table 6.2 and Table 5.5 that even in a more practical setting, N-A can still outperform the best competing solution (FasRe) by 27.83% for TTI and 54.09% for PTI. More importantly, when comparing N-A and N-V under the same update interval 120 s, it can be concluded that 3.27% and 7.5% improvement in TTI and PTI can be achieved due the more accurate traffic information provided by VANETs. Finally, when N-A is compared with N-V under the best interval (i.e. 10 seconds), the reduction in TTI and PTI brought by N-V is 5.23% and 9.38%, respectively.

Additionally, N-V can outperform the best setting of practical N-A under all various update intervals. This observation ascertains that my proposed adaptive mechanism for operational parameters selection can replace the previous manual tuning process using the concept of “level” by providing competitive results without any human intervention.

Chapter 7
Conclusion and Future Work

Chapter 7 concludes this thesis by recalling the investigated research problem, summarizing the contributions made to state-of-the-art, and proposing three recommendations for future work.

7.1 Problem Overview

Urban traffic congestions have arisen since the beginning of urbanization around six decades ago. It incurs a huge amount of monetary loss as a result of excessive travel time and fuel consumptions in both developed and developing countries. Nowadays, thanks to the wide penetration of mobile devices and big data related technologies, commonly used intelligent transportation systems can reduce recurrent traffic congestion down to an acceptable or at least predictable level. However, to the best of my knowledge, there is still no solid improvement to the non-recurrent traffic congestion, due to its rigorous requirement for real-time decision making. This type of traffic congestion is often caused by en-route events, such as an unplanned parade, road works, sudden changes in weather conditions, car crashes and so on. This non-recurrent traffic congestion can propagate to wider areas in a very short time period. Consequently, the uncertainty of drivers' trips, which is supposed to go through or around the event-impacted area, will be significantly increased. Therefore, the objective of this thesis is to propose a vehicle rerouting system that can effectively and practically reduce traffic congestion due to unpredictable events in real-time.

7.2 Contributions to the State-of-the-Art

To address the unexpected urban traffic problem, this thesis proposes a highly practical and novel adaptive system called Next Road Rerouting (NRR). The three major contributions in this thesis are summarized as follows:

- **Improved multi-agent system architecture for vehicle rerouting in Adaptive Traffic Signal Control (ATSC) systems**. There are two common multi-agent system (MAS) designs in the state-of-the-art traffic management. For the first MAS design in ATSC, the agent is defined as regional computer and incoming lanes of each controlled intersection. The agent coordination mechanism is achieved by adapting the parameter "cycle" in traffic light scheduling. This parameter defines the time difference of the same traffic light timing plan for two consecutive junctions. Compared to this MAS architecture, my proposed architecture NRR extends the concept of agent by adding outgoing lanes. It also ensures coordination between neighbouring agents such that each of them gets accurate real-time traffic information about traffic conditions in its outgoing lanes. Moreover, vehicles running on the incoming lanes of each agent are required to send their destinations, if rerouting is needed, so that the geographical distance to each destination can be computed. Thus, MAS design in NRR makes use of all road information and improves the global trip delay rather than the local intersection delay ATSC focuses. Additionally, compared to the second common MAS design (i.e. often seen in vehicle routing system) which defines each moving vehicle as an agent, my NRR design defines an agent as a junction-centred region instead of a moving vehicle, hence greatly decreases the system complexity. In particular, the error-prone and impractical route choice exchange for agent coordination is also avoided.

- **NRR satisfies the rigorous real time requirement of non-recurrent congestion by initially computing only the next road direction**. In the literature, many vehicle rerouting systems use time consuming multi-hop approach to aggregate the real-time global traffic conditions. Moreover, the algorithms used in these systems for route assignment are all based on the typical shortest path finding algorithm, and need to run for many iterations to find the user equilibrium (UE) or system optimum (SO) solution. In the face of an en-route event, a meaningful rerouting consists in finding an alternative direction to bypass the incurred congestion, rather than seeking a full route a complete a trip. Thus, the usage of typical shortest path finding algorithms with the acquisation of global traffic info is unnecessary for this particular problem. In addition, the iterative algorithm does not fit the rigorous real time requirements of non-recurrent congestion. NRR uses local traffic info to compute the most promising next road only for a vehicle to avoid the congestion. Meanwhile, NRR reduces the potential negative impact on global traffic since the destination of each rerouting request is considered in minimizing the heuristic routing cost function.
- **Increased practicability in assumptions, design and validation methodology**. The state-of-the-art systems often assume the route choice for each vehicle is always available to share among each other. This assumption is critical for shifting UE (i.e. local optimum) to SO (i.e. global optimum) according to game theory. However, the route choice is in fact not precisely available, especially when drivers are moving in unfamiliar regions. Conversely, NRR only needs driver's destination which is already available for most vehicles already on the road. The design for calculating next road direction only fits driver's intuition and rigorous real-time requirement of actions in response to en-route events. Finally, instead of performing experiments on macroscopic (flow-level) small-scaled grid map, the methodology of performance evaluation used in NRR is done by using microscopic (vehicle-level), medium-scaled

quasi-realistic urban scenario. Therefore, the evaluation results of NRR are more convincing.

The effectiveness of NRR has been validated using the microscopic traffic simulator SUMO under both synthetic and quasi-realistic urban scenarios. The performance of NRR is evaluated and compared to two commonly used variations of VNS, ShoRe and FasRe, which are shortest travel distance rerouting and fastest travel speed rerouting respectively. Specifically, ShoRe and FasRe are using static A* algorithm one junction away from the closed road. In addition, two other approaches are used to set the evaluation benchmark. One is en-route event (ERE), which means "do nothing" when an event occurs. ERE shows how non-recurrent congestion can deteriorate the surrounding traffic conditions. The other is original (ORG), which is the best case of traffic, when no event happens in the observed area of simulation.

