Safe and Efficient Intelligent Intersection Control of Autonomous Vehicles

Qiang Lu
University of Denver

Safe and Efficient Intelligent Intersection Control of Autonomous Vehicles

---

A Dissertation

Presented to

the Faculty of the Daniel Felix Ritchie School of Engineering and Computer Science

University of Denver

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

---

by

Qiang Lu

March 2019

Advisor: Mohammad H. Mahoor

Author: Qiang Lu
Title: Safe and Efficient Intelligent Intersection Control of Autonomous Vehicles
Advisor: Mohammad H. Mahoor
Degree Date: March 2019

ABSTRACT

In this dissertation, we address a problem of safe and efficient intersection crossing traffic management of autonomous and connected ground traffic. Toward this objective, we propose several algorithms to handle different traffic environments. First, an algorithm that is called the Discrete-time occupancies trajectory (DTOT) based Intersection traffic Coordination Algorithm (DICA) is proposed. All vehicles in the system are Connected and Autonomous Vehicles (CAVs) and capable of wireless Vehicle-to-Intersection communication. The main advantage of DICA is that it enables us to utilize the intersection space more efficiently resulting in less delay for vehicles to cross the intersection. In the proposed framework, an intersection coordinates the motions of CAVs based on their proposed DTOTs to let them cross the intersection efficiently while avoiding collisions. In case when there is a potential collision between vehicles' DTOTs, the intersection modifies conflicting DTOTs to avoid the collision and requests CAVs to approach and cross the intersection according to the modified DTOTs. We also prove that the basic DICA is deadlock free and starvation free. We show that the basic DICA has a computational complexity of $\mathcal{O}(n^2L_m^3)$ where n is the number of vehicles granted to cross an intersection and L_m is the maximum length of intersection crossing routes. To improve the overall computational efficiency of the algorithm, the basic DICA is enhanced by several computational techniques. The enhanced algorithm has a reduced computational complexity of $\mathcal{O}(n^2L_m\log_2 L_m)$.

The problem of evacuating emergency vehicles as quickly as possible through autonomous and connected intersection traffic is also addressed in this dissertation. The proposed Reactive DICA aims to determine an efficient vehicle-passing sequence which allows the emergency vehicle to cross an intersection as soon as possible while the travel times of other normal vehicles are minimally affected. When there are no emergency vehicles within the intersection area, the vehicles are controlled by DICA. When there are emergency vehicles entering communication range, we prioritize emergency vehicles through the optimal ordering of vehicles. Since the number of possible vehicle-passing sequences increases rapidly with the number of vehicles, finding an efficient sequence of vehicles in a short time is the main challenge of the study. A genetic algorithm is proposed to solve the optimization problem which finds the optimal vehicle sequence in real time that gives the emergency vehicles the highest priority.

We then address an optimization problem of autonomous intersection control which provides the optimal trajectory for every entering vehicle. Based on the algorithm DICA, we improve the conservative way of trajectory generation which is the key part of DICA to be an optimization approach using mixed integer programming. The new algorithm is named Mixed integer programming based Intersection Coordination Algorithm (MICA) with the objective of maximizing the final position of a new head vehicle over a fixed time interval. Constraints from space conflicting vehicles are modeled using binary variables to represent the vehicle's future crossing behavior. The influence of immediate front vehicles of the vehicle of interest is also modeled as constraints in the problem formulation to obtain a feasible optimal trajectory while potential collisions are safely avoided. Finally, based on MICA, we propose a novel vehicle-intersection interaction mechanism MICACO which is designed to handle imperfect communication, i.e., message delay and loss. To ensure the successful delivery of messages, we add two more message types and corresponding simple rules. State machines of intersection and vehicles are designed properly to ensure the safety of every vehicle.

We verify the efficiency of the proposed algorithms through simulations using SUMO. The simulation results show that DICA performs better than another existing intersection management scheme: Concurrent Algorithm in [1]. The overall throughput, as well as the computational efficiency of the computationally enhanced DICA, are also compared with those of an optimized traffic light control. The efficiency of the proposed Reactive DICA is validated through comparisons with DICA and a reactive traffic light algorithm. The results show that Reactive DICA is able to decrease the travel times of emergency vehicles significantly in light and medium traffic volumes without causing any noticeable performance degradation of normal vehicles. The simulation results show that MICA is able to reduce congestions of an intersection significantly compared with DICA. We also show MICACO's performance through comparisons with MICA and an optimized traffic light.

ACKNOWLEDGEMENTS

First of all, I would like to extend my sincere gratitude to my advisor Dr. Mohammad H. Mahoor, and co-advisor Dr. Kyoung-Dae Kim for their instructive advice and useful suggestions on my dissertation and during my whole Ph.D. I am deeply grateful for their help in the completion of this dissertation. I learned a lot of knowledge from their excellent performance in teaching and research capabilities as well as high motivation and responsibility to students.

I am also deeply indebted to all the other professors and staff like Dr. Jun Jason Zhang, Dr. Kimon P. Valavanis, Dr. Amin Khodaei, Dr. David Gao etc. for their direct and indirect help to me.

Special thanks should go to my friends who have put considerable time and effort into their comments on the draft.

Finally, I am indebted to my parents, my sister for their continuous support and encouragement. I also want to thank my fiancée for all her love, support, and understanding.

TABLE OF CONTENTS

Acknowledgements v
List of tables ix
List of figures x

1 Introduction 1
1.1 Motivation and Background 1
1.2 Contribution 7
1.3 Organization of the Dissertation 9

2 Literature Review 11
2.1 Autonomous Control of Highway Systems 11
2.1.1 Vehicle Platooning 11
2.1.2 Algorithms and Protocols for Individual Vehicles 12
2.2 Autonomous Intersection Control 14
2.2.1 Centralized Control 15
2.2.2 Decentralized Control 18
2.2.3 Emergency Vehicle Handling 20
2.2.4 Communication Uncertainties 21
2.2.5 Mixed Traffic 23
2.2.6 Machine Learning 23

3 DTOT-based Intersection Traffic Management 25
3.1 Assumptions 25
3.2 Interaction between an ICA and a CAV 27
3.3 DTOT-based Intersection Traffic Coordination 30
3.3.1 Collision Avoidance with Front Vehicles 31
3.3.2 Vehicles for Collision Avoidance 33
3.3.3 DTOT Update 35
3.4 Liveness Analysis 36
3.5 Simulation 39
3.5.1 Simulation Setup 39
3.5.2 Simulation Results 41
3.6 Summary 44

4 Computational Complexity Improvements of DICA 45
4.1 Computational Complexity Analysis 45
4.2 Algorithm Improvements 49
4.2.1 Reduced Number of Vehicles for the Space-Time Conflict Check 50
4.2.2 Efficient Space Conflict Check 51
4.2.3 Approximate Occupancy Time Interval Calculation 52
4.2.4 Efficient Occupancies Comparison 53
4.3 Simulation 56
4.3.1 Simulation Setup 56

4.3.2 Simulation Results 58
4.4 Conclusion 66

5 Reactive DICA: an Approach for Expedited Crossing of Emergency Vehicles 67
5.1 Reactive DICA 67
5.2 Genetic Algorithm for Vehicle Ordering 73
5.2.1 Chromosome Encoding and Feasibility Check 75
5.2.2 Crossover and Mutation 76
5.2.3 Fitness and Generating New Generation 78
5.2.4 Stopping Criterion 79
5.3 Simulation 79
5.3.1 Simulation Setup 79
5.3.2 Reactive Traffic Light 81
5.3.3 Simulation Results 82
5.4 Summary 87

6 MICA: Optimal Control of Autonomous and Connected Intersection Traffic 88
6.1 Vehicle-Intersection Interaction 88
6.2 Problem Formulation of MICA 90
6.2.1 Single Vehicle Trajectory Generation 90
6.2.2 Mixed Integer Programming for Collision Avoidance 92
6.2.3 Optimization Problem Formulation 98
6.2.4 MICA 99
6.3 Simulation 99
6.3.1 Simulation Setup 100
6.3.2 Simulation Results 101
6.4 Summary 104

7 MICACO: MICA considering Communication Uncertainties 105
7.1 MICACO 105
7.1.1 New Interaction Mechanism 105
7.1.2 CAV State Machine 108
7.1.3 ICA State Machine 109
7.2 Simulation 110
7.2.1 Simulation Setup 111
7.2.2 Simulation Results 112
7.3 Summary 115

8 Conclusion and Future Work 116
8.1 Conclusion 116
8.2 Future Work 118
8.3 Publications 119

References **120**

LIST OF TABLES

3 DTOT-based Intersection Traffic Management

3.1 Simulation results comparison. 40

4 Computational Complexity Improvements of DICA

4.1 Parameters used for various traffic volumes and patterns. (∗ Expected number of vehicles per 10 minutes.) 58

4.2 Computation time comparison between enhanced DICA and optimized traffic light 61

4.3 Average trip time comparison between major roads and minor roads in unbalanced traffic 62

5 Reactive DICA: an Approach for Expedited Crossing of Emergency Vehicles

5.1 Parameters used for various traffic volumes and patterns. (∗ Expected number of vehicles per 10 minutes.) 81

5.2 Computation time comparison between ES and GA 84

5.3 Comparison of maximum trip times of normal vehicles between DICA and R-DICA 86

6 MICA: Optimal Control of Autonomous and Connected Intersection Traffic

6.1 Parameters used for various traffic volumes and patterns. (∗ Expected number of vehicles per 10 minutes.) 100

6.2 Jain's fairness index of the optimized traffic light, DICA and MICA. 102

7 MICACO: MICA considering Communication Uncertainties

7.1 Parameters used for various traffic volumes and patterns. (∗ Expected number of vehicles per 10 minutes.) 110

7.2 Jain's fairness index 114

LIST OF FIGURES

3 DTOT-based Intersection Traffic Management

3.1 DTOTs of two conflicting vehicles. 26
3.2 Interaction between a CAV and an ICA. 27
3.3 Example situations of front vehicles. 32
3.4 A screenshot of simulation which illustrates a situation when vehicles with conflicting routes cross the intersection simultaneously. 41
3.5 The histogram of Trip Times of crossed vehicles in a simulation. . . 43
3.6 Comparison of Trip Times of 3 different cases between DICA and Concurrent Algorithm. 43

4 Computational Complexity Improvements of DICA

4.1 Two different cases for the shortest intersection crossing time (T_m) calculation 49
4.2 Approximate occupancy time interval calculation for a vehicle with the through route 51
4.3 Comparison of computation times 59
4.4 The number of vehicles which wait to cross the intersection over time. 61
4.5 Histogram of the inter-vehicle distances within the intersection. . . . 63
4.6 Performance comparison between the enhanced DICA and optimized traffic light 64
4.7 Flow rate ratio when traffic volume changes from 100 to 500 65

5 Reactive DICA: an Approach for Expedited Crossing of Emergency Vehicles

5.1 Three different situations for separation time. 69
5.2 Control flow diagram of the ICA in R-DICA. 72
5.3 An example of the permutation encoding scheme. 75
5.4 Example situation of vehicles whose sequence to be optimized in intersection space. 76
5.5 An example of one-point crossover. 77
5.6 An example of mutation. 78
5.7 Reactive traffic light diagram. 81
5.8 A series of screenshots of simulation. 83
5.9 Performance comparison of EVs. 84
5.10 Performance comparison of normal vehicles. 86

6 MICA: Optimal Control of Autonomous and Connected Intersection Traffic

6.1 Interaction between a CAV and ICA. 89
6.2 Example of conflicting area. 94
6.3 Examples of three types of front vehicles (we use the simplest intersection structure to save space). 96

6.4 Comparison of crossed vehicles' performance among the optimized traffic light, DICA and MICA 101
6.5 Histograms of trip times when employing the optimized traffic light, DICA and MICA 103
6.6 Overall Performance comparison between DICA and MICA 103

7 MICACO: MICA considering Communication Uncertainties

7.1 Interaction between a CAV and the ICA handling communication problems. 106
7.2 State machine of a CAV. 108
7.3 State machine of an ICA interaction instance. 109
7.4 Traffic control performance comparison between MICA and MICACO (a) throughput, (b) effective average trip time. 112
7.5 Performance comparison between the optimized traffic light algorithm and MICACO 113

CHAPTER ONE

Introduction

1.1 Motivation and Background

Recent years, due to the rapidly increasing demand for transportation from the larger and larger population, roads have become more and more congested. Careful city planning and smart traffic control strategies [2] can usually alleviate such transportation problems, but the unexpected growth of population and vehicle usage leads to persistent congestions. As a potential solution to address the serious congestion problems, autonomous transportation systems have attracted a lot of research and development efforts from academia, industry, and government. For example, during the mid-1990s the California PATH (Partners for Advanced Transportation Technology) launched the Automated Highway System (AHS) program [3] and the US DARPA (Defense Advanced Research Projects Agency) held a series of autonomous vehicle challenges during the 2000s [4] to take advantages in sensing and computation technologies. Also, many large companies and startups have already made decisions to hugely invest in developing their own self-driving vehicles or vehicles with advanced driving assistance systems [5]. However, despite many recent successful road testing results of several self-driving cars from companies like Waymo, Uber, Baidu, etc., it is hard to argue that the overall system-wide traffic safety, as well as throughput, will be improved substantially when we have a few Connected and Autonomous Vehicles (CAVs) among all other conventional vehicles. In fact, the potential of autonomous vehicles in terms of traffic efficiency and safety will be unleashed when most cars on roads are autonomous and connected [6]. Thus, in addition to many efforts to make today's traffic more efficient by improving utilization of traditional traffic infrastructure such as the works presented in [7–9], we believe that it is also very important to develop traffic control algorithms that take advantages of connectivity and autonomy of autonomous vehicles to prepare for the next generation transportation system. However, while there have been many efforts toward this direction, the development of safe and efficient autonomous transportation systems is still at its early stage. Among many research problems like vehicle path planning [10], autonomous parking control [11], collision avoidance [12, 13], relation between occupant experience and intersection capacity [14], intersection management of mixed traffic [15], intersection control based on real-time traffic surveillance [16–18], etc. that should be addressed toward this objective, we are particularly interested in addressing a problem of safe and efficient intersection crossing traffic management of autonomous connected traffic. Compared with highways, road intersections are more complicated places where accidents are more likely to happen and become the bottleneck of traffic performance improvement.

The problem to be solved in this field is how to improve the safety and throughput of intersection crossing traffic without using traffic lights or stop signs by leveraging the advantages of autonomous vehicles. In literature, there are a number of notable results for autonomous intersection crossing traffic management. Some researchers started their research from heuristic algorithms since those algorithms have a very little computational requirement which makes them good candidates for real-time applications if the performance is acceptable. Aaditya Prakash Chouhan and Gourinath Banda [19] proposed three different heuristic strategies to find collision free safe constant speed for a vehicle to cross an intersection while computation load is minimized whenever possible. [20] proposed a robust algorithm against external disturbances, model mismatches, nondeterministic delay of network and processing time of the intersection manager. Instead of using constant velocity, the algorithm let each vehicle creates an optimal position trajectory and tracks it. Turning vehicles are not bounded by the low speed before entering the intersection which increases the throughput of the intersection.

Many researchers have also investigated various optimization approaches to increase throughput. In [21], Lee et al. proposed an algorithm, called the Cooperative Vehicle Intersection Control (CVIC), which manipulates every individual vehicle's driving motion by providing them proper acceleration or deceleration rates so that vehicles can cross the intersection safely. Wu et al. [22] introduced a new intersection traffic management framework that is formulated as a combinatorial optimization problem and solved the problem approximately using the ant colony system algorithm [23]. Fei Yan et al. [24] combined vehicles whose routes are compatible with each other into mini groups and obtained an efficient vehicle passing sequence by their proposed genetic algorithm. A nonlinear programming formulation for autonomous intersection control was developed in [25] where the nonlinear constraints were relaxed by a set of linear inequalities. Most of them are centralized approaches in which control decisions are made typically by a central agent. Decentralized intersection control approaches have also been proposed in the literature. For example, [26] formulated a decentralized framework whereby each autonomous vehicle minimizes its energy consumption under the throughput-maximizing timing constraints and hard safety constraints to avoid rear-end and lateral collisions. A complete analytical solution of the decentralized problems was presented in the paper.

These approaches are trying to avoid the appearance of vehicles with conflicting routes inside an intersection at the same time to ensure safety which uses simple models that the entire intersection is considered as a conflict zone. Some researchers studied elaborate models that allow conflicting vehicles to exist inside an intersection simultaneously to further improve performance. To achieve this objective, most approaches discretized the intersection space into grid cells so that vehicles of conflicting routes can exist at the same time within an intersection but not within the same cell. In [27], Kurt Dresner and Peter Stone presented a reservation-based approach Autonomous Intersection Management (AIM) which allows route-conflicting vehicles enter the intersection simultaneously as long as they don't occupy the same cell at the same time. In AIM, vehicles request and receive time slots from the intersection during which they may pass. However, no global coordination is made for crossing vehicles to obtain optimal traffic flow. Following AIM, similar researches and improved approaches were proposed like expediting the crossing of emergency vehicles [28], determining the priority of requests using auctions [29, 30], optimization using integer program [31] and mixed integer linear program [32], etc. Representative centralized approaches also include auction-based intersection managements proposed in [29, 30]. A series of decentralized approaches only based on vehicle-to-vehicle (V2V) communications were proposed in [33–36] by Reza Azimi et al. In their papers, the intersection area is considered as a grid which is divided into small cells. Each cell in the intersection grid is associated with a unique identifier. One of their advanced intersection protocols named AMP-IP (Advanced Maximum Progression Intersection Protocol) allows the lower-priority vehicle to go ahead and cross the conflicting cell before the arrival of the higher-priority vehicle if there is enough time for the lower-priority vehicle to clear the cell. V2V and V2I were both used in the cooperative scheduling mechanism called TP-AIM [37] where for a considered vehicle, V2V provides the preceding vehicle's future motion information and V2I gives the assigned priority from the intersection manager.

Roughly speaking, these approaches are all based on the grid cell partitioning approach of an intersection space. In [27], the effect of the grid cell granularity on the computational efficiency of an intersection traffic management framework such as AIM was studied. As the paper pointed out, higher granularity gives more flexibility for better traffic throughput. However, the computational complexity increases proportionally to the square of the granularity. On the other hand, when the cell size becomes large for better computational efficiency, one can see that the intersection space is not utilized efficiently resulting in lower traffic throughput. Therefore, to overcome this trade-off issue between the granularity and computational efficiency of an algorithm, it might be a good alternative approach to utilize each vehicle's actual occupancy instead of grid cells to improve the overall traffic throughput. And this has motivated our researches on this topic.

There are many researches focusing on optimization of autonomous intersection control and lots of efforts on allowing route-conflicting vehicles to exist at a same intersection to improve efficiency. However, few researches have been done on combining two strategies, i.e. optimize the intersection control while allowing the simultaneous appearances of conflicting vehicles inside an intersection. Dai et al. [38] transformed the intersection control model into a convex optimization problem with an objective function of multiple criteria like safe speeds and accelerations to improve the quality of experience from a passenger's perspective. They linearized the collision avoidance constraints through scheduling the entering priority of vehicles at the intersection to reduce computational cost. However, this linearization may decrease performance since, for example, the scheduling requires the earlier arrival vehicle cross the collision area first for two possibly colliding vehicles without considering its speed. Reference [39] proposed a vehicle-intersection coordination scheme which generates smooth flows of traffic by preventing any pair of conflicting vehicles from approaching their Cross-Collision Point at the same time. Considering states of all vehicles, the paper formulated a constrained nonlinear optimization problem to minimize the risk function to generate safe trajectories of vehicles with unused time and space in the intersection space area reduced. Whereas, the algorithm needs more computational cost to coordinate more vehicles since they are optimizing the traffic flow of all approaching vehicles at the same time. And the optimization problem is nonlinear thus the solution is not guaranteed to be the global optimum. Based on conflict region concept, [40] formulated the intersection control problem as a mixed linear integer program which produced an arrival schedule composed by a desired time and speed of arrival at the intersection for every vehicle. Following this, [41] proposed three optimal control strategies to solve the motion planning problem to comply with the collision-free schedules at intersections. The strategies are different mostly on how strict they are about keeping the schedule. The results show that when the traffic volume is high, the strategy of following schedule strictly leads to both lowest arrival time and energy expenditure.

Most existing intersection control algorithms were proposed based on the assumption of perfect communication, i.e. they assume messages are delivered instantly to the receiver and no messages will be lost. In terms of intersection traffic control algorithms considering communication uncertainties, very few researches have been done in this field. A delay-tolerant intersection control algorithm was proposed by Bowen Zheng et al. [42] to guarantee the safety and liveness properties for typical four-way single-lane intersections. They used timeouts to help vehicles and intersection managers decide what to do next to ensure the safety (their behaviors were modeled as finite state machines). The results show that increasing communication delays may significantly decrease the intersection performance. The algorithm does not allow vehicles with conflicting movements to enter the intersection at the same time. The algorithm was expanded and generalized in [43] to handle multiple lane intersections and interconnected intersections. Some of the timing and security issues in transportation networks with VANET-based intelligent intersections were modeled and analyzed quantitatively in the paper. The paper also showed how imperfect communication will affect the performance and safety of a single intersection and a network of intersections. However, these approaches which handle communication uncertainties usually don't include optimization to further improve the traffic efficiency. The study of optimal intersection control considering imperfect communication is still at its very early age.

1.2 Contribution

As an approach to address the granularity issue mentioned in Section 1.1, we propose a novel intersection traffic management scheme based on the idea of the *Discrete-Time Occupancies Trajectory (DTOT)*. Conceptually, a DTOT is a discrete-time sequence of a vehicle's actual occupancy within an intersection space. The proposed DTOT-based Intersection Control Algorithm (DICA) allows the flexibility that each vehicle can choose its path as well as motions along the path that a CAV wants to take to cross an intersection. A CAV who is approaching an intersection will check whether it is the *Head Vehicle* on its lane. A CAV becomes a head vehicle only when no vehicles exist between the CAV and the intersection, or the vehicle which is immediately in front of the CAV has begun to enter the intersection. If the CAV is a head vehicle, it will propose its request to the intersection. From the request, a vehicle's enter time and exit time to the intersection, route and other detailed information of passing the intersection can be found. Then the intersection responds with a DTOT to the vehicle to avoid collisions and improve the overall intersection throughput. The DTOT corresponds to a trajectory with achievable speed and acceleration for the CAV. If the CAV is not a head vehicle, it will just follow other cars. Thus, the management scheme is only dealing with head vehicles which reduces largely the communication needs for vehicles and the computational complexity of the central control agent. It is assumed that a CAV always want to go through an intersection as quickly as possible. So CAVs in DICA always try their best to reach maximum allowed speed within the intersection.

Then, we provide an in-depth analysis of the original DICA to show that it satisfies the liveness property in terms of deadlock as well as starvation issues and also to derive the overall computational complexity of the algorithm. We propose several computational approaches to improve the overall computational efficiency of the DICA and also enhance the algorithm accordingly so that it can be operated in real-time for autonomous and connected intersection crossing traffic management. We also present simulation results that show the improved computational efficiency of the enhanced algorithm and the overall throughput performance in comparison with that of an optimized traffic light control.

Following computationally enhancing DICA, we extend DICA approach to include emergency vehicles in the traffic to be controlled. Our goal is to let emergency vehicles cross intersections as fast as possible while maintaining adequate traffic performance. In this dissertation, we assume that emergency vehicles are taking normal routes which means that they will not travel in a wrong lane. A genetic algorithm is proposed to find the optimal passing sequence of vehicles whose trajectories can be rearranged. This optimal sequence aims to make the emergency vehicles cross the intersection in the fastest way. When there is no emergency vehicle inside the communication region, vehicles are controlled by DICA. Thus, the proposed algorithm is called Reactive DICA. Among many sequence forming approaches [22, 24, 29, 44] in literature, [24] is the most similar approach with ours which also proposed a genetic algorithm to form vehicle sequences. However, unlike the approach proposed in this dissertation, they are essentially not allowing vehicles with conflicting routes to be inside the intersection at the same time.

We also address the optimization problem of safe and efficient intersection crossing traffic management which provides the optimal trajectory for every head vehicle. The overall idea is similar to that in [39] while the employed approaches are different, i.e. [39] uses MPC and our paper uses Mixed Integer Programming (MIP). The proposed algorithm is called *MICA* (MIP-based Intersection Coordination Algorithm) since it is optimizing every new head vehicle's trajectory using MIP under various constraints. The constraints on the new head vehicle's possible motions are modeled using binary variables. Potential collisions with the immediate front vehicles are also avoided by the modeled constraints for such vehicles. This optimal control algorithm allows later confirmed vehicle to cross the intersection earlier as long as no collisions will happen to increase the overall performance. A huge improvement of performance is observed compared with our DICA algorithm which already outperforms conventional traffic light control and Concurrent Algorithm [1].

Finally, based on our previous research on intelligent intersection coordination algorithms [45–48], we present the preliminary results of an algorithm to handle communication uncertainties properly while controlling intersection traffics. The algorithm is optimizing every new head vehicle's trajectory under imperfect communication using Mixed Integer Programming (MIP) with various constraints thus named MICACO (MIP-based Intersection Coordination Algorithm considering COmmunication uncertainties). MICACO ensures safety and finds the optimal trajectory for every head vehicle under the realistic communication environment where there are packet delays and losses.

1.3 Organization of the Dissertation

The remainder of this dissertation is organized into seven chapters.

- In Chapter 2, we review the literature on autonomous transportation, especially on autonomous intersection control.
- In Chapter 3, we introduce the assumptions and interactions between a vehicle and an intersection in DICA. Then we explain the essential functions in DICA

in detail and analyze the liveness of DICA. Finally, DICA is simulated and compared with Concurrent Algorithm [1] to validate its performance.

- In Chapter 4, we analyze the computational complexity of DICA in detail first and then proposed several computational techniques to improve overall computational efficiency. The improved performance is validated through comparisons with original DICA and an optimized traffic light control algorithm.
- Chapter 5 describes Reactive DICA especially the proposed genetic algorithm in detail. Simulation results of Reactive DICA are compared with a reactive traffic light algorithm to show its performance.
- In Chapter 6, we formulate the trajectory optimization problem with each constraint explained detailedly. Then we propose MICA algorithm incorporating the MIP optimization approach. The efficiency of MICA is evaluated against DICA and an optimized traffic light algorithm through extensive simulations.
- Based on MICA, Chapter 7 presents the preliminary result on a new algorithm MICACO which handles imperfect communications i.e. packet delay and loss. The efficiency of MICACO is evaluated through simulation results compared with MICA and an optimized traffic light algorithm.
- Finally, Chapter 8 draws the conclusion of the dissertation and provides potential future work.

CHAPTER TWO

Literature Review

2.1 Autonomous Control of Highway Systems

Highway congestion has brought intolerable burdens to most urban residents. Governments have made many attempts including building more roads and raising tolls or other related taxes, promoting public transportation or better vehicle occupancy (carpooling) and developing a high-speed communications network that in many ways to reduce travel demand [49]. In the meantime, researches of Intelligent Vehicle/Highway System (IVHS) provide new ways to ease the congestion on highways. There are mainly two types of methods to increase highway capacity while ensuring safety, *1.* vehicle platooning, represented by the AHS which had been deeply developed at the University of California Partners for PATH program, in cooperation with the State of California Department of Transportation (Caltrans) and the United States Federal Highway Administration (FHWA) since 1989 [3, 49]. *2.* algorithms and protocols for each individual vehicle to drive fast and safely, represented by references [50, 51].

2.1.1 Vehicle Platooning

Organizing traffic into platoons which are groups of up to 20 tightly spaced cars can increase highway capacity greatly. 8000 vehicles per hour per lane could be achieved while today's highways with manually controlled vehicles only have 2000 capacity [3]. Platooning also has a benefit of reducing aerodynamic drag because vehicles are tightly spaced. People cannot react quickly enough to drive safely with such small headways, so every vehicle in a platoon should be automated. In the approach of [49], a car joins a platoon when it enters the AHS and exits as a one-car platoon or part of an exiting platoon when it approaches its destination. Inside AHS, the car is controlled automatically by computers and a series of maneuvers are executed by the car including splitting from and joining platoons and lane changing to navigate through the highway network. [49] introduced the finite state machine method which is used for the execution of maneuvers. Join control law, platoon leader control law and other laws were given in [3] to ensure collision avoidance of AHS. What is noteworthy is that the onboard vehicle control system is a hybrid control system which consists of a discrete event dynamical system (the coordination layer) and a continuous-time dynamical system (the regulation and physical layers).

The platoon concept is useful but does not take full advantage of the microscope centralized possibilities because it does not consider the interaction of all vehicles [51]. In a platoon, only leaders (and free agents) can initiate maneuvers, while followers maintain platoon formation at all times [3].

2.1.2 Algorithms and Protocols for Individual Vehicles

For an individual autonomous vehicle, many algorithms and protocols have been proposed for its longitudinal and lateral motion on a highway system while the vehicle's safety is ensured.

Longitudinal Control

The longitudinal control of a vehicle is to control the vehicle's speed and its adaptation to road features by adjusting the throttle [52] and the brake pedal as needed [53]. A longitudinal control system will handle issues like nonlinear vehicle dynamics, operations of vehicles from high-speed cruising to a complete full-stop, and other operations under the communication constraints [54]. Following we describe several different controllers that are designed to ensure the safety of longitudinal motion of each vehicle.