In comparison to the commonly used existing solutions (i.e. ShoRe and FasRe), NRR can achieve a reduction of average trip time and an improvement of travel time reliability up to 38.02% and 65.42% respectively in a quasi-realistic scenario of Cologne city center. With the help of high resolution traffic information by VANETs, and k-means based adaptive mechanism for operational parameter selection, the gain can further be improved by 5.23% and 9.38% for travel time and travel time reliability. Moreover, NRR can even improve the traffic conditions for more than half of non-rerouted vehicles, with the cost of 1.91% total travel length increase, and 4.46% vehicles rerouted only. Besides, my evaluation results reveal also the devastating impact on traffic when overusing selfish rerouting (i.e., VNS) and highlight the benefit of the smart altruistic rerouting strategy (i.e., NRR). Therefore, the conclusion can be draw that the thesis objective has been achieved. Specifically, as stated in the statement of research problem, the non-recurrent urban road congestion has been efficiently reduced under two practical constraints.

7.3 Recommendation for Future Work

To further improve the proposed NRR in reducing urban non-recurrent traffic congestion, three recommendations are suggested as follows:

- **Investigate the traffic impact of various road events according to realistic dataset.** This thesis shows a significant reduction of unexpected road congestion when applying NRR under one road closure in the centre of an urban road network. In practice, there are various types of en-route events that can lead to non-recurrent congestion. These events differ in the number of closed roads, the location of closed roads, the number of lanes closed or restricted on a certain road, the duration for road closure and so on. Thus, various events will have different impact on traffic conditions, and they call for different approaches to cope. For instance, when several consecutive minor roads are closed in a remote region like residential areas, this may cause heavy congestion in the vicinity but is unlikely to propagate to wider areas with major roads. Conversely, if even only one road is closed in the city centre during peak hour; it will increase the trip uncertainty for a great number of vehicles. When more realistic data become accessible in the future, the aforementioned research could be done to design a proactive vehicle rerouting solution able to act prior to congestion propagation.
- **Optimize the traffic signal decision by integrating route guidance with traffic light scheduling in the existing ATSC**. This thesis describes the idea of introducing vehicle rerouting functionality (NRR) based on the existing ATSC. NRR makes rerouting decision using the traffic information (i.e. degree of saturation) provided by induction loops in ATSC, and uses similar system architecture as ATSC. However, NRR does not merge the two types of decision (i.e. route guidance by NRR and traffic light scheduling by ATSC) into one single optimized decision to improve the traffic conditions. The recommended idea here is to extend the concept of "traffic signal" in traditional ATSC, making it not

confined to "traffic light signal" by introducing a "route guidance signal". It has been proven through decades of successful practice around the world that ATSC can reduce the delay when going through major intersections. However, the reduction of average trip time, which ATSC ignores, is more important to improve the performance of the whole road network. For instance, when en-route events occur, the alternative routes which NRR suggests will also need ATSC to assign higher priority traffic light plan to accommodate the sudden increased traffic.

- **Replace destination query process in NRR by the up-to-date prediction techniques.** In the typical rerouting process of NRR, as shown in Figure 4.2, the steps 3 and 4 combined represent the process for rerouting confirmation. The agent authorised by TOC broadcasts the rerouting alarm to all vehicles nearby, then if the vehicle needs to be rerouted, it sends the confirmation along with its intended destination back to the agent controlling it. Specifically, the reason why NRR needs to know the destination is to compute the value of the factor "geographic distance to destination", which is the key factor making the proposed routing cost function shifting its perspective from local to global. This is practical when compared to the state-of-the-art approaches, in which they assume the route choices are available for exchange among vehicles. However, it is still possible that some drivers will refuse to share their destinations for privacy reasons. Moreover, driver behaviour (e.g. manually choose destination location on the map shown in VNS and click the button to send) is involved in this rerouting confirmation process response. This increases the risk that the whole rerouting process might be stuck waiting for drivers' response, thus significantly increase the react time. By introducing destination prediction technology using large scale trajectory data [97][98], as the agent knows its own location and controls incoming and outgoing lanes, the improved process can be described as follows: the agent broadcasts the rerouting alarm message that contains the location of the closed road, as well as the suggested

next road directions for several most possible destination regions, without explicitly asking drivers' accurate destination location. Consequently, NRR becomes more practical and efficient by preventing the human involvement.

Bibliography

[1] 2014 Revision of World Urbanization Prospects. [Online]. Available: http://esa.un.org/unpd/wup/Publications/Files/WUP2014-Report.pdf

[2] D. Schrank, et al. *Urban Mobility Scorecard.* Technical Report August, Texas A&M Transportation Institute and INRIX, Inc, 2015.

[3] Beijing Traffic Management Bureau, Official Website, [Online]. Available: http://www.bjjtgl.gov.cn/jgj/ywsj/index.html

[4] Z. Yang, et al. "Review of Beijing's Comprehensive Motor Vehicle Emission Control Programs", *White Paper, International Council on Clean Transportation (ICCT)*, October 2015.

[5] E. W. Dijkstra, "A note on two problems in connexion with graphs," Numer. Math., vol. 1, no. 1, pp. 269–271, Dec. 1959.

[6] Dial, Robert B. "Algorithm 360: Shortest-path forest with topological ordering." Communications of the ACM issue.12, no.11 (1969):pp 632-633.

[7] Fredman, Michael L., and Robert Endre Tarjan. "Fibonacci heaps and their uses in improved network optimization algorithms." Journal of the ACM (JACM) issue. 34, no. 3 (1987): 596-615.

[8] Zhan, F. Benjamin, and Charles E. Noon. "Shortest path algorithms: an evaluation using real road networks." Transportation Science 32.1 (1998): 65-73.

[9] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern*., vol. 4, no. 2, pp. 100–107, Jul. 1968.