Reference [55] presented the theorem for perpetual safety of two cars on a lane that if each car makes the worst-case assumption about the car immediately in front of it, then the safety of multiple cars on a lane can be guaranteed. Reference [56] used a simple discrete-time kinematic model to represent the longitudinal motion of a vehicle, $x_{t+h} = f(x_t, v_t) = x_t + v_t h$, where x_t, v_t, h correspond to x-axis (forward direction) position, linear velocity of a vehicle at time t, and sampling period respectively. The paper formulated an MPC problem for the vehicle to ensure the generation of safety-guaranteed motions.

Lateral Control

Typically lane changing provides a maneuver for a fast-moving vehicle to pass a slow one, which can be observed everywhere on the highway [57]. A framework of lane change decision-making under typical urban driving conditions was proposed by Gipps [58], which includes the effects of traffic lights, obstacles, and types of surrounding vehicles. Taking into account the potential conflict objectives and assuming logical driver behavior, the model focuses on the decision-making process.

Reference [56] constructed the MPC problem for Lane Change of a vehicle on multi-lane traffic and introduced the Lane Change Protocol that a vehicle can initiate its lane change action only when its state satisfies certain relations with neighboring vehicles. The paper also proposed the Yield Protocol which is incorporated into the MPC motion planning framework to ensure the liveness of Lane Change. A novel MPC-based approach for active steering control design was presented in [50]. Experimental results showed that a vehicle under MPC feedback policy was capable of stabilizing with a speed up to 21 m/s on slippery surfaces such as snow-covered or icy roads. The paper designed and experimentally tested three types of MPC controllers including a nonlinear MPC, an LTV MPC, and an LTV MPC controller of low order.

2.2 Autonomous Intersection Control

Intersection traffic control is more of importance since it is more likely to have accidents at intersections compared with highways. Besides accidents, the trip delays from the impact of intersections also lead to waste of human and natural resources. To tackle intersection control problems, many solutions have been proposed in the literature. The main problem is how to provide an intersection control algorithm with guaranteed safety and improved efficiency. Currently, all intersections are controlled by either traffic lights or stop signs in the US. Other countries are also using similar mechanisms to control intersection traffics. While stop signs are useful for intersections of light traffic, traffic lights are able to control more congested intersections. The performance of traffic lights highly depends on the cycle lengths of different phases which can be optimized based on traffic conditions. The optimization can be done offline to have fixed cycle durations, for example, Jose Garcia-Nieto et al. [59] optimized traffic light cycle offline based on particle swarm optimization. However, traffic lights with fixed cycle durations will make vehicles wait for the green light even there are no vehicles of conflicting movements. Some researchers are also working on adaptive or dynamic traffic lights which adapt the traffic light cycle online based on real-time traffic conditions [8, 60, 61] and optimal control of autonomous vehicles based on traffic light information [62]. To leverage the computing, sensing, communicating capabilities of future fully autonomous vehicles, more researches are being done on algorithms to control intersection traffic without traffic lights or stops signs. Our algorithms are also only dealing with autonomous vehicles, hence we mainly review intersection control approaches of autonomous vehicles in this section.

2.2.1 Centralized Control

Most of existing intersection control algorithms in literature are centralized approaches in which control decisions are made typically by a central agent. In [27], Kurt Dresner and Peter Stone presented a reservation-based approach Autonomous Intersection Management (AIM) which allows route-conflicting vehicles enter the intersection simultaneously as long as they don't occupy the same grid cell at the same time. In AIM, vehicles request and receive time slots from the intersection during which they may pass. Later they improved AIM by enabling vehicles to accelerate in the intersection area [63]. However, no global coordination is made for crossing vehicles to obtain optimal traffic flow in both papers. Following AIM, similar researches and modified approaches were proposed like expediting the crossing of emergency vehicles [28], determining the priority of requests using auctions [29,30], optimization using integer program [31] and mixed integer linear program [32], etc. Dustin Carlino, Stephen D. Boyles and Peter Stone [29] proposed an auction-based intersection traffic management in which vehicles can bid for a faster crossing of an intersection. They also designed a benevolent system agent to prevent the scheme from being biased towards wealthy driver agents. Levin et al. [31] studied the approach to choose the optimal subset of vehicles to move at every time step by formulating an Integer Program (IP). Arbitrary objective functions can be admitted by the IP so a more general class of policies can be applied to optimize the order of vehicles to cross the intersection. They also proposed AIM* which uses a mixed integer linear program to optimally choose vehicle reservations and ensure enough separation at all conflicting points that vehicles might intersect [32]. A rolling horizon algorithm was also presented to extend AIM* to a larger number of vehicles in real time. Matthew Hausknecht et al. [64] investigated the performance of AIM for networks of autonomous intersections and the impact of dynamically reversing traffic flow along certain lanes in a road network.

Reference [55] designed smart intersections where vehicles negotiate the intersection crossings through interaction of centralized and distributed decision making. The route that was taken by a car i is described by an ordered pair R(i)=(O(i), D(i)), of origin and destination, respectively. A scheme was proposed which consists of a time-slot allocation intersection crossing algorithm, and an algorithm for updating failsafe maneuvers of each vehicle so as to avoid collisions while crossing an intersection. A car will enter the intersection only if it can exit safely. An interesting result about collision avoidance at intersections is shown in [13], that it is an NP-hard problem to check the membership in the maximal controlled invariant set, which is the largest set of states for which there exists a control that avoids collisions. An algorithm with provable error bounds is proposed to solve such a problem approximately in polynomial running time. The paper provides and proves a tight bound on the approximated solution.

In recent years, many researches have been done to improve intersection control performance by using optimization approaches. In [21], Joyoung Lee and Byungkyu Park proposed an algorithm, called the Cooperative Vehicle Intersection Control (CVIC), which manipulates every individual vehicle's driving motion by providing them proper acceleration or deceleration rates so that vehicles can cross the intersection safely. They formulated an optimization problem that has the objective function of minimizing the total length of overlapped vehicular trajectories. Later Joyoung Lee et al. [65] studied the CVIC framework in the case of a corridor consisting of multiple intersections. Wu et al. [22] introduced a new intersection traffic management framework that was formulated as a combinatorial optimization problem and solved the problem approximately using the ant colony system algorithm [23]. In [66], authors proposed Intersection Management using Cooperative Adaptive Cruise Control (iCACC) which models an intersection as a group of possible conflicting points and optimizes the arrival of vehicles at those points to minimize the total intersection delay of all approaching vehicles. Fei Yan et al. [24] combined vehicles whose routes are compatible with each other into mini groups and obtained an efficient vehicle passing sequence by their proposed genetic algorithm. Jean Gregoire et al. [67] proposed a cooperative motion-planning algorithm which decomposes path-velocity to have the optimal coordination among crossing vehicles. Kyoung-Dae Kim and P.R. Kumar [1, 56] developed a Model Predictive Control (MPC) framework which integrates decisions made by the intersection in the discrete domain for vehicle ordering with decisions made by each vehicle in the continuous domain to dynamically generate a sequence of collision-free motions. Their paper considered two algorithms, a simple First In First Out (FIFO) Crossing algorithm and a Concurrent Algorithm, and shown the corresponding system-wide safety and crossing traffic liveness. A nonlinear programming formulation for autonomous intersection control was developed in [25] where the nonlinear constraints were relaxed by a set of linear inequalities. While the objective function of the optimization problem in [25] involves the travel time, other studies [38, 39] are trying to solve similar control problem using an objective function with multiple criteria like safe speeds and accelerations while avoiding collisions. S. Alireza Fayazi et al. proposed an urban intersection traffic management scheme which regularly calculates an optimal arrival schedule of approaching vehicles to ensure safety and largely reduce the number of stops and intersection delays [68, 69]. Their method encourages vehicle platoon formation implicitly and they also modified their design to support mixed traffic in [69]. A polling-systems-based algorithm was proposed in [70] with provable guarantees on safety and performance. In the algorithm, a rigorous upper bound is provided for the expected waiting time.

2.2.2 Decentralized Control

Compared with centralized control approaches, infrastructure support is not needed in decentralized control. And also the single point of failure problem does not exist for decentralized algorithms.

Since 2011, researcher Reza Azimi has proposed a series of reliable intersection protocols that use only vehicle-to-vehicle (V2V) communications. The installation of a centralized infrastructure at each intersection is impractical due to the high overall system cost. Azimi advocates the use of V2V communications and distributed intersection algorithms that run in each vehicle. He designed the vehicular network protocols that integrate mobile wireless communications standards such as Dedicated Short Range Communications (DSRC) and Wireless Access in a Vehicular Environment (WAVE). Collision Detection Algorithm for Intersections (CDAI) which uses a priority-based policy, Stop-Sign Protocol (SSP), Throughput Enhancement Protocol (TEP) and Throughput Enhancement Protocol with Agreement (TEPA) are proposed in [33]. SSP is similar to actual stop-sign situation that every vehicle must stop before an intersection when there is a stop-sign. Using TEP, vehicles stop at the intersection only if the CDAI predicts a collision and assigns a lower priority to them based on the message it receives from all vehicles at the intersection. TEPA is built on TEP and is explicitly designed to handle lost V2V messages.

In [34], Azimi et al. defined an intersection as a perfect square box that predefines the entry and exit points for each lane to which it is connected. The intersection area was discretized as grid cells and each cell was associated with a unique identifier. More advanced protocols CC-IP (Concurrent Crossing-Intersection Protocol) and MP-IP (Maximum Progression-Intersection Protocol) which were improved from [33] were proposed. In CC-IP, a CROSS message is broadcasted when a vehicle enters the intersection area, other vehicles can simultaneously pass through the intersection if they detect no potential collision with the vehicle which is already crossing the intersection, otherwise the vehicle stops before the intersection area. However, in MP-IP, the vehicle which has the lower priority uses CDAI and finds out the first common cell (trajectory intersecting cell, abbreviated as TIC) with the crossing higher-priority vehicle. Then, instead of not entering the intersection as in CC-IP, the vehicle stops at the cell just before entering the TIC.

Reference [35] illustrates more advanced protocols about intersection using vehicular networks based on the work of [33, 34]. After detailed describing of MP-IP and AMP-IP (Advanced Maximum Progression Intersection Protocol), the paper proves that these protocols avoid deadlock situations inside the intersection area. The improvement of AMP-IP is that the protocol allows lower priority vehicles to advance and cross the conflicting cell before the higher priority vehicle arrives. A Safety Time Interval of 2s is used to help lower-priority vehicles decide whether they can go through the conflicting cell safely. Simulation results show that the latest V2V intersection protocol AMP-IP yields over 85% overall performance improvement over the common traffic light models. In [71], Azimi also investigates the use of their proposed V2V-intersection protocols for autonomous driving at roundabouts. The improvement in safety and throughput when the intersection protocols are used to traverse roundabouts is also quantified.

In the approach [72] proposed by Wu et al., the estimated arrival time is shared wirelessly among vehicles to obtain the best passing sequence. Yue J. Zhang et al. proposed a decentralized optimal control framework which coordinates a continuous flow of CAVs crossing two adjacent intersections online [73]. The obtained solution gives the optimal acceleration/deceleration for each vehicle at any time to minimize fuel consumption. However, the optimal control solution's feasibility depends on the initial condition of each CAV when they enter the so-called control zone of each intersection. Later, they showed that there exists a feasibility region defined by every CAV's arrival time and speed, and can be fully determined to the CAV before it enters the control zone [74]. [26] formulated a decentralized framework whereby each autonomous vehicle minimizes its energy consumption under the throughput-maximizing timing constraints and hard safety constraints to avoid rear-end and lateral collisions. A complete analytical solution of the decentralized problems was presented in the paper.

Aashiq Parker and Geoff Nitschken [75] compared AIM with their intersection management scheme which uses a decentralized neuro-evolution approach to adapt vehicle controllers for groups of autonomous vehicles. The synthesis of collective driving behavior is automated by neuro-evolution and the corresponding efficacy is demonstrated.

2.2.3 Emergency Vehicle Handling

Human lives and the amount of financial loss highly depend on the response time of emergency vehicles (i.e. from the time the emergency service is called to the time help is offered). The travel time of emergency vehicles to the accident scene is critical to the response time. So it is very useful and helpful to reduce travel time of emergency vehicles on roads, especially on intersections where congestions are more likely to happen. The survival chance of injured people in an accident falls sharply if they reach the operating table later than 60 minutes after the accident [76]. Hence, shortening the travel times of crossing intersections for emergency vehicles will help to save lives. In reality, the current way to handle emergency vehicles is similar to using Vehicle-to-Vehicle (V2V) communication (siren and lights) to warn non-emergency vehicles on roads to yield to the emergency vehicle. Some drivers cannot respond quickly to the warnings which may result in additional time delay for emergency vehicles and even serious accidents. In most of the existing approaches [28, 77, 78] for emergency vehicles, the travel times of non-emergency vehicles will be affected significantly and the advantages of autonomous vehicles are not fully taken.

Many studies have been done to allow emergency vehicles to have a faster travel across intersections. Based on MAS (Multi-Agent System), [77] introduced a state machine for the intersection controller to change traffic signal status according to lane occupations when an emergency vehicle is approaching. Some researchers have explored the priority evacuation of emergency vehicles under an autonomous and connected traffic environment. Wantanee Viriyasitavat and Ozan K. Tonguz proposed an intersection control system that only uses Vehicle-to-Vehicle (V2V) communication to give emergency vehicles priority of crossing [78]. The paper proposed that at an intersection, a leader should be elected from all approaching vehicles to serve as the temporary traffic light infrastructure and stop at the intersection to coordinate the traffic. The green signal is always given to the lane of detected emergency vehicles and through coordination "green-wave" signals are displayed for the emergency vehicles to let them move at a faster speed. Kurt Dresner and Peter Stone proposed a simple way to deal with emergency vehicles under their intersection control framework AIM (Autonomous Intersection Management) [28]. Their algorithm only grants reservations to vehicles in the lanes that have approaching emergency vehicles which allow the emergency vehicle to continue on its way relatively unhindered.

2.2.4 Communication Uncertainties

Most existing intersection control algorithms in literature were proposed based on the assumption of perfect communication, i.e. they assume all messages are delivered instantly from a sender to a receiver and no message will be lost. To handle communication delay, Crossroads algorithm [79] synchronizes all vehicles' local clocks with the intersection manager and then let vehicles request their target velocities. Thus both the intersection manager and the requesting vehicles have the same notion of time which avoids the use of extra buffer due to the uncertainty of message trip delays. However, the algorithm requires constant crossing velocity which results in slow speeds for turning vehicles. To improve Crossroads, Mohammad Khayatian et al. [20] proposed a time and space aware algorithm which is robust against external disturbances, model mismatches, nondeterministic delay of network and processing time of the intersection manager. Instead of using constant velocity to cross [79], the algorithm let each vehicle computes an optimal reference position trajectory and uses a PID controller to track it. Thus, turning vehicles are not bounded by low speeds before entering the intersection which increases the throughput of the intersection from that of [79].