[10] S. Koenig and M. Likhachev, "Improved fast replanning for robot navigation in unknown terrain," *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, 2002, pp. 968-975 vol.1.

[11] Likhachev, Maxim, Geoffrey J. Gordon, and Sebastian Thrun. "ARA*: Anytime A* with provable bounds on sub-optimality." Advances in Neural Information Processing Systems. 2003..

[12] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, , and S. Thrun. "Anytime dynamic a*: An anytime, replanning algorithm". In Int. Conf. on Automated Planning and Scheduling, 2005.

[13] Kanoh, Hitoshi. "Dynamic route planning for car navigation systems using virus genetic algorithms." International Journal of Knowledge-based and Intelligent Engineering Systems 11.1 (2007): 65-78.

[14] C. L. P. Chen, J. Zhou and W. Zhao, "A Real-Time Vehicle Navigation Algorithm in Sensor Network Environments," in *IEEE Transactions on*

Intelligent Transportation Systems, vol. 13, no. 4, pp. 1657-1666, Dec. 2012.

[15] R. Doolan and G. M. Muntean, "VANET-Enabled Eco-Friendly Road Characteristics-Aware Routing for Vehicular Traffic," *Vehicular Technology Conference (VTC Spring), 2013 IEEE 77th*, Dresden, 2013, pp. 1-5.

[16] I. Chabini and S. Lan, "Adaptations of the A∗ algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 3, no. 1, pp. 60–74, Mar. 2002.

[17] S. Wang, S. Djahel, J. McManis, C. McKenna, and L. Murphy, "Comprehensive performance analysis and comparison of vehicles routing algorithms in smart cities," in *Proc. 5th Global Inf. Infrastruct. Netw. Symp.,* Trento, Italy, Oct. 2013, pp. 1–8.

[18] M. Zeeny, Multiple-Criteria Decision Making. New York: McGraw- Hill, 1982.

[19] J. Nzouonta, N. Rajgure, G. Wang, and C. Borcea, "VANET routing on city roads using real-time vehicular traffic information," IEEE Transactions on Vehicular Technology, vol. 58, no. 7, pp. 3609–3626, 2009.

[20] J. G. Wardrop, "Some theoretical aspects of road traffic research," *Proc. ICE—Eng. Div.*, vol. 1, no. 36, pp. 252–378, Jan. 1952.

[21] Nash, John F. "Equilibrium points in n-person games." Proc. Nat. Acad. Sci. USA 36.1 (1950): 48-49.

[22] Fukushima, Masao. "A modified Frank-Wolfe algorithm for solving the traffic assignment problem." Transportation Research Part B: Methodological 18.2 (1984): 169-177.

[23] Cherkassky, Boris V., Andrew V. Goldberg, and Tomasz Radzik. "Shortest paths algorithms: Theory and experimental evaluation." Mathematical programming 73.2 (1996): 129-174.

[24] Wagner, Dorothea, and Thomas Willhalm. "Speed-up techniques for shortest-path computations." Proceedings of the 24th annual conference on Theoretical aspects of computer science. Springer-Verlag, 2007.

[25] Fu, Liping, D. Sun, and Laurence R. Rilett. "Heuristic shortest path algorithms for transportation applications: state of the art", Computers & Operations Research 33.11 (2006): 3324-3343.

[26] Kaindl H, Kainz G. "Bidirectional heuristic search reconsidered", Journal of Artificial Intelligence Research 1997;7: 283–317.

[27] Dillenburg JF, Nelson PC. "Improving search efficiency using possible subgoals". Mathematical and Computer Modelling 1995;22(4–7):397–414.

[28] Jagadeesh GR, Srikanthan T, Quek KH. "Heuristic techniques for accelerating hierarchical routing on road networks". IEEE Transactions on Intelligent Transportation Systems 2002;3(4):301–9.

[29] J. de Dios Ortuzar and L. G. Willumsen, *Modelling Transport 4th Edition*. New York, NY, USA: Wiley, 2011.

[30] Patriksson, Michael. The traffic assignment problem: models and methods. Courier Dover Publications, 2015.

[31] BPR (1964) Traffic Assignment Manual: Bureau of Public Roads, U.S. Department of Commerce, Washington, D.C.

[32] HCM-Highway Capacity Manual (2010), Transportation Research Board, National Research Council, Washington, D.C., USA, 2010.

[33] T. Roughgarden and E. Tardos, "How bad is selfish routing?" *Journal of ACM*, vol. 49, no. 2, pp. 236–259, 2002.

[34] O. Jahn, R. H. Möhring, A. S. Schulz, and N. E. Stier-Moses, "System-optimal routing of traffic flows with user constraints in networks with congestion," *Oper. Res.*, vol. 53, no. 4, pp. 600–616, Jul./Aug. 2005.

[35] Steinberg, Richard, and Willard I. Zangwill. "The prevalence of Braess' paradox." Transportation Science 17.3 (1983): 301-318.

[36] Rapoport, Amnon, et al. "Choice of routes in congested traffic networks: Experimental tests of the Braess Paradox." Games and Economic Behavior 65.2 (2009): 538-571.

[37] Braess, Dietrich, Anna Nagurney, and Tina Wakolbinger. "On a paradox of traffic planning." Transportation science 39.4 (2005): 446-450.

[38] Duell, Melissa, et al. "Large-scale dynamic traffic assignment: practical lessons from an application in Sydney, Australia." Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on. IEEE, 2015.

[39] Chiu, Yi-Chang, et al. "Dynamic traffic assignment: A primer." Transportation Research E-Circular E-C153 (2011).

[40] C. Gawron, "Simulation-based traffic assignment: Computing user equi- libria in large street networks," Ph.D dissertation, Inst. Comput. Sci., Univ. Cologne, Köln, Germany, 1999.