Bowen Zheng et al. [42] proposed a delay-tolerant intersection control algorithm which guarantees the safety and liveness of a typical four-way single-lane intersection. The behavior of vehicles and the intersection manager are modeled as finite state machines which use timeouts to help on decisions like what to do next to ensure safety. The results show that increasing communication latency can significantly reduce intersection performance. Vehicles with conflicting movements are not allowed to enter an intersection at the same time. The algorithm was extended and generalized in [43] to handle multi-lane intersections and intersection networks. This paper models and quantifies some time and security issues in the traffic network based on VANET intelligent intersection. It also demonstrated how imperfect communications will affect the performance and security of an individual intersection and intersection networks. Based on DSRC, Rusheng Zhang et al. proposed virtual traffic lights algorithm [80] which puts traffic control devices inside vehicles and elects a vehicle to act as the traffic light which needs to stop before the intersection to control all vehicles approaching the same intersection.

2.2.5 Mixed Traffic

Most above intersection control mechanisms don't work with human-driven vehicles who are hard to predict and coordinate with. Kurt Dresner and Peter Stone [28, 81] augmented AIM to allow human-driven vehicles by periodically cycling a green light for a road or lane based on the percentage of human-driven vehicles in the traffic. An interesting problem of semiautonomous multivehicle safety has been studied by Rajeev Verma and Domitilla Del Vechhhio [82]. They mainly made research about the problem of collision avoidance between autonomous vehicles and human-driven vehicles. And a formal hybrid control approach to design semiautonomous multivehicle systems that are guaranteed to be safe is given. Based on a spatio-temporal reservation scheme, Luis Conde Bento et al. [83] proposed to reserve all possible space for a legacy vehicle to ensure safety. Xiangjun Qian et al. [44] proposed that legacy vehicles can use car-following to cross and avoid collisions with other autonomous vehicles. In Qian's algorithm, a legacy car can cross the intersection if there is another autonomous vehicle to follow or all other higher-priority vehicles have passed the intersection. Their algorithm is based on the assumption that legacy vehicles are able to keep a safe distance from leading vehicles. Some researchers also studied the control mechanism when there is only part of the traffic are autonomous vehicles [84].

2.2.6 Machine Learning

Machine learning is an emerging aspect in traffic management and control, which also shows its capability in different areas such as traffic coordination with power [85], motion prediction [86], power system [87–90], medical science [91], energy [92–96], etc. Kurt Dresner and Peter Stone [97] proposed a learning-based intersection control mechanism which switches policies online to best suit traffic conditions. Their switching mechanism performs much better than previous static configured intersection control policies. Matteo Vasirani and Sascha Ossowski [98] studied multiagent learning on autonomous intersection control where they designed a learning mechanism for intersection managers such that a global profit increase of the manager team is able to result in an average trip time decrease for drivers. Several attempts have been made by researchers [99–101] to apply deep reinforcement learning on traffic control when all vehicles are detected. Rusheng Zhang et al. [102] proposed a deep Q-learning to control traffic under partial observation of vehicles, i.e. not all vehicles are equipped with DSRC. Their simulation results show that reinforcement learning has the ability to effectively reduce the average delay of vehicles due to intersection even with a low DSRC penetration rate.

CHAPTER THREE

DTOT-based Intersection Traffic Management

As introduced in Chapter 1, DTOT is the abbreviation of Discrete-Time Occupancy Trajectory which is a sequence of a vehicle's actual occupancy within an intersection space. Based on the concept of DTOT, we propose a novel intersection control algorithm named DICA (DTOT-based Intersection traffic Coordination Algorithm) in this chapter.

3.1 Assumptions

In this section, we introduce the basic idea and algorithm of DICA that is developed for autonomous and connected intersection crossing traffic in which all vehicles are Connected and Autonomous Vehicles (CAVs) and capable of wireless vehicular communication. We assume that an intersection has the wireless communication capability as well as a computation unit so that it can exchange information with vehicles and perform necessary computations to coordinate vehicles to cross the intersection safely. In DICA, there is no traffic light that controls the intersection crossing traffic. Instead, each vehicle communicates with the *Intersection Control Agent* (ICA), to get permission to access the intersection. As shown in Figure 3.1, an intersection consists of two regions. The bigger region in the figure, which we call the *communication region*, is defined by the wireless vehicular communication range. The smaller region in the figure, which we call the *intersection region*, is the area within an intersection that is shared by all roads connected to the intersec-

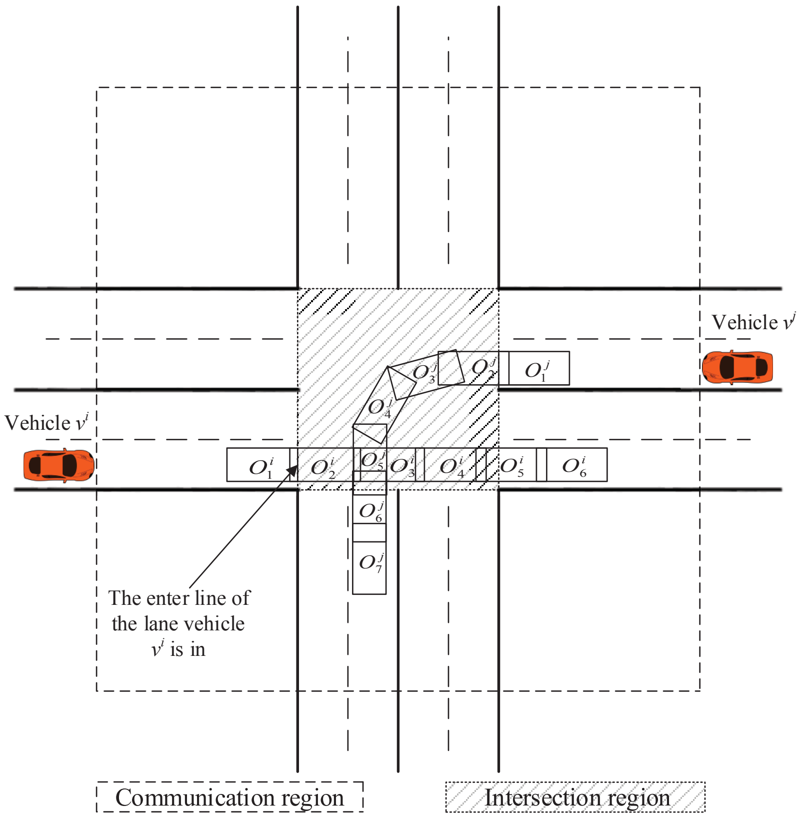

Vehicle v^j
Vehicle v^i
O_1^j O_2^j O_3^j O_4^j O_5^j O_6^j O_7^j
O_1^i O_2^i O_3^i O_4^i O_5^i O_6^i
The enter line of the lane vehicle v^i is in
Communication region
Intersection region

Figure 3.1: DTOTs of two conflicting vehicles. (O_q^p represents the q-th occupancy in a vehicle v^p's DTOT. Note that occupancies in this figure are intentionally made very sparse for clear illustration purpose. DTOT starts with the occupancy in which the vehicle's front bumper first contacts the enter line of its lane of an intersection, and ends with the occupancy that the vehicle is completely out of the intersection region.)

tion. We also assume that each vehicle is equipped with an RFID (Radio Frequency IDentification) chip and there are detectors installed at the entrance of the communication region so that the ICA can detect each vehicle's VIN (Vehicle Identification Number), the lane on which a vehicle is approaching an intersection, and the time when a vehicle enters the communication region. Since all vehicles are autonomous, we assume that each vehicle can obtain its position, speed, and the relative distance to an intersection precisely and also can avoid collisions with other vehicles autonomously when it is approaching an intersection. With regard to wireless vehicular connectivity, we only require information exchange between CAVs and the ICA. Thus, there is no V2V communication.

3.2 Interaction between an ICA and a CAV

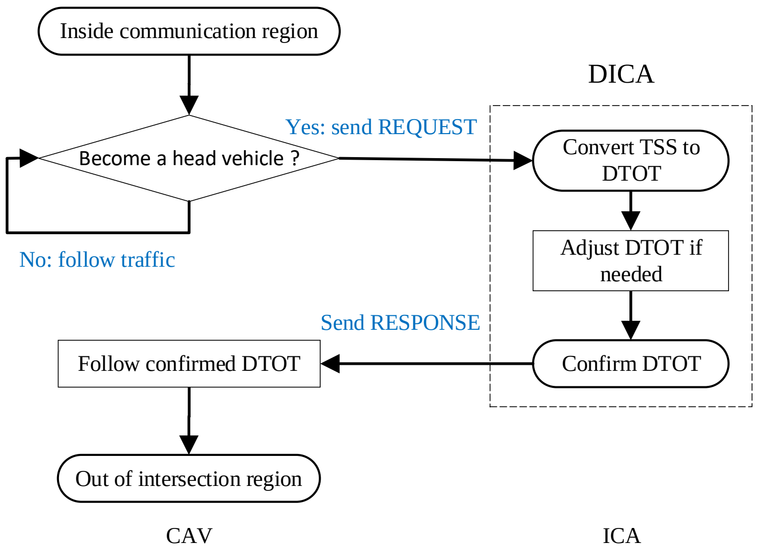


Figure 3.2: Interaction between a CAV and an ICA.

A CAV is considered a *head vehicle* in its lane if there are no vehicles in front of it or the vehicle which is immediately in front of it has begun to enter the intersection region. As shown in Figure 3.2, the interaction between a CAV and the ICA is initiated from the CAV, when it becomes a head vehicle, by sending a REQUEST message to reserve a sequence of spaces and times to cross the intersection. The ICA knows whether a vehicle is a head vehicle or not according to the list of vehicles for each lane. Thus, a REQUEST message not from a head vehicle will be neglected by the ICA. The list can be constructed in the ICA since, as explained earlier, the ICA knows each vehicle's VIN, the lane on which the vehicle is approaching, and the time when a vehicle passes a detector installed at the boundary of the communication region of an intersection. Each REQUEST message contains information that is necessary to reserve the space and time within the intersection region to cross the intersection such as (i) the VIN, (ii) the Vehicle Size (VS), and (iii) the Timed State Sequence (TSS). The VS is simply the length and width of the vehicle and the TSS is the discrete time state trajectory of the vehicle starting from the entering moment of an intersection region to the moment when the vehicle crosses the intersection region completely. Note that it is implicitly assumed that each discrete time state of a vehicle in TSS is also timed. This means that if a vehicle state $\mathbf{x}_t$ is given, then we can say that a vehicle possesses the state $\mathbf{x}$ at time t. For simplicity of our discussion, we assume that the state $\mathbf{x}$ of a vehicle consists of the (x, y) coordinate of the vehicle's location and the orientation θ. We also assume that, while it is possible that each vehicle can have different sampling period to generate its TSS, all vehicles use the same sampling period which is small enough to generate a close approximation of the vehicle's actual continuous motion within an intersection.

The ICA converts the TSS to the corresponding DTOT using the VS information which is also contained in the received REQUEST message. The DTOT is simply a sequence of timed rectangular spaces that a vehicle needs to occupy within an intersection region to cross the intersection. Now, the ICA uses all confirmed DTOTs to adjust the requested DTOT to avoid collisions if needed. The ICA then converts the collision-free DTOT to TSS and sends it back to the vehicle using a RESPONSE message which contains (i) the VIN and (ii) TSS so that the vehicle can follow the confirmed DTOT to cross the intersection. More detailed explanation on how to process the requested TSS to generate a confirmed DTOT is presented in the following section. In the sequel, we say that a vehicle is a *confirmed vehicle* if it has received a confirmed DTOT from the ICA. And we assume that every vehicle is able to follow the confirmed DTOT precisely. In practice, vehicles will have tracking errors to follow a given DTOT. To avoid potential collisions with other vehicles, we can increase the size of every occupancy in the DTOT by the upper bound of tracking errors. Since the focus of this chapter is to develop an algorithm for the ICA for safer and higher throughput intersection crossing traffic, we simply assume that we have an ideal wireless vehicular communication performance such that all REQUEST and RESPONSE messages are exchanged correctly and timely. However, it is important to note that, despite such an ideal communication assumption, our DTOT-based algorithm can still be applicable in practice with small modifications of the algorithm to take into account the communication unreliability. For instance, typically we may face two problems (i.e. package delay and lost) to handle the imperfect communications existing between CAVs and the ICA in real situations. We could use the upper bound of the package delay to extend every occupancy in a DTOT which is safe for vehicles but a little bit conservative. For package lost problem, an ACK message can be added to confirm the delivery of REQUEST and RESPONSE messages whose details can be found in the next section. A CAV will send REQUEST again if it does not receive the ACK message from the ICA. The same strategy could be applied to the ICA and RESPONSE message.

Algorithm 1 DICA (**D**TOT-based **I**ntersection traffic **C**oordination **A**lgorithm)

```
1: Let $\mathcal{S}$ be the set of confirmed vehicles and $n = |\mathcal{S}|$.
2: Let $v^i$ be the vehicle to be considered for confirmation.
3: Convert $TSS(v^i)$ to $DTOT(v^i)$
4: Call checkFV($\mathcal{S}, DTOT(v^i)$) $\to DTOT(v^i)$
5: Call getCV($\mathcal{S}, DTOT(v^i)$) $\to \mathcal{C}$
6: while $\mathcal{C} \neq \emptyset$ do
7:     Pop the first vehicle in $\mathcal{C} \to v^j$
8:     Call updateDTOT($DTOT(v^i), DTOT(v^j)$) $\to DTOT(v^i)$
9:     Call getCV($\mathcal{S}, DTOT(v^i)$) $\to \mathcal{C}$
10: end while
11: Store $DTOT(v^i)$ for vehicle $v^i$
12: Convert $DTOT(v^i)$ to $TSS(v^i)$
13: Send $TSS(v^i)$ to vehicle $v^i$
```

3.3 DTOT-based Intersection Traffic Coordination

The ICA processes a REQUEST message from a head vehicle according to the procedures shown in Algorithm 1 which we call the DTOT-based Intersection traffic Coordination Algorithm (DICA). As shown in the algorithm, we use TSS(v) and DTOT(v) to denote the TSS and DTOT for a vehicle v respectively. We also use $\mathcal{S}$ to denote the set of vehicles which have already been confirmed at the time when a REQUEST message is being processed. We say that two vehicles are *space-time conflicting* if their trajectories are conflicting not only in space but also in time. More precisely, two vehicles are considered to be in space-time conflict in our algorithm when their DTOTs have at least one pair of occupancies that conflict in both space and time. We use another set $\mathcal{C}$ in Algorithm 1 to represent the subset of $\mathcal{S}$ which contains the set of vehicles whose confirmed DTOTs have space-time conflict with the DTOT of the vehicle that is currently being processed for confirmation. Vehicles in $\mathcal{C}$ are ordered in ascending order of a certain attribute of their confirmed DTOTs. To explain this attribute more clearly, let us consider a situation when DICA processes a vehicle v^i's DTOT and there are two vehicles v^j and v^k in the set $\mathcal{C}$. Now let us suppose that DTOT(v^j) starts to space-time conflict with DTOT(v^i) from its n-th occupancy and DTOT(v^k) starts to space-time conflict with DTOT(v^i) from its m-th occupancy. If we use O^p_q to denote the q-th occupancy within DTOT(v^p) and $\tau(O^p_q)$ be the time when the vehicle v^p occupies O^p_q, then we say that, in this particular situation, $\tau(O^j_n)$ is the first time at which v^j starts to collide with v^i. Similarly, $\tau(O^k_m)$ is the time at which v^k starts to collide with v^i. In the sequel, this specific time instant for each vehicle in $\mathcal{C}$ is represented by the variable '*firstTimeAtCollision*'. In this particular situation, $\tau(O^j_n)$ and $\tau(O^k_m)$ are denoted by $v^j.firstTimeAtCollision$ and $v^k.firstTimeAtCollision$, respectively. Vehicles in the set $\mathcal{C}$ are ordered according to this variable. Specifically, if $v^j.firstTimeAtCollision$ is earlier than $v^k.firstTimeAtCollision$, then v^j gets higher priority than v^k and vice versa. To see more clearly how the '*firstTimeAtCollision*' is determined, we can consider an illustrative example shown in Figure 3.1. In the figure, DTOT(v^i) and DTOT(v^j) have space conflicts in $\{O^i_2, O^i_3\}$ and $\{O^j_5, O^j_6,\}$. If we assume that these occupancies are also conflicting in time, then $v^j.firstTimeAtCollision$ with respect to the vehicle v^i is $\tau(O^j_5)$.

As shown in Algorithm 1, when the ICA receives a REQUEST message from a head vehicle v^i, it first converts the TSS(v^i) into the corresponding DTOT(v^i) using the vehicle's VS. Then the ICA calls the function `checkFV()` to determine if there exist *front vehicles* (See Section 3.3.1 for more details about front vehicles.) that affect the vehicle v^i's motion and also to adjust v^i's DTOT if needed. Then the function `getCV()` is called to determine the set $\mathcal{C}$ which is the set of vehicles whose DTOTs are space-time conflicting with DTOT(v^i). The `updateDTOT()` function adjusts DTOT(v^i) appropriately so that DTOT(v^i) avoids space-time conflict with other vehicle's DTOT. These two functions are iteratively called within the `while` loop until the set $\mathcal{C}$ becomes empty, which indicates that no vehicles in the set $\mathcal{C}$ will collide with the vehicle v^i. After DTOT(v^i) is appropriately adjusted and confirmed that there is no space-time conflict with all other confirmed vehicles, then the confirmed DTOT(v^i) is converted into TSS(v^i). Finally, the ICA sends the confirmed TSS(v^i) back to the vehicle v^i so that the vehicle can cross the intersection safely by following the confirmed DTOT. In the following sections, we provide a more detailed explanation on the subfunctions called within DICA.

3.3.1 Collision Avoidance with Front Vehicles

As shown in Figure 3.3, there are two types of front vehicles when a vehicle v^i is approaching and crossing an intersection. In DICA, a vehicle is considered as a front vehicle of v^i if the vehicle comes from another lane but has the same

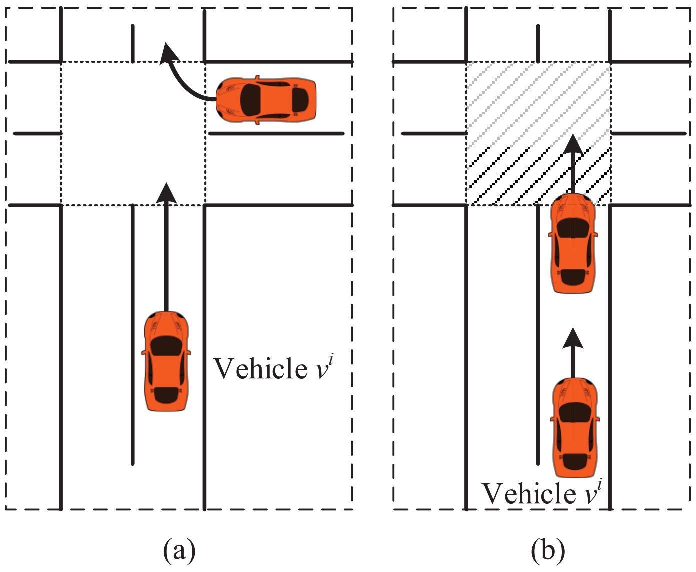


Figure 3.3: Example situations of front vehicles: (a) vehicles with different routes but same exit lane, and (b) vehicles with same intersection crossing routes.

exit lane as vehicle v^i or the vehicle is immediately in front of v^i and has the exact same intersection crossing route as that of v^i. For a vehicle v^i, if there is another confirmed vehicle whose exit lane is the same as that of vehicle v^i and will exit the intersection earlier, then they may collide immediately after crossing the intersection if the speed of vehicle v^i is higher than that of the other confirmed vehicle. To address this problem, AIM [27] adopted a simple strategy which gives one second separation time between these two vehicles. However, it is important to note that the separation time should depend on the speeds of the two vehicles. Hence, instead of using a fixed separation time approach, we use an approach that restricts the maximum speed of the following vehicle by the speed of the front vehicle. In the example situation (a) shown in Figure 3.3, the vehicle v^i's maximum allowed speed within an intersection is restricted by the front vehicle's exit speed. If there is another confirmed vehicle that has the same intersection crossing route as vehicle v^i, we adjust v^i's speed to leave adequate distance between them. In Algorithm 1, the function `checkFV()` looks for the existence of above mentioned front vehicles from all confirmed vehicles and delay the new head vehicle to avoid potential collisions if needed.

3.3.2 Vehicles for Collision Avoidance

Algorithm 2 getCV($\mathcal{S}, DTOT(v^i)$)

```
1: $\mathcal{C} = \emptyset$
2: for $v^j$ in $\mathcal{S}$ do
3:   for $O^j_{k_j}$ in $DTOT(v^j)$ do
4:     if $v^j$ not in $\mathcal{C}$ then
5:       for $O^i_{k_i}$ in $DTOT(v^i)$ do
6:         if $O^j_{k_j} \cap O^i_{k_i} \neq \emptyset$ then
7:           Call getOTI($O^j_{k_j}$) $\rightarrow I(O^j_{k_j}) := [\tau_{lb}(O^j_{k_j}), \tau_{ub}(O^j_{k_j})]$
8:           Call getOTI($O^i_{k_i}$) $\rightarrow I(O^i_{k_i}) := [\tau_{lb}(O^i_{k_i}), \tau_{ub}(O^i_{k_i})]$
9:           if $I(O^j_{k_j}) \cap I(O^i_{k_i}) \neq \emptyset$ then
10:            Assign $\tau_{lb}(O^j_{k_j}) \rightarrow v^j.firstTimeAtCollision$
11:            Push $v^j$ into $\mathcal{C}$
12:          end if
13:        end if
14:      end for
15:    end if
16:  end for
17: end for
18: Sort $\mathcal{C}$ in ascending order of $firstTimeAtCollision$
```

The function `getCV()` returns the set $\mathcal{C}$ that contains vehicles which will cause potential collisions inside the intersection with vehicle v^i. To better understand the operation of function `getCV()`, it is necessary to introduce the way we check the space-time conflict between two occupancies from DTOTs of two vehicles. For every individual occupancy in a DTOT of a vehicle, we define the entrance time (τ_{lb}) and the exit time (τ_{ub}) of the occupancy as the times when the vehicle first contacts and is totally out of the occupancy. These two times can be estimated by taking the times of the previous and next occupancies which are the closest to the occupancy while having no overlapping area. As an example, for the occupancy O_4^j of the vehicle v^j in Figure 3.1, the entrance time $\tau_{lb}(O_4^j)$ and the exit time $\tau_{ub}(O_4^j)$ of that occupancy can be determined by $\tau(O_2^j)$ and $\tau(O_6^j)$, respectively. Note that a DTOT for a vehicle consists of many more numbers of occupancies in practice. Hence, the entrance times and exit times determined in this way can be very close to the actual entrance and exit times of the occupancy. For the first several occupancies in a DTOT, there may not be a previous occupancy that has no overlapping area with themselves. For these occupancies, we simply take the first occupancy's time in the DTOT as these occupancies' entrance time. As an example shown in Figure 3.1, we use $\tau(O_1^j)$ as the entrance time $\tau_{lb}(O_2^j)$ for the occupancy O_2^j. Similarly, we take the last occupancy's time as the exit time τ_{ub} for the last several occupancies in a DTOT.

As shown in Algorithm 2, the function `getCV()` determines the set $\mathcal{C}$ by checking space-time conflict for every pair of occupancies (O_n^i, O_m^j) for all n, m, and j in the set $\mathcal{S}$. Since an occupancy in a DTOT is represented as a rectangle, it is relatively straightforward to do a space conflict checking. For this, Algorithm 2 simply checks if two rectangles have a non-empty intersection or not. If a pair of occupancies (O_n^i, O_m^j) are space-conflicting, then the function continues to investigate these occupancies to determine if they are in time-conflict as well. The above-explained entrance and exit times of an occupancy are used for this purpose. For a given occupancy O, the function `getOTI()` calculates these entrance $\tau_{lb}(O)$ and exit $\tau_{ub}(O)$ times for that occupancy and returns a corresponding time interval $I(O) := [\tau_{lb}(O), \tau_{ub}(O)]$ which we call the *occupancy time interval* in the sequel. Then the two occupancy time intervals for the pair of space-conflicting occupancies are compared to determine if these occupancies are also occupied around the same time. If a pair of occupancies (O_n^i, O_m^j) are conflicting in both space and time, then the vehicle v^j is included in the set $\mathcal{C}$ and the corresponding *firstTimeAtCollision* is determined so that the vehicle v^j is appropriately ordered within the set $\mathcal{C}$.

3.3.3 DTOT Update

The first vehicle v in the set $\mathcal{C}$ is the earliest vehicle that is space-time conflicting with vehicle v^i. Then, in line 6 of Algorithm 1, the function `updateDTOT()` modifies vehicle v^i's DTOT to avoid collision with vehicle v based on space-time conflicting occupancies between vehicles v^i and v. However, it is still uncertain whether $\mathcal{C}$ will be empty or not after this update of avoiding collision with vehicle v. In fact, it is still possible that the modified DTOT of vehicle v^i will be in space-time conflict with DTOTs of other confirmed vehicles. Hence, to ensure that vehicle v^i avoids collision with all other confirmed vehicles, it is necessary to construct $\mathcal{C}$ based on the updated vehicle v^i's DTOT and update the DTOT again to avoid collision with the first vehicle in the set. This process is repeated in the `while` loop in Algorithm 1 until the set $\mathcal{C}$ becomes empty which means that vehicle v^i is not conflicting with any confirmed vehicles. When a vehicle proposes its DTOT to the ICA, we assume that it prefers to select the fastest way to pass the intersection which means the vehicle will try to use the maximum allowed speed to cross. Our current strategy for updating a vehicle's DTOT is to delay the vehicle until other confirmed vehicles cross the intersection safely. Note that, since the times of occupancies in a vehicle's DTOT are always delayed whenever the vehicle's DTOT is updated, it is guaranteed that the vehicle can always meet the updated DTOT by simply decelerating to experience a long time before entering the intersection. The worst case is that a vehicle may need to stop and wait for some time before an intersection to meet the given confirmed TSS from the ICA.

3.4 Liveness Analysis

A *deadlock* is a situation where two or more processes are unable to proceed and each process is waiting for another one to finish because they are competing for shared resources. In an intersection crossing traffic, a deadlock could happen when several vehicles are trying to cross the intersection at the same time. For example, if the coordination between vehicles who want to cross an intersection is not done appropriately, then a deadlock may occur between two vehicles on a same lane. As discussed in [27], it is possible that even when the vehicle in front cannot get confirmed due to the conflict of its intersection crossing route with those of other vehicles which are already confirmed to enter and cross an intersection, the vehicle in the back may get confirmed because its intersection crossing route is not conflicting with other confirmed vehicles' crossing routes. And the vehicle successfully reserves the space for its intersection crossing route within an intersection. In this situation, the front vehicle cannot get confirmed since some part of the intersection crossing route of it conflicts with that of the behind vehicle which is already confirmed and also the behind vehicle cannot proceed to cross the intersection due to the unconfirmed front vehicle. A deadlock situation may also occur when several vehicles from different directions want to cross an intersection at the same time. This type of deadlock situation is discussed in detail in [35] for the case of four vehicles in which none of the vehicles can progress inside the intersection because each of the vehicles' next occupancies is already occupied by other vehicles. Now we show that DICA in Algorithm 1 is free from these deadlock situations.

Proposition 1. *DICA is deadlock free.*

Proof. Let $\mathcal{S}_k$ denote the set of confirmed vehicles at the k-th time step of DICA. Then, we show that the set $\mathcal{S}_k$ is deadlock free for all $k = 0, 1, 2, \cdots$ by induction. First, at time step $k = 0$, it is easy to see that there is no deadlock in $\mathcal{S}_0$ since no vehicle is confirmed yet, i.e., $|\mathcal{S}_0| = 0$ where $|\cdot|$ denotes the cardinality of a set. Then, at time step $k > 0$, let us suppose that $\mathcal{S}_k$ is deadlock free and a new head vehicle v^i is under consideration for confirmation. Note that, as discussed in Section 3.2, a vehicle is considered by DICA for confirmation only if it is the head vehicle on its lane. Hence, it is trivial to see that there won't be a deadlock situation between the vehicle v^i and other vehicle $v^{i'}$ which is behind v^i since $v^{i'} \notin \mathcal{S}_k$. Next, let us note that once a vehicle v^j is in $\mathcal{S}_k$, then the vehicle's DTOT will not be changed while and after a new vehicle v^i is processed to be confirmed by DICA. Hence, it is easy to see that any vehicle which is in $\mathcal{S}_k$ at time step k remains deadlock free at the next time step $(k+1)$. Now suppose that the new vehicle v^i has been confirmed by DICA at time step k and included in the set of confirmed vehicle at time step $(k+1)$, i.e., $v^i \in \mathcal{S}_{k+1} = \mathcal{S}_k \cup \{v^i\}$. Since all vehicles in $\mathcal{S}_k \subset \mathcal{S}_{k+1}$ are deadlock free, if the new vehicle v^i is deadlock free, then we know that $\mathcal{S}_{k+1}$ is deadlock free and this proves the deadlock free property of DICA. In fact, it is straightforward to see that v^i is also deadlock free after its DTOT is updated and confirmed by DICA. First, note that modification of the vehicle v^i's DTOT is not affected by any vehicle $v \notin \mathcal{S}_k$. Instead, it is affected only by vehicles which are already in the $\mathcal{S}_k$. Since all vehicles in $\mathcal{S}_k$ are deadlock free and eventually proceed to cross and exit the intersection, the vehicle v^i's DTOT is also updated so that the vehicle v^i will eventually enter and cross the intersection while all vehicles in $\mathcal{S}_k$ cross the intersection safely. Thus, the vehicle v^i is also deadlock free at time step $(k+1)$ and this concludes the proof of this proposition. □

In an intersection crossing traffic, a *starvation* situation may occur when vehicles from a certain direction are waiting for a very long time or even indefinitely to be allowed to enter and cross an intersection while vehicles from other directions are continuously allowed to cross the intersection. Now we show that a starvation situation will not occur in an intersection crossing traffic that is coordinated by DICA.