[41] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO—Simulation of Urban MObility," *Int. J. Adv. Syst. Meas*., vol. 5, no. 3/4, pp. 128–138, Dec. 2012.

[42] Daganzo, Carlos F., and Yosef Sheffi. "On stochastic models of traffic assignment." Transportation science 11.3 (1977): 253-274.

[43] Russell, S. and Norvig, P. Artificial Intelligence: A Modern Approach (3rd edition), Prentice-Hall, 2010.

[44] Chen, Bo, and Harry H. Cheng. "A review of the applications of agent technology in traffic and transportation systems." Intelligent Transportation Systems, IEEE Transactions on 11.2 (2010): 485-497.

[45] A.G.Sims and K.W.Dobinson, "The Sydney coordinate adaptive traffic (SCAT) system philosophy and benefits," *IEEE Trans. Veh. Technol.,* vol. 29, no. 2, pp. 130–137, May 1980.

[46] P. B. Hunt, D. I. Robertson, R. D. Bretherton, and M. C. Royle, "The SCOOT on-line traffic signal optimization technique," *Traffic Eng. Control,* vol. 23, no. 4, pp. 190–192, Apr. 1982.

[47] [Online]. Available: http://www.scats.com.au/packages-hardware-controllers.html.

[48] [Online]. Available: http://www.scats.com.au/why-choose-scats.html.

[49] [Online]. Available: http://www.scoot-utc.com/WhereSCOOTUsed.php?menu=Overview.

[50] Selinger, Matt, and Luke Schmidt. "Adaptive Traffic Control Systems in the United States: Updated Summary and Comparison." (2010).

[51] El-Tantawy, Samah, Baher Abdulhai, and Hossam Abdelgawad. "Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC): methodology and large-scale application on downtown Toronto." *Intelligent Transportation Systems, IEEE Transactions on*, 14.3 (2013): 1140-1150.

[52] [Online]. Available: http://rhythmtraffic.com/insyncs-performance/.

[53] K. Pandit, D. Ghosal, H. M. Zhang, and C.-N. Chuah, "Adaptive traffic signal control with vehicular ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 62, no. 4, pp. 1459–1471, May 2013.

[54] B. Asadi and A. Vahidi, "Predictive cruise control: Utilizing upcoming traffic signal information for improving fuel economy and reducing trip time," *IEEE Trans. Control Syst. Technol.*, vol. 19, no. 3, pp. 707–714, May 2011.

[55] S. Lim and D. Rus, "Congestion-aware multi-agent path planning: Distributed algorithm and applications," *Comput. J.*, vol. 57, no. 6, pp. 825–839, Jul. 2013.

[56] Bar-Gera, Hillel. "Origin-based algorithm for the traffic assignment problem." Transportation Science 36.4 (2002): 398-417.

[57] Shin-ichi Inoue, Takuya Maruyama, "Computational Experience on Advanced Algorithms for User Equilibrium Traffic Assignment Problem and Its Convergence Error", *Procedia - Social and Behavioral Sciences,* Volume 43, 2012, Pages 445-456.

[58] Perederieieva O, Ehrgott M, Wang JY, Raith A. "A computational study of traffic assignment algorithms", in Australasian Transport Research Forum, Brisbane, Australia 2013 Oct, pp. 1-18.

[59] Nie, Y. "A Note on Bar-Gera's Algorithm for the Origin-Based Traffic Assignment Problem". Transportation Science, Vol. 46, pp. 27–38, 2012.

[60] Dial, R. B., "A Path-based User-equilibrium Traffic Assignment Algorithm That Obviates Path Storage and Enumeration", Transportation Research Part B, Vol. 40, Issue 10, pp.917–936, 2006.

[61] Meng, Q., Lee, D.-H., Cheu, R. L., Yang, H., "Logit-based stochastic user equilibrium problem for entry-exit toll schemes". *Journal of Transportation Engineering* 130 (6), 805–813, 2004.

[62] O. Landsiedel, Lemp, J. D., Kockelman, K. M., Damien, P., "The continuous cross-nested logit model: formulation and application for departure time choice". *Transportation Research Part B* 44 (5), 646–661, 2010.

[63] Meng, Q., Liu, Z., Wang, S., "Optimal distance tolls under congestion pricing and continuously distributed value of time". *Transportation Research Part E,* 48 (5), 937–957, 2012.

[64] R. Claes, T. Holvoet and D. Weyns, "A Decentralized Approach for Anticipatory Vehicle Routing Using Delegate Multiagent Systems", in *IEEE Transactions on Intelligent Transportation Systems,* vol. 12, no. 2, pp. 364-373, June 2011.

[65] H.F.Wedde and S.Senge, “BeeJamA: A distributed, self-adaptive vehicle routing guidance approach,” *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 4, pp. 1882–1895, Dec. 2013.

[66] P. Desai, S. W. Loke, A. Desai, and J. Singh, “CARAVAN: Congestion avoidance and route allocation using virtual agent negotiation,” *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1197–1207, Sep. 2013.

[67] D. J. Wilkie, et al. "Self-Aware Traffic Route Planning." *Twenty-Fifth AAAI Conference on Artificial Intelligence.* 2011.

[68] Wilkie, David, Cenk Baykal, and Ming C. Lin. "Participatory route planning." *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2014.

[69] Liu, Ruilin, Hongzhang Liu, Daehan Kwak, Yong Xiang, Cristian Borcea, Badri Nath, and Liviu Iftode. "Themis: A participatory navigation system for balanced traffic routing." *In Vehicular Networking Conference (VNC), IEEE*, pp. 159-166., 2014.

[70] Ali R. Güner, Alper Murat, Ratna Babu Chinnam, “Dynamic routing under recurrent and non-recurrent congestion using real-time ITS information”, *Computers & Operations Research,* Volume 39, Issue 2, February 2012, Pages 358-373.