Proposition 2. *DICA is starvation free.*

Proof. First, let us recall that, as discussed in Section 3.2, DICA considers a vehicle for confirmation only when the vehicle becomes the head vehicle on its lane. Now let $\sigma(v)$ be the vehicle v's entrance time to the communication region of an intersection, $\mathcal{H}$ be the set of head vehicles which is ordered by $\sigma(v)$ for all $v \in \mathcal{H}$, and $\mathcal{H}^-$ be the set of vehicles which are approaching to cross an intersection but not included in the set $\mathcal{H}$. Clearly, $|\mathcal{H}|$ is bounded by the number of all lanes from which vehicles are approaching an intersection to cross and $|\mathcal{H}^-|$ is also bounded by both the number of lanes and the length of lanes within the communication region of an intersection. Note that DICA processes vehicles in $\mathcal{H}$ for confirmation according to the order of vehicles in $\mathcal{H}$. Once the first vehicle in $\mathcal{H}$ is processed and gets confirmed, then the vehicle is removed from $\mathcal{H}$. Note that if DICA is not starvation free, then there must exist at least one vehicle $v \in \mathcal{H}$ such that the vehicle v will never (or at least take an unnecessarily very long time to) become the first element in the ordered set $\mathcal{H}$. Thus, to prove the starvation free property of DICA, it suffices to show that, for any vehicle $v \in \mathcal{H}$, the vehicle v will be removed from $\mathcal{H}$ in finite time. To show this, we can consider the last vehicle v_{last} in the ordered set $\mathcal{H}$. If $\sigma(v_{last}) \leq \sigma(v)$ for all $v \in \mathcal{H}^-$, then the vehicle v_{last} will be cleared right after all other vehicles in $\mathcal{H}$ are confirmed and this is the earliest time for v_{last} to be removed from $\mathcal{H}$. On the other hand, if $\sigma(v_{last}) > \sigma(v)$ for all $v \in \mathcal{H}^-$ as the worst situation for v_{last}, then the vehicle v_{last} might need to wait until all $(|\mathcal{H}| + |\mathcal{H}^-|)$ vehicles get confirmed to be considered for confirmation. Thus, it is clear that the vehicle v_{last} will be cleared from $\mathcal{H}$ in finite time. $\square$

3.5 Simulation

In this section, we present simulation results of DICA and the Concurrent Algorithm [1] under the same configurations. Compared with Concurrent Algorithm, DICA provides a more efficient way of coordinating vehicles to avoid unnecessary delay.

3.5.1 Simulation Setup

Traffic simulation is performed by the microscopic road traffic simulation package SUMO (Simulation of Urban MObility) [103]. This simulator is widely used in the research community, which makes it easy to compare the performance of different algorithms. Our intelligent intersection management algorithm is implemented by the Traffic Control Interface (TraCI) in SUMO.

The simulated scenario in our simulation is the traffic of a typical isolated four-way intersection with two incoming lanes and one outgoing lane on each road. $v_m = 70\ km/h$ is set as the maximum allowed speed for incoming roads. We generate vehicles with a random velocity within the range of $40\% * v_m$ and v_m when they enter the communication region. To simulate intersection traffic as real as possible, we use different maximum allowed speeds for vehicles with different routes. Although there are no specific speed limits for vehicles who are turning left or right, people are still using some lower speed to feel comfortable and maintain safety. We choose conservative speed limits for turning based on experience from driving in daily life. We use 25 km/h for right turning and 35 km/h for left turning. For vehicles with through route, 65 km/h is set as the speed limit. The time step we used in the simulation is 0.05 s. The maximum acceleration and deceleration rates are 2 m/s^2 and $-4.5\ m/s^2$. Vehicles have a size of 5 meters length and 1.8 meters width. Since, in some cases, a vehicle may need to stop just before the enter line of the intersection region to avoid collisions with other vehicles, the distance from the enter line of the communication region to the enter line of the intersection region should be long enough so that a vehicle can stop from its maximum speed v_m. Thus, from the value used for $v_m = 70\ km/h$ the maximum deceleration rate $a_{min} = -4.5\ m/s^2$, we need at least $-v_m^2/(2a_{min}) \approx 42.03\ m$. So, we use 50 m for the distance from the enter line of the communication region to the enter line of the intersection region. We evaluate the performance of our algorithm in situations where vehicles are spawned randomly from each direction at different probabilities. In our simulation, we consider three different traffic volumes. For each traffic volume, through different traffic generation time and randomly generated vehicles' routes, we test each case for twelve different traffic patterns. Average data of twelve different traffic patterns are used as the result for that case. Each simulation run is terminated when a certain time limit (10 min) has been reached. Fig 3.4 shows a screenshot of simulation in SUMO when vehicles of different routes appear within the intersection simultaneously without an occurrence of collision. Inside the intersection, the straight going vehicle from East goes inside the intersection shortly after the vehicle from North to South clears the conflicting space. Vehicles whose DTOTs is not conflicting with these two can pass the intersection at the same time, for example the right-turning vehicle from South in the figure.

Table 3.1: Simulation results comparison.

	Volume	ρ	Crossed Vehicles			$\bar{\tau}_e$
			$\bar{\tau}$	σ_τ	η	
Concurrent	Volume 1	93.99%	13.98	8.58	51.74%	14.85
	Volume 2	60.24%	89.08	40.91	95.34%	150.77
	Volume 3	30.91%	125.09	48.33	97.17%	408.90
DICA	Volume 1	95.11%	7.21	2.97	10.13 %	7.57
	Volume 2	91.09%	20.47	15.54	55.86%	22.53
	Volume 3	57.09%	50.60	32.87	84.37 %	88.92

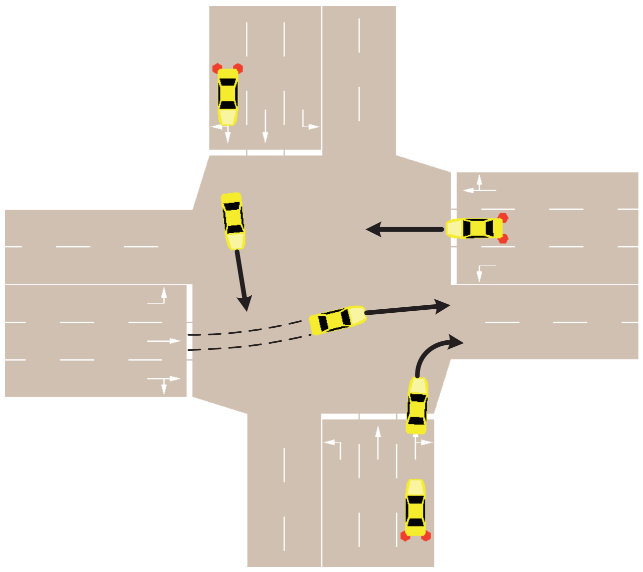


Figure 3.4: A screenshot of simulation which illustrates a situation when vehicles with conflicting routes cross the intersection simultaneously.

All the simulations in this dissertation were run on a 64bit Windows computer, and its processor is Intel(R) Core(TM) i7-4770 CPU @ 3.40 GHz with 8 GB RAM.

3.5.2 Simulation Results

Performance improvement has been validated through extensive simulations of DICA and Concurrent Algorithm. Simulation results of different volumes are shown in Table 3.1. To evaluate and compare the performance, we define several performance measures. *Trip time (τ)* is the difference between actual exit time of the intersection and the time the vehicle enters the intersection communication range. *Average trip time ($\bar{\tau}$)* is the average value of trip times of all crossed vehicles. *Standard deviation (σ_τ)* is computed based on the trip times of all crossed vehicles. *Throughput (ρ)* is defined as the percentage of the number of crossed vehicles against the number of total generated vehicles. *Stopped rate (η)* is obtained by dividing stopped vehicles number by crossed vehicles number.

However, note that neither the average trip time nor the throughput alone is sufficient to correctly evaluate the performance of an algorithm. In some cases, it could be possible that one algorithm shows better performance on average trip time while another algorithm performs better on throughput. Thus, both of the two measures should be considered together to correctly compare and evaluate the performances of different intersection control algorithms. We calculated the ratio of average trip time to throughput, which is called *effective average trip time (*$\bar{\tau}_e$*)* and believe it could show the performance of an algorithm more comprehensively, i.e. $\bar{\tau}_e = \bar{\tau}/\rho$.

Compared with Concurrent Algorithm, DICA increases the throughput and largely decreases the average trip time. As shown in Table 3.1, both algorithms have less and less throughput with the increase of traffic volume. Also, a larger decrease in throughput is shown in Concurrent Algorithm compared with DICA. DICA's smaller values of standard deviation imply that DICA is fairer than Concurrent Algorithm. Stopped rate shows that fewer vehicles experience a stop at the intersection enter line for our DICA algorithm thus saves energy. Effective average trip time could tell us comprehensive information about the performance of an intersection control algorithm. Efficiency and fairness of an algorithm are integrated in this measure. With more vehicles crossed and less average trip time, DICA has much less value of effective average trip time than that of Concurrent Algorithm.

The histogram of trip times of crossed vehicles in one simulation run from case 1 and one particular traffic pattern is shown in Figure 3.5. From the figure we can see that under the same traffic setting, for the crossed vehicles, DICA results in less and concentrated trip times. On the other side, Concurrent Algorithm leads to much longer and wider distributed trip times. Compared with Concurrent Algorithm, DICA is fairer and much efficient for crossed vehicles.

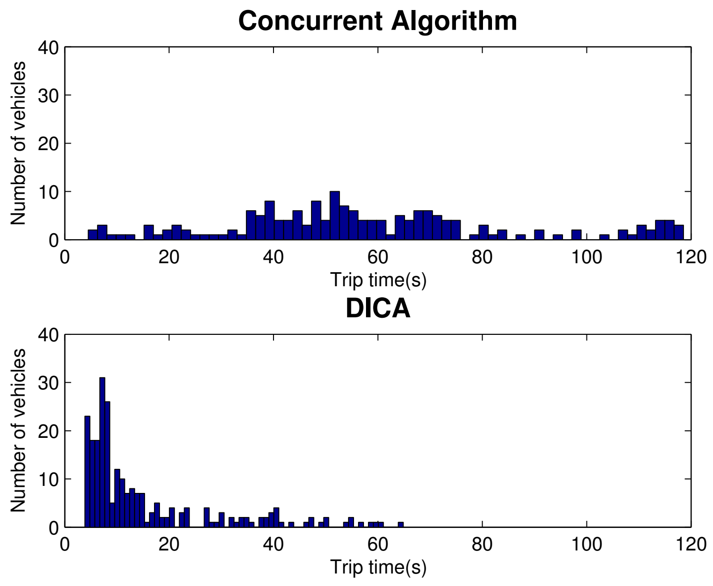


Figure 3.5: The histogram of Trip Times of crossed vehicles in a simulation.

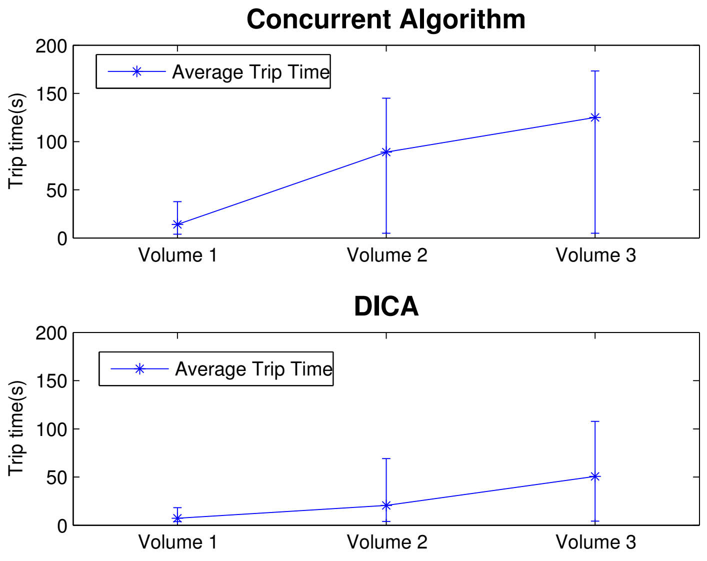


Figure 3.6: Comparison of Trip Times of 3 different cases between DICA and Concurrent Algorithm.

Figure 3.6 shows the maximum, average and minimum trip time for both algorithms under three different volumes. With heavier traffic volume, both algorithms have large maximum trip times and average trip times. However, DICA always performs better than Concurrent Algorithm in all three volumes.

3.6 Summary

In this chapter, we have developed an intelligent intersection control algorithm DICA employing the concept DTOT. V2I interaction protocol has been established for interactions between vehicles and an intersection. DICA is able to manage limited intersection space at a more accurate and efficient way. Simulation results show that our algorithm achieves less effective average trip time compared with Concurrent Algorithm proposed by [1].

CHAPTER FOUR

Computational Complexity Improvements of DICA

In this chapter, we analyze the overall computational complexity of DICA and improve it in several computational technical approaches. We also enhance the algorithm accordingly so that it is possible to operate the algorithm in real-time for autonomous and connected intersection traffic management.

4.1 Computational Complexity Analysis

In this section, we analyze the computational complexity of the DICA shown in Algorithm 1. Recall that $\mathcal{S}$ is the set of vehicles within the communication region of an intersection that have been confirmed to cross. Let us assume that there are n vehicles in $\mathcal{S}$, i.e., $|\mathcal{S}| = n$. Then we have the following result on the computational complexity analysis of DICA.

Proposition 3. *The DICA has $\mathcal{O}(n^2 L_m^3)$ computational complexity where L_m is the maximum length of the intersection crossing routes in an intersection.*

Proof. Let v^i be the vehicle that is currently being processed by the ICA for intersection crossing confirmation. Furthermore, let $N_m := \max_{k \in \mathcal{S}'} N^k$, where $\mathcal{S}' = \mathcal{S} \cup \{v^i\}$ and N^k is the number of occupancies in vehicle k's DTOT. Then, in line 3 (Algorithm 1), it is easy to see that creating a DTOT from the TSS and vehicle size information in vehicle v^i's REQUEST message involves only $\mathcal{O}(N_m)$ computational complexity. In line 4 (Algorithm 1), as explained in Section 3.3, the front vehicle checking function `checkFV()` does a simple comparison with every confirmed vehicle in $\mathcal{S}$ to see if there are any vehicles that might affect vehicle v^i's DTOT and modifies the DTOT if it is necessary to ensure enough separation time and distance between vehicle v^i and other vehicles in front. This process requires $\mathcal{O}(nN_m)$ computational complexity. Then, in line 5 (Algorithm 1), the function `getCV()` is called to identify the set $\mathcal{C}$ of vehicles in $\mathcal{S}$ whose DTOTs might be in space-time conflict with vehicle v^i's DTOT. (Note that, as shown in Algorithm 2, $\mathcal{C}$ is an ordered set according to time of collision and it is clearly $\mathcal{C} \subseteq \mathcal{S}$.) Thus, to return the set $\mathcal{C}$ from the set $\mathcal{S}$, this function performs n times of space-time conflict checking between vehicle v^i and the vehicles in $\mathcal{S}$. If a non-empty set $\mathcal{C}$ is returned in line 5 (Algorithm 1), then, in lines $6 \sim 10$ (Algorithm 1), vehicle v^i's DTOT is iteratively updated until the set $\mathcal{C}$ becomes empty within the `while` loop. (As one can see in Algorithms 1 and 2, these steps are indeed the main part of the DICA and involve some computationally expensive operations. Hence, we describe the computational complexity of steps within the `while` loop separately in the next paragraph.) After the `while` loop, as the last steps in Algorithm 1 in lines from 11 to 13, the space-time conflict free DTOT for vehicle v^i is stored, converted into TSS, and then sent to v^i so that the vehicle can cross the intersection according to the DTOT. Clearly, these steps are fairly simple in terms of computation and in fact require $\mathcal{O}(1)$ complexity. Next, we analyze the computational complexity of the steps within the `while` loop.

Space-time conflict checking steps: As described in Section 3.3, space-time conflict checking in function `getCV()` is done using the DTOTs of confirmed vehicles. Specifically, the two nested `if` blocks from line 6 to line 13 in Algorithm 2 perform this operation. Space conflict checking is performed if there exist non-empty intersections between two occupancies: one from the DTOT of vehicle v^i and another from the DTOT of one of the vehicles in the set $\mathcal{S}$. This is done in the outer `if` block and requires $n \cdot N_m^2$ iterations in the worst case. If two vehicles have a space conflict, then Algorithm 2 proceeds to check for a time conflict. To check time overlapping between two space conflicting occupancies, the function needs to calculate time intervals for these occupancies during which each vehicle occupies its occupancy. This can be achieved easily by comparing occupancy times between occupancies within the same DTOT. As an example, for a given occupancy O_k^i, which is the k-th occupancy within vehicle v^i's DTOT, the lower and upper bounds for the occupancy time can be determined by space overlapping checking between the occupancies O_k^i and $O_{k'}^i$ for $k' = \{1, \cdots, N_m\} \setminus k$. Thus, the two function calls to `getOTI()` within the `if` block involve the computational complexity of $\mathcal{O}(N_m)$. Once the occupancy time intervals are determined, it is a straightforward calculation to check time overlapping as shown in line 9 of Algorithm 2, and it takes $\mathcal{O}(1)$ computational complexity. After identifying all space-time conflicting vehicles from the set $\mathcal{S}$ and storing them in the set $\mathcal{C}$, Algorithm 2 then sorts the set $\mathcal{C}$ according to the ascending order of occupancy times of space-time conflicting occupancies and returns the set. Note that $|\mathcal{C}| \leq n$ and $n \ll N_m$ in general. Hence, this sorting operation can be done with $\mathcal{O}(nlog_2n)$ computational complexity. If we consider all these calculation steps in the `getCV()` function, then one can see that the overall computational complexity for space-time conflict checking steps in `getCV()` is $\mathcal{O}(nN_m^3)$.

DTOT adjustment for collision avoidance: Once the set $\mathcal{C}$ is returned by the function `getCV()`, the DICA updates vehicle v^i's DTOT to avoid space-time conflict with the first vehicle v^j in the set $\mathcal{C}$, as shown in line 7 (Algorithm 1). As described in Section 3.3, our update strategy to avoid space-time conflicts is to make vehicle v^i enter the intersection area a bit later to give vehicle v^j enough time to cross the intersection safely. For this, the DICA first needs to compute the delay time needed to avoid the space-time conflict with vehicle v^j. As the occupancy time interval $I(O_k^j)$ for vehicle v^j's earliest space-time conflicting occupancy has already been determined from the function `getCV()`, it is easy to calculate this delay time in this update process. Once the delay time is determined, then the remaining step is simply changing the times of all the occupancies in vehicle v^i's DTOT that are to be delayed, and this results in $\mathcal{O}(N_m)$ computational complexity.

As described above, the number of vehicles in the set $\mathcal{S}$ is n when the function `getCV()` is called for the first time in line 5 (Algorithm 1). Then, within the `while` loop, the function `updateDTOT()` adjusts vehicle v^i's DTOT to avoid collision with the first vehicle in the set $\mathcal{C}$, and this step reduces the number of vehicles in the set $\mathcal{C}$ that can potentially collide with vehicle v^i at least by one. Thus, in the worst case, the number of vehicles in the set $\mathcal{C}$ returned by the second call of `getCV()` within the `while` loop is $(n-1)$. If we assume the worst case for all the following iterations within the `while` loop until the set $\mathcal{C}$ becomes empty, then it is easy to see that the functions `getCV()` and `updateDTOT()` are called n times within the `while` loop. This implies that, as the computational complexity of the function `updateDTOT()` is significantly lower than that of the function `getCV()`, the overall computational complexity of the `while` loop can be considered as $\mathcal{O}(n^2N_m^3)$.

Note that the maximum number of occupancies N_m depends on both the time that it takes for a vehicle to cross the intersection and the discrete time step used to construct the DTOT by the ICA. If we let h be the discrete time step used by the ICA and T_m be the time it takes for a vehicle to completely cross an intersection when the vehicle starts from rest and accelerates to cross the intersection as quickly as possible, then we have $\bar{N}_m := T_m/h$ as an upper bound for N_m. Note that T_m depends on the length of an intersection crossing route that a vehicle takes to cross an intersection. If we let L_m be the maximum length out of all intersection crossing routes for an intersection, then $\bar{N}_m$ can be expressed in terms of L_m instead of T_m. Specifically, if L_m is long enough so that a vehicle can reach its maximum allowed speed v_m within

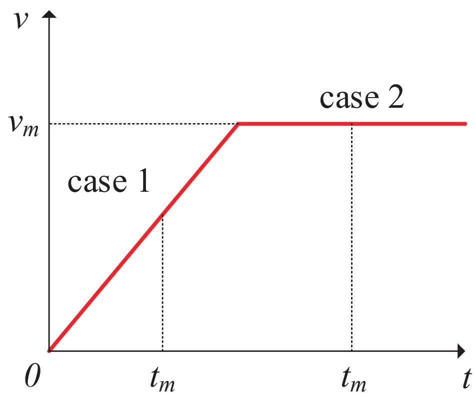


Figure 4.1: Two different cases for the shortest intersection crossing time (T_m) calculation. (Case 1 is the situation when L_m is too short to reach v_m and case 2 is the situation when L_m is long enough to reach v_m while a vehicle is crossing an intersection.)

an intersection before it completely crosses the intersection, then it can be shown that $\bar{N}_m = (2a_mL_m + v_m^2)/(2a_mv_mh)$, where a_m is the maximum acceleration rate of a vehicle. On the other hand, if L_m is not long enough for a vehicle to reach v_m while crossing an intersection, then it is also relatively straightforward to show that $\bar{N}_m = (\sqrt{2L_m/a_m})/h$. (These two different cases are illustrated in Figure 4.1.) If we fix the values for h, v_m, and a_m, then one can see that $\bar{N}_m$ for the former case is proportional to L_m, while for the latter case, $\bar{N}_m$ is proportional to the square root of L_m. Hence, if we substitute L_m for N_m in the computational complexity $\mathcal{O}(n^2N_m^3)$ that we derived above, then we finally have $\mathcal{O}(n^2L_m^3)$ as the overall computational complexity of the DICA.

□

4.2 Algorithm Improvements

According to the computational complexity analysis result described in the previous section, it is true that the original DICA that is shown in Algorithms 1 and 2 is somewhat conservative in terms of computational cost to be used in practice. In this section, we present several approaches that can be used to improve the overall computational efficiency of the algorithm.

4.2.1 Reduced Number of Vehicles for the Space-Time Conflict Check

As shown in Algorithm 2, all confirmed vehicles in the set $\mathcal{S}$ are examined to obtain the set of space-time conflicting vehicles $\mathcal{C}$ for a new unconfirmed head vehicle v^i. However, we see that this computation process can be improved by excluding vehicles that cannot be in space-time conflict with vehicle v^i under any circumstances from the set $\mathcal{S}$. For example, a confirmed vehicle $v^j \in \mathcal{S}$ that has an intersection crossing time interval that is not overlapping with vehicle v^i's intersection crossing time interval can be excluded. Note that the intersection crossing time interval of a confirmed vehicle can be easily determined by the lower bound of the occupancy time $\tau_{lb}(O_{first})$ of the vehicle's first occupancy O_{first} and the upper bound of the occupancy time $\tau_{ub}(O_{last})$ of the vehicle's last occupancy O_{last} in the vehicle's confirmed DTOT. In addition to these vehicles, vehicles in the set $\mathcal{S}$ whose intersection crossing routes are compatible with that of vehicle v^i can also be excluded. Hence, if we let $\mathcal{S}^*$ be the subset of all confirmed vehicles in set $\mathcal{S}$ that can be obtained after excluding all above-mentioned vehicles in determining the set $\mathcal{C}$, then the resulting computational complexity for the space-time conflict checking in function `getCV()` becomes $\mathcal{O}(\alpha_1 n N_m^3)$, where $\alpha_1 := \tilde{n}/n$, $\tilde{n} = |\mathcal{S}^*|$, $n = |\mathcal{S}|$, and N_m is the maximum number of occupancies of all vehicles that are in the set $\mathcal{S}$ and also the vehicle that is currently under consideration for confirmation. (See the proof of Proposition 3 for the precise definition of N_m.)

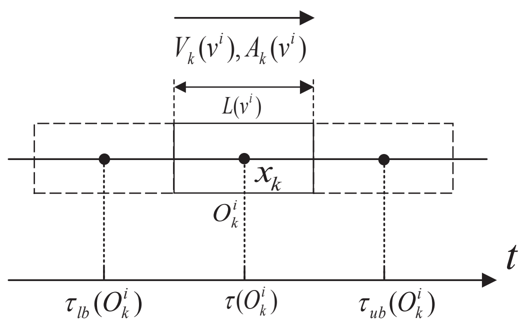


Figure 4.2: Approximate occupancy time interval calculation for a vehicle with the through route

4.2.2 Efficient Space Conflict Check

Note that any two vehicles coming from different directions can collide with each other only within some parts of their intersection crossing routes. Thus, not all occupancies of a vehicle's DTOT needs to be checked for space conflict with another vehicle's DTOT. For example, the two vehicles v^i and v^j in Figure 3.1 have very short ranges of intersection crossing routes that are space conflicting with each other. Thus, the occupancies to be checked can be reduced to $\{O_2^i, O_3^i\}$ and $\{O_5^j, O_6^j\}$ from their entire DTOTs. As the number of occupancies in a DTOT is very large in general, this can improve computational speed considerably. Note that, as the intersection crossing routes are fixed for a specific intersection, we can predetermine these space conflicting short ranges offline only once for all pairs of incompatible intersection crossing routes. Hence, this extra preparation process does not incur an additional computational cost during the online operation of the DICA. If we use DTOT* to denote the subset of the original DTOT for a vehicle that can be obtained from this approach, then the computational complexity of the function `getCV()` in Algorithm 2 can be expressed as $\mathcal{O}(\alpha_2^3 n N_m^3)$, where $\alpha_2 := \tilde{N}_m / N_m$ and $\tilde{N}_m$ is the maximum number of occupancies for all vehicles that are in the set $\mathcal{S}^*$ and the vehicle that is currently under consideration for confirmation.

4.2.3 Approximate Occupancy Time Interval Calculation

As explained in Section 3, the ICA checks if an occupancy of a vehicle is conflicting in time with another vehicle's occupancy using occupancy time intervals that can be obtained from each vehicle's DTOT. However, the method for obtaining an occupancy time interval presented in the proof of Proposition 3 is somewhat naive in the sense of computational complexity. In fact, as analyzed in the proof, such an exhaustive search involves a computational complexity of $\mathcal{O}(N_m)$. To simplify this computation process, we propose estimating the occupancy time interval for a certain occupancy based on the vehicle's speed, length, and acceleration rate instead of performing the exhaustive search. To clarify this idea, let us consider an example. For simplicity, we consider a case when a vehicle is moving in a straight line as shown in Figure 4.2. Let O_k^i be the occupancy for which the DICA needs to determine the occupancy time interval $I(O_k^i) = [\tau_{lb}(O_k^i), \tau_{ub}(O_k^i)]$, $L(v^i)$ be the vehicle length of vehicle v^i, h be the sampling time interval, and x_k be the center position of the O_k^i along the straight line. Then the algorithm first estimates the vehicle's speed and acceleration rate around the occupancy O_k^i from x_k, x_{k-1}, x_{k+1}, and h. Occupancies at x_{k-1}, x_{k+1} are very close to the occupancy O_k^i and are not shown in Figure 4.2 for simplicity. Specifically, if we let $V_{k^-}(v^i)$ and $V_{k^+}(v^i)$ be the speed of vehicle v^i from O_{k-1}^i to O_k^i and from O_k^i to O_{k+1}^i, respectively, then these speeds can be approximated as follows:

$$V_{k^-}(v^i) \approx \frac{x_k - x_{k-1}}{h}, \quad V_{k^+}(v^i) \approx \frac{x_{k+1} - x_k}{h}$$

From these speeds, we now approximate the acceleration rate of the vehicle as follows:

$$A_k(v^i) \approx \frac{V_{k^+}(v^i) - V_{k^-}(v^i)}{h}$$

where $A_k(v^i)$ denotes the acceleration of vehicle v^i at the occupancy O_k^i. If we take the average of the speeds around O_k^i, then we can also approximate $V_k(v^i)$, which is the speed of vehicle v^i at O_k^i. Note that, as the length of vehicle $L(v^i)$ is just a few meters in general, the actual motion of vehicle v^i within the occupancy O_k^i can be approximated fairly accurately by $V_k(v^i)$ and $A_k(v^i)$.