[71] Kaparias, Ioannis, and Michael GH Bell. "A reliability-based dynamic re-routing algorithm for in-vehicle navigation." *Intelligent Transportation Systems (ITSC), 13th International IEEE Conference on*., 2010.

[72] Gao, S., Frejinger, E. and Ben-Akiva, M. E., “Adaptive Route Choices in Risky Traffic Networks: A Prospect Theory Approach”. *Transportation Research Part C,* 2010, 18(5):727-740.

[73] Soora Rasouli, Harry Timmermans, “Applications of theories and models of choice and decision-making under conditions of uncertainty in travel behavior research”, *Travel Behaviour and Society*, Vol. 1, Issue 3, Sept. 2014, pp. 79-90.

[74] Pramod Anantharam, Payam Barnaghi, Krishnaprasad Thirunarayan, and Amit Sheth. 2015. “Extracting City Traffic Events from Social Streams”. *ACM Trans. Intell. Syst. Technol.* 6, 4, Article 43 (July 2015), 27 pages.

[75] Linsey Xiaolin Pang, Sanjay Chawla, Wei Liu, Yu Zheng, “On detection of emerging anomalous traffic patterns using GPS data”, Data & Knowledge Engineering, Volume 87, September 2013, Pages 357-373.

[76] N. T. Osogami, H. Mizuta, and T. Ide, “Identifying the optimal road closure with simulation,” in *Proceedings of the 20th ITS World Congress Tokyo*, 2013.

[77] Skabardonis, Alexander, Pravin Varaiya, and Karl Petty. "Measuring recurrent and nonrecurrent traffic congestion." *Transportation Research Record: Journal of the Transportation Research Board* 1856 (2003): 118-124.

[78] S. Uppoor, O. Trullols-Cruces, M. Fiore, and J. M. Barcelo-Ordinas, “Generation and analysis of a large-scale urban vehicular mobility dataset,” *IEEE Trans. Mobile Comput*., vol. 13, no. 5, pp. 1061–1075, May 2014.

[79] A. Wegener et al., "TraCI: An interface for coupling road traffic and network simulators," in *Proc. 11th Commun. Netw. Simul. Symp.,* Ottawa, ON, Canada, Apr. 2008, pp. 155–163.

[80] A. Polus, "A study of travel time and reliability on arterial routes", *Transportation*, Vol. 8, No. 2, pp.141-151, 1979.

[81] A. Garcia, D. Reame, and R. L. Smith, "Fictitious play for finding system optimal routings in dynamic traffic network," *Transp. Res. B, Methodol.,* vol. 34, no. 2, pp. 147–156, Feb. 2000.

[82] K. Lyman and R. L. Bertini, "Using travel time reliability measures to improve regional transportation planning and operations," *Transp. Res. Rec., J. Transp. Res. Board,* no. 2046, pp. 1–10, Dec. 2008.

[83] J. Y. Wang, M. Ehrgott, and A. Chen, "A bi-objective user equilibrium model of travel time reliability in a road network," *Transp. Res. B, Methodol.,* vol. 66, pp. 4–15, Aug. 2014.

[84] J. H. Banks, *Introduction to Transportation Engineering*. New York, NY, USA: McGraw-Hill, 2002.

[85] J. Pan, I. S. Popa, K. Zeitouni, and C. Borcea, "Proactive vehicular traffic rerouting for lower travel time," *IEEE Trans. Veh. Technol.,* vol. 62, no. 8, pp. 3551–3568, Oct. 2013.

[86] Tan, Pang-Ning, Michael Steinbach, and Vipin Kumar. *Introduction to data mining.* Vol. 1. Boston: Pearson Addison Wesley, 2006.

[87] S. Wang, S. Djahel, and J. McManis, "An adaptive and VANETs-based next road re-routing system for unexpected urban traffic congestion avoidance," *in Proc. IEEE VNC,* Dec. 2015, pp. 196–203.

[88] M. Haklay and P. Weber, "OpenStreetMap: User-generated street maps," *IEEE Pervasive Comput.,* vol. 7, no. 4, pp. 12–18, Oct.–Dec. 2008.

[89] C. Varschen and P. Wagner, "Mikroskopische Modellierung der Personenverkehrsnachfrage auf Basis von Zeitverwendungstagebüchern," *Stadt Region Land,* vol. 81, pp. 63–69, Dec. 2006.

[90] [Online]. Available: http://ops.fhwa.dot.gov/publications/tt_reliability/brochure/.

[91] [Online]. Available: http://www.tomtom.com/en ie/trafficindex/#/list.

[92] ETSI TS 102 637-2, Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service. ETSI, Sophia Antipolis Cedex, France, 2010.

[93] IEEE Vehicular Technology Society, "IEEE standard for wireless access in vehicular environments (WAVE). Multi-channel operation," IEEE Std 1609.4-2010, 2010.

[94] F. Bai, D. D. Stancil, and H. Krishnan, "Toward understanding characteristics of dedicated short range communications (DSRC) from a perspective of vehicular network engineers," *in Proc. 16th Annu. Int. Conf. MobiCom*, 2010, pp. 329-340

[95] J. A. Hartigan and M. A. Wong, "Algorithm AS136: A k-means Clustering Algorithm," *Applied Statistics*, vol. 28, pp. 100-108, 1979.

[96] M. Conti, S. Giordano, “Mobile ad hoc networking: milestones, challenges, and new research directions,” *Communications Magazine, IEEE*, vol.52, no.1, pp.85-96, January 2014.

[97] Krumm, John, and Eric Horvitz. "Predestination: Inferring destinations from partial trajectories." UbiComp 2006: Ubiquitous Computing. Springer Berlin Heidelberg, 2006. 243-260.

[98] Trinh Minh Tri Do, Daniel Gatica-Perez, “Where and what: Using smartphones to predict next locations and applications in daily life”, Pervasive and Mobile Computing, Volume 12, June 2014, Pages 79-91.

[99] S. Wang, S. Djahel, and J. McManis, “A multi-agent based vehicle re-routing system for unexpected traffic congestion avoidance,” in *Proc. 17th IEEE ITSC*, Qingdao, China, Oct. 2014, pp. 2541–2548.

[100] [Online]. Available: https://www.tomtom.com/

[101] http://www.ibm.com/support/knowledgecenter/#!/SSTMV4/welcome

[102] T. Bouali, M. El Hami, A. Ben Sassi, SM. Senouci, T. Sophy, A. Kribeche, “A Vehicular Network Architecture for Data Collection: Application to an Itinerary Planning Service in Smart Cities”, *IEEE GIIS 2015*, Guadalajara, Mexico, 28-30 October 2015.

[103] R. K. Schmidt, B. Kloiber, F. Schuttler, and T. Strang, “Degradation of communication range in VANETs caused by interference 2.0 - real-world experiment,” in *Lecture Notes in Computer Science. Springer Science Business Media,* 2011, pp. 176–188.

[104] N. Haouari, S. Moussaoui, M. Guerroumi, SM. Senouci, “Local Density Estimation for Vehicular Ad-hoc Networks”, *IEEE GIIS2016* – under submission, Porto, Portugal, 19-21 October 2016.

[105] S. Zemouri, S. Djahel and J. Murphy, "A Short-Term Vehicular Density Prediction Scheme for Enhanced Beaconing Control," *2015 IEEE Global Communications Conference (GLOBECOM)*, San Diego, CA, 2015.

Appendix A – Key Code Snippets for Simulation

A.1 Crop the Large-Scale Simulation Scenario

In this thesis, several subsets of a large-scale simulation scenario, TAPASCologne 0.24.0, are used for multiple simulation experiments. For instance, in Chapter 3, the three subsets: city centre, suburban, and remote area scenario are cut from the scenario of the greater Cologne area and used for evaluating the performance of vehicle routing algorithms. Additionally, in Chapter 5 and Chapter 6, the city centre of the same large-scale simulation scenario is used for evaluating the performance of proposed Next Road Rerouting system.

To get the subset of the original large scale simulation scenario is a non-trivial technical problem. This process consists in two steps: crop the map given the limited spatial boundary, and crop the traffic demand given the same limited spatial boundary and limited temporal range.

The Python implementation of my way to crop the large-scale simulation scenario in SUMO is given as follows:

```
1.  # This script crops the specific map and its corresponding demand data given a map ra
    nge.
2.  # Please refer to http://sumo.dlr.de/wiki/NETCONVERT for the parameter setting.
3.  import argparse
4.  import subprocess
5.  import os
6.  import sys
7.  import xml.etree.cElementTree as ET
8.
9.
10. def get_options():
11.     arg_parser = argparse.ArgumentParser()
```

```
12.     arg_parser.add_argument('--in_dir', default=r"C:\SUMO\Scenarios\TAPASCologne-
    0.24.0",
13.                             help='The root directory of input')
14.     arg_parser.add_argument('--geo_bound', default='12200,13000,14000,14200',
15.                             help='The geographic boundary for cropping the map')
16.     arg_parser.add_argument('--time_bound', default='0,3600',
17.                             help='The time boundary coordinates for cropping the traf
    fic demand')
18.     arg_parser.add_argument('--out_dir', default=r"C:\SUMO\Scenarios\adanrr",
19.                             help='The root directory of output')
20.     arg_parser.add_argument('--out_fn', default=r"cologne_mini",
21.                             help='The file name of three output scenario files')
22.     return arg_parser.parse_args()
23.
24.
25. def path_conv(in_dir, out_dir, out_fn):
26.
27.     in_map = os.path.join(in_dir, 'cologne2.net.xml')
28.     in_demand = os.path.join(in_dir, 'cologne6to8.rou.xml')
29.     in_vtype = os.path.join(in_dir, 'vtypes.add.xml')
30.
31.     mid_demand = os.path.join(out_dir, out_fn+'_mid.rou.xml')
32.
33.     out_map = os.path.join(out_dir, out_fn+'.net.xml')
34.     out_demand = os.path.join(out_dir, out_fn+'.rou.xml')
35.     out_cfg = os.path.join(out_dir, out_fn+'.sumocfg')
36.
37.     return in_map, in_demand, in_vtype, mid_demand, out_map, out_demand, out_cfg
38.
39.
40. def crop_demand_by_time(in_file, time_bound, out_file):
41.     tree = ET.parse(in_file)
42.     root = tree.getroot()
43.     new_root = ET.Element(root.tag, root.attrib)
44.     min_bound, max_bound = time_bound.split(',')
45.     item_count = 0
46.     for child in root:
47.         item_count += 1
48.         cur_dpt_time = float(child.attrib['depart'])
49.         if item_count == 1:
50.             start_time = cur_dpt_time
51.             min_bound = start_time + float(min_bound)
52.             max_bound = start_time + float(max_bound)
53.         elif min_bound <= cur_dpt_time < max_bound:
54.             new_child = ET.SubElement(new_root, child.tag, child.attrib)
55.             for child2 in child:
56.                 ET.SubElement(new_child, child2.tag, child2.attrib)
57.         elif cur_dpt_time >= max_bound:
58.             ET.ElementTree(new_root).write(out_file)
59.             return
60.
61.
62. def create_sumocfg(net_file, demand_file, cfg_file, vtype_file):
63.
64.     tree = ET.parse(r"C:\SUMO\Scenarios\TAPASCologne-0.24.0\cologne.sumocfg")
65.     root = tree.getroot()
66.     new_root = ET.Element(root.tag, root.attrib)
67.
68.     for child in root:
69.         new_child = ET.SubElement(new_root, child.tag, child.attrib)
```

```
        for child2 in child:
            if child2.tag == "net-file":
                child2.attrib['value'] = net_file
            elif child2.tag == "route-files":
                child2.attrib['value'] = demand_file
            elif child2.tag == "additional-files":
                child2.attrib['value'] = vtype_file
            ET.SubElement(new_child, child2.tag, child2.attrib)
    ET.ElementTree(new_root).write(cfg_file)


def main():
    print "Step1 - Reading arguments..."
    args = get_options()
    in_map, in_demand, in_vtype, mid_demand, out_map, out_demand, out_cfg = \
        path_conv(args.in_dir, args.out_dir, args.out_fn)

    print "Step2 - Cropping original map..."
    sumo_netcon = os.path.join(os.environ.get('SUMO_HOME'), 'bin', 'netconvert.exe')

    netcon_proc = subprocess.Popen([sumo_netcon, '-s', in_map, '--keep-edges.in-boundary', args.geo_bound,
                        '--remove-edges.isolated', '-o', out_map], stdout=sys.stdout)
    netcon_proc.wait()

    print "Step3 - Cropping original demand by time boundary..."
    crop_demand_by_time(in_demand, args.time_bound, mid_demand)

    print "Step4 - Cropping original traffic demand by cropped map boundary..."
    sumo_cutroute = os.path.join(os.environ.get('SUMO_HOME'), 'tools', 'route', 'cutRoutes.py')
    cutrou_proc = subprocess.Popen(['python', sumo_cutroute, out_map, mid_demand, '--routes-output',
                                    out_demand, '--orig-net', in_map], stdout=sys.stdout)
    cutrou_proc.wait()

    print "Step5 - Generating SUMO configuration file for cropped scenario..."
    create_sumocfg(out_map, out_demand, out_cfg, in_vtype)
    print "Cropping succeed!"

if __name__ == '__main__':
    main()
```

The above script is written using Python 2.7, and tested on Windows 8 64-bit Professional and SUMO 0.25.0. I appreciate the suggestions from two senior developers in SUMO community: Jakob Erdmann and Michael Behrisch.

A.2 Two-Step Rerouting in NRR

In this thesis, the two-step rerouting process in the proposed NRR has been previously described using a sequence diagram Figure 4.2. To show more details, the implementation of this two-step rerouting process using Python 2.7 is shown as follows:

First step rerouting, choose a suitable next road direction to follow:

```
def next_road_rerouting(sumo_map, closed_roads, act_tls):
    '''''
    The first step of next road rerouting.

    Args:
        sumo_map: The objects of test map in SUMO format.
        closed_roads: The ids of the closed road segments.
        act_tls: The objects of the junctions where NRR enables.
    '''
    # key: vehicle id;
    # value: [suggested next road id, destination road id]
    global NEXT_ROAD_DICT
    # key: "origin road id,destination road id";
    # value: the length in meters of the shortest path; 99999.99 if not connected
    global REACHABILITY_DICT

    for k in act_tls:
        for j in k.getIncoming():
            j_id = j.getID()
            # get vehicles' ids on the road j_id
            road_vehs = traci.edge.getLastStepVehicleIDs(j_id)
            if len(road_vehs) != 0:
                # get id of the first vehicle on the road j_id
                first_veh_id = road_vehs[-1]
                first_veh_route = traci.vehicle.getRoute(first_veh_id)
                # check if the next road direction already suggested for this vehicle

                if not NEXT_ROAD_DICT.has_key(first_veh_id):
                    # check if the first vehicle needs to be rerouted
                    if is_affected(closed_roads, first_veh_route):
                        # get the destination road id of the first vehicle
                        first_veh_destrd_id = first_veh_route[-1]
                        nr_id = next_road_id(sumo_map, closed_roads, sumo_map.getEdge
(first_veh_destrd_id), k)
                        # ignore when the suggested road is not available to drive
                        if nr_id == -1 or \
                                        REACHABILITY_DICT[j_id+','+nr_id] == '99999.9
9' or \
                                        REACHABILITY_DICT[nr_id+','+first_veh_destrd_
id] == '99999.99':
                            continue
                        NEXT_ROAD_DICT[first_veh_id] = [nr_id, first_veh_destrd_id]
                        traci.vehicle.setRoute(first_veh_id, [j_id, nr_id])
```

Second step rerouting, when drives into the suggested next road, get the full route for the rest journey using VNS:

```
1.  def update_route():
2.      '''''
3.      The second step of next road rerouting.
4.
5.      Check whether the vehicle already entered its suggested next road.
6.      If it is the case, retrieve its intended destination
7.      and use VNS to get full route for its rest journey.
8.
9.      '''
10.     global NEXT_ROAD_DICT
11.     if NEXT_ROAD_DICT != {}:
12.         for k in NEXT_ROAD_DICT.keys():
13.             if k in traci.vehicle.getIDList():
14.                 if traci.vehicle.getRoadID(k) == NEXT_ROAD_DICT[k][0]:
15.                     traci.vehicle.changeTarget(k, NEXT_ROAD_DICT[k][1])
16.                     traci.vehicle.rerouteTraveltime(k)
17.                     del(NEXT_ROAD_DICT[k])
```

A.3 Adaptive Selection for NRR Parameters

In the framework of the two-step rerouting process, there are two adaptive mechanisms for selecting algorithmic and operational parameters.

The implementation of the function `next_road_id` (i.e. invoked in line 32 of the first snippet in Appendix A.2) is shown as below, in which the adaptive weight allocation (i.e. algorithmic parameters) mechanism can be found between line 58 and line 82. The code between line 84 and line 109 is used for evaluating NRR with different weights allocations in Chapter 5.3.5.

```
1.  def next_road_id(sumo_net, closed_roads, dest_road_obj, tl_obj, scenario_name):
2.      '''''
3.      The key process first step of next road rerouting.
4.
5.      Get the id of the NRR suggested next road using routing cost function.
6.
7.      Args:
8.          sumo_net: The map data in sumo format.
9.          closed_roads: The objects of closed roads
10.         dest_road_obj: The sumo object of the destination road.
11.         tl_obj: The object of the junction where NRR agents enabled.
```

```
        scenario_name: The name of NRR version represents different weight allocations.

    Returns:
        The id of the NRR suggested next road in string type.
    '''

    # Only for outgoing roads
    cost_dict = {}
    travel_time_dict = {}
    road_occupancy_dict = {}
    geo_dist_dict = {}
    geo_close_dict = {}

    # If the CURRENT road is destination, return minus one
    # Means don't need to do next road rerouting
    if tl_obj is dest_road_obj.getToNode() or len(tl_obj.getOutgoing()) <= 1:
        return -1

    for i in tl_obj.getOutgoing():
        i_id = i.getID()
        # If one of its NEXT roads is destination, return it. NRR finished
        if i_id == dest_road_obj.getID():
            return i_id

        travel_time_dict[i_id] = travel_time(i)
        road_occupancy_dict[i_id] = road_occupancy(i)
        geo_close_dict[i_id] = geoclose_cng(sumo_net.getEdge(closed_roads[0]), sumo_net.getEdge(closed_roads[-1]), i)
        geo_dist_dict[i_id] = geodist_dest(i_id, dest_road_obj.getID())

    max_dict = {}
    max_dict["tt"] = max(travel_time_dict.values())
    max_dict["oc"] = max(road_occupancy_dict.values())
    max_dict["gd"] = max(geo_dist_dict.values())
    max_dict["gc"] = max(geo_close_dict.values())

    min_dict = {}
    min_dict["tt"] = min(travel_time_dict.values())
    min_dict["oc"] = min(road_occupancy_dict.values())
    min_dict["gd"] = min(geo_dist_dict.values())
    min_dict["gc"] = min(geo_close_dict.values())

    deno_tt = max_dict["tt"]-min_dict["tt"]
    deno_oc = max_dict["oc"]-min_dict["oc"]
    deno_gd = max_dict["gd"]-min_dict["gd"]
    deno_gc = max_dict["gc"]-min_dict["gc"]

    # adaptive weights allocation
    if scenario_name == 'nrr_ada':
        if np.mean(travel_time_dict.values()) == 0:
            cv_tt = 0.0
        else:
            cv_tt = np.std(travel_time_dict.values())/np.mean(travel_time_dict.values())
        if np.mean(road_occupancy_dict.values()) == 0:
            cv_oc = 0.0
        else:
            cv_oc = np.std(road_occupancy_dict.values())/np.mean(road_occupancy_dict.values())
```

```
        if np.mean(geo_dist_dict.values()) == 0:
            cv_gd = 0.0
        else:
            cv_gd = np.std(geo_dist_dict.values())/np.mean(geo_dist_dict.values())
        if np.mean(geo_close_dict.values()) == 0:
            cv_gc = 0.0
        else:
            cv_gc = np.std(geo_close_dict.values())/np.mean(geo_close_dict.values())

        cv_sum = cv_tt + cv_oc + cv_gd + cv_gc

        w_tt = cv_tt/cv_sum
        w_oc = cv_oc/cv_sum
        w_gd = cv_gd/cv_sum
        w_gc = cv_gc/cv_sum

    # five other compared weights allocations
    if scenario_name == 'nrr_even':
        w_tt = 0.25
        w_oc = 0.25
        w_gd = 0.25
        w_gc = 0.25
    if scenario_name == 'nrr_tt':
        w_tt = 1.00
        w_oc = 0.00
        w_gd = 0.00
        w_gc = 0.00
    if scenario_name == 'nrr_oc':
        w_tt = 0.00
        w_oc = 1.00
        w_gd = 0.00
        w_gc = 0.00
    if scenario_name == 'nrr_gd':
        w_tt = 0.00
        w_oc = 0.00
        w_gd = 1.00
        w_gc = 0.00
    if scenario_name == 'nrr_gc':
        w_tt = 0.00
        w_oc = 0.00
        w_gd = 0.00
        w_gc = 1.00

    for i in tl_obj.getOutgoing():
        i_id = i.getID()

        if deno_tt == 0.0:
            term1 = 0.0
        else:
            term1 = w_tt * ((travel_time_dict[i_id]-min_dict["tt"])/deno_tt)
        if deno_oc == 0.0:
            term2 = 0.0
        else:
            term2 = w_oc * ((road_occupancy_dict[i_id]-min_dict["oc"])/deno_oc)
        if deno_gd == 0.0:
            term3 = 0.0
        else:
            term3 = w_gd * ((geo_dist_dict[i_id]-min_dict["gd"])/deno_gd)
        if deno_gc == 0.0:
```

```
            term4 = 0.0
        else:
            term4 = w_gc * ((geo_close_dict[i_id]-min_dict["gc"])/deno_gc)
        cost_dict[i_id] = term1 + term2 + term3 + term4
    return min(cost_dict.iterkeys(), key=lambda k: cost_dict[k])
```

In addition to the code implementation shown above, the rigorous description of next road choice has already been presented using equations from Equation 5.1 to Equation 5.4.

The implementation of adaptive selection for NRR agents (i.e. operational parameter) is shown as follows:

```
from pylab import *
from scipy.cluster.vq import *

def nrr_juncs_kmeans():

    global STD  # {junction_obj: the standard deviation of occupancy of outgoing lane
s}

    input_data = vstack((array(STD.values())))
    initial_centroids = array([[max(STD.values())], [median(STD.values())]])

    centroids, _ = kmeans(input_data, initial_centroids)
    idx, _ = vq(input_data, centroids)

    return array(STD.keys())[idx == 0]
```