As it is a straightforward process to estimate $\tau_{lb}(O_k^i)$ and $\tau_{ub}(O_k^i)$ from $L(v^i)$, $V_k(v^i)$, and $A_k(v^i)$, we omit the details of these calculations in this paper. For the case when the vehicle is moving on a curved path, we can still use the same method to approximate $V_k(v^i)$ and $A_k(v^i)$. However, in this case, we may need to add a short extra distance to the $L(v^i)$ to estimate $\tau_{lb}(O_k^i)$ and $\tau_{ub}(O_k^i)$ more accurately. Such an extra distance can be simply determined by the curvature of the path that is represented by the DTOT of a vehicle. Finally, if we apply this approximation method for an occupancy time interval calculation in the `getOTI()` function, then the computational complexity of the function `getCV()` improves from $\mathcal{O}(n^2 N_m^3)$ to $\mathcal{O}(n^2 N_m^2)$.

4.2.4 Efficient Occupancies Comparison

In addition to all the techniques described above, the overall computational complexity of Algorithm 1 can be improved further if we employ an efficient searching method, such as the bisection method, in the process of time-conflict checking between two DTOT*s. If we employ this bisection approach for time-conflict checking as shown in Algorithm 3, then the computational complexity of the function `getCV()` can be improved significantly from $\mathcal{O}(n^2 N_m^3)$ to $\mathcal{O}(n^2 N_m^2 \log_2 N_m)$.

All of the improvement techniques discussed in this section are incorporated into the function `getCV()` to improve the overall computational complexity of the space-time conflict checking process. Algorithm 3 shows this modified `getCV()` function, which is now called `enhanced_getCV()`. In Algorithm 3, $\mathcal{S}^*$ represents the set of

Algorithm 3 enhanced_getCV($\mathcal{S}^*, DTOT(v^i)$)

```
1: Let $\mathcal{S}^*$ be the reduced set of $\mathcal{S}$.
2: Let $DTOT^*$ be the reduced $DTOT$.
3: $\mathcal{C} = \emptyset$
4: for $v^j$ in $\mathcal{S}^*$ do
5:    for $O^j_{k_j}$ in $DTOT^*(v^j)$ do
6:       if $v^j$ not in $\mathcal{C}$ then
7:          $high = |DTOT^*(v^i)| - 1$
8:          $low = 0$
9:          while $low \neq high$ do
10:            $middle = (high + low)/2$
11:            Call getEstOTI($O^j_{k_j}$) $\rightarrow I(O^j_{k_j})$
12:            Call getEstOTI($O^i_{middle}$) $\rightarrow I(O^i_{middle})$
13:            if $I(O^j_{k_j}) \cap I(O^i_{middle}) \neq \emptyset$ then
14:               Assign $\tau_{lb}(O^j_{k_j}) \rightarrow v^j.firstTimeAtCollision$
15:               Push $v^j$ into $\mathcal{C}$
16:            else if $\tau(O^j_{k_j}) > \tau(O^i_{middle})$ then
17:               $low = middle$
18:            else if $\tau(O^j_{k_j}) < \tau(O^i_{middle})$ then
19:               $high = middle$
20:            end if
21:         end while
22:      end if
23:   end for
24: end for
25: Sort $\mathcal{C}$ in ascending order of *firstTimeAtCollision*
```

already confirmed vehicles that are obtained from the process in Section 4.2.1 and DTOT* represents the subset of original DTOTs for a vehicle that can be obtained from the approach in Section 4.2.2. The function `getOTI()` within the `while` loop is now replaced by the new function `getEstOTI()` that approximately calculates the occupancy time interval as described in Section 4.2.3. Lastly, the approach for efficient time conflict checking that is presented in Section 4.2.4 is implemented throughout the `while` loop of the DICA.

Proposition 4. *The enhanced DICA has $\mathcal{O}(\alpha n^2 L_m \log_2 L_m)$ computational complexity where $\alpha := \alpha_1^2 \alpha_2 \ll 1$, n is the number of vehicles already confirmed to cross* *an intersection, and L_m is the maximum length of intersection crossing routes in an intersection.*

Proof. First, note that the only part in Algorithm 1 that is affected by this proposed enhancement is the number of confirmed vehicles to be considered for a space-time conflict check, which is reduced from $n = |\mathcal{S}|$ to $\tilde{n} = |\mathcal{S}^*|$, where $\tilde{n} = \alpha_1 n$ and $\alpha_1 \in (0, 1]$. Thus, in Algorithm 1, the functions `enhanced_getCV()` and `updateDTOT()` are now called $\alpha_1 n$ times. Next, we also note that, as nothing is changed by this improvement in the `updateDTOT()` function whose computational complexity is already significantly lower than that of the function `getCV()`, it suffices to analyze the computational complexity of the function `enhanced_getCV()` presented in Algorithm 3 for the overall computational complexity of the enhanced DICA.

Now, as one can see in Algorithm 3, the entire block within the outer `for` loop is executed for $\alpha_1 n$ times as the number of confirmed vehicles to be checked for a space-time conflict with vehicle v^i is reduced from n to $\alpha_1 n$ owing to the approach discussed in Section 4.2.1. Then, within the `for` loop, for each vehicle v^j in the set $\mathcal{S}^*$, occupancies from each vehicle's DTOT are evaluated for space and time conflicts, which *typically* requires an N_m^2 times occupancy comparison operation, where N_m is the maximum number of occupancies in a vehicle's DTOT. However, in the `enhanced_getCV()` function, we first note that the maximum number of occupancies for each vehicle's DTOT to be tested for space-time conflict is reduced from N_m to $\tilde{N}_m$, where $\tilde{N}_m = \alpha_2 N_m$ and $\alpha_2 \in (0, 1]$ owing to the approach presented in Section 4.2.2. Another important improvement is that the computational complexity for the occupancy time interval calculation is improved from $\mathcal{O}(N_m)$ to $\mathcal{O}(1)$ within another enhanced function `getEstOTI()` as discussed in Section 4.2.3. Therefore, the overall computational complexity of the outer `for` loop can be estimated as $\mathcal{O}(\alpha_1 \alpha_2^2 n N_m^2)$. However, note that this is the case when we use the same occupancy comparison method as used in the original `getCV()` function. As shown in Algorithm 3, the process of occupancy comparison is now performed based on the bisection search method. Roughly speaking, for a given n and N_m, this efficient search method improves the overall computational complexity of the function from $\mathcal{O}(nN_m^2)$ to $\mathcal{O}(nN_m \log_2 N_m)$, as discussed in Section 4.2.4. If we combine this and others discussed above for the overall computational complexity of the `enhanced_getCV()` function, then we have $\mathcal{O}(\alpha_1\alpha_2 nN_m \log_2 N_m)$. Recall that, as the `enhanced_getCV()` function is called $\alpha_1 n$ times in the main `while` loop as discussed above, we have $\mathcal{O}(\alpha_1^2\alpha_2 n^2 N_m \log_2 N_m)$ as the overall computational complexity of the DICA.

As we have analyzed already in the proof of Proposition 3, N_m is linearly proportional to the maximum length of intersection crossing routes L_m. Hence, if we substitute L_m for N_m, then we finally have $\mathcal{O}(\alpha n^2 L_m \log_2 L_m)$ as the overall computational complexity of the enhanced DICA,where $\alpha := \alpha_1^2\alpha_2 \ll 1$. □

4.3 Simulation

In this section, we present simulation results that demonstrate the improved performance of the enhanced DICA over the original algorithm. The performance of the enhanced algorithm is also compared with that of an optimized traffic light control algorithm.

4.3.1 Simulation Setup

To evaluate the performance of the original DICA and the enhanced DICA, we implemented both algorithms in SUMO [103], and performed extensive intersection traffic simulations. In our simulation, the simulated situation was an intersection crossing traffic on a typical isolated four-way intersection with three incoming lanes, one of which is a dedicated lane for left-turning vehicles, and two outgoing lanes on each road. We set 70 km/h as the maximum allowed speed v_m for all incoming vehicles. To make the simulation more realistic, we let vehicles approach the intersection with different speeds when they entered the communication region of the intersection. Specifically, when a new vehicle was spawned outside of the communication region, its initial speed was randomly assigned within the range from 40% to 100% of the maximum allowed speed v_m. Thus, a vehicle kept this random initial speed until it entered the communication region and then it either followed another vehicle or was confirmed by the ICA with a feasible DTOT. The maximum acceleration (a_{max}) and deceleration (a_{min}) rates for vehicles that are used in simulations are 2 m/s^2 and 4.5 m/s^2, respectively. The size of a vehicle used in simulations is 5 meters long and 1.8 meters wide. The distance from the enter line of the communication region to that of the intersection region is set as 50 m. The time step that is used in simulations is 0.05 seconds. In most cases, a simulation terminates when the simulation time reaches 10 minutes.

In our simulations, vehicles were spawned according to several random variables to generate various traffic volumes as well as traffic patterns. Specifically, p_V is the probability that a vehicle is spawned. p_L, p_S, p_R are the probabilities that the new vehicle has a left-turning, through or right-turning route. Thus, by adjusting p_V, we could generate various traffic volumes. As shown in Table 4.1, we set $p_L = 0.2$, $p_S = 0.6$, and $p_R = 0.2$ for all traffic volume cases so that 20% of all incoming vehicles had left-turning routes, 60% had straight routes, and the other 20% had right-turning routes. We use three random seeds to generate three different intersection traffic patterns for each traffic volume. Thus, to obtain simulation data for each traffic volume, we run three simulations of different traffic patterns for each simulation and then use the averages of these simulation results as the result for each traffic volume case. The intersection crossing traffic generated in most of our simulations was balanced traffic in the sense that the numbers of vehicles generated for each incoming

Table 4.1: Parameters used for various traffic volumes and patterns. (∗ Expected number of vehicles per 10 minutes.)

Parameter	Value
Traffic volumes∗	100 / 200 / 300 / 400 / 500
p_V	0.03 / 0.06 / 0.08 / 0.11 / 0.14
p_L	0.20
p_S	0.60
p_R	0.20
Random seeds	12 / 21 / 66

road were about the same. However, for a simulation to show the starvation free property of the proposed DICA, the intersection traffic was purposely designed to be unbalanced, whereby the number of vehicles for minor approaching roads was roughly 30% that of the vehicles on major roads.

In the following discussion, *simulation time* means the simulated time used in a simulation program and *computation time*, which will be discussed later in Section 4.3.2, means the actual elapsed time that it takes for a computer to run a simulation. Furthermore, in Section 4.3.2, the traffic control performance of the enhanced DICA is compared with that of a traffic light algorithm with fixed cycles. To have a comparable traffic light program, we computed the optimal signal cycles for different traffic volume cases by using the exponential cycle length model $C_0 = 1.5Le^{1.8Y}$ from [104]. In the model, L represents the total lost time within the cycle. The lost time for each phase is assumed to be 4 s [105]. Thus, $L = 4 \times 4\ s = 16\ s$. Y is the sum of critical phase flow ratios. The duration of the yellow light for each phase is 3 s.

4.3.2 Simulation Results

The computation times and performances of three different traffic patterns for all five volume cases were obtained from the simulations.

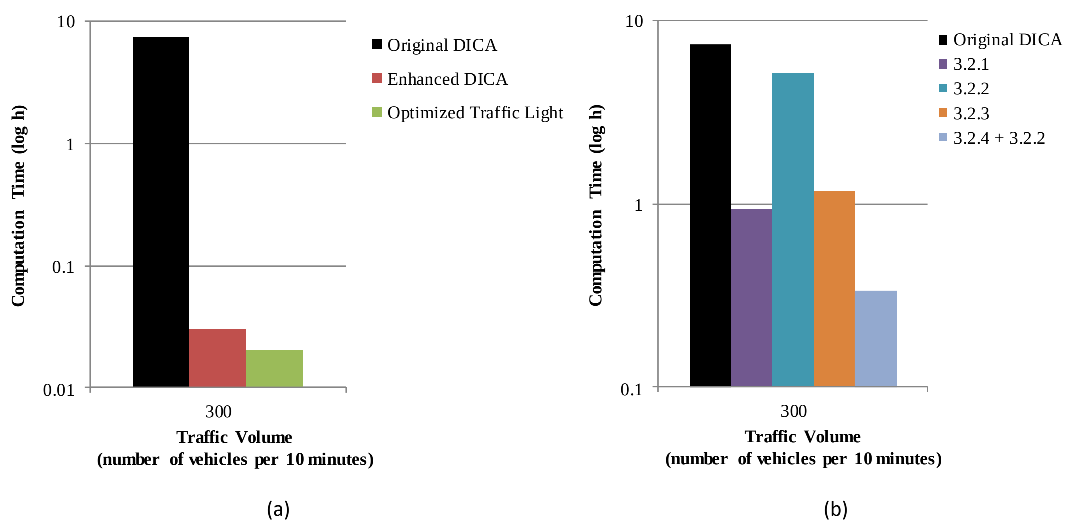


Figure 4.3: Comparison of computation times for traffic volume with 300 vehicles per 10 min. (The symbol 4.x represents the improvement technique in Section 4.2,where x = { 1, 2, 3, 4 }.) (a) original DICA with different algorithms and (b) original DICA with different improvement techniques

Computation Time

Figure 4.3 (a) compares the computation times of the original DICA, the enhanced DICA, and the optimized traffic light algorithm. Figure 4.3 (b) shows how much computational improvement was achieved through each computational improvement technique discussed in Sections 4.2.1, 4.2.2, 4.2.3, and 4.2.4. Note that, as the computational improvement technique in Section 4.2.4 is implemented based on the computational improvement technique in Section 4.2.2, we had to combine techniques from both Sections 4.2.4 and 4.2.2 to indirectly show the improvement due to the technique in Section 4.2.4. Here, we show a comparison of computation times for only one traffic volume case with 300 vehicles per 10 min, as the trends for other volume cases are similar. The vertical axis in Figure 4.3 is the computation time in units of one hour, which is represented on a logarithmic scale. As shown in Figure 4.3 (a), the enhanced DICA that implements all improvements discussed in Section 4.2 takes significantly less computation time, i.e., only 0.4% of the computation time of the original algorithm. When we apply each computational improvement technique individually, our result shows that the enhanced DICA takes about 11% of the computation time of the original DICA with the technique in Section 4.2.1, 59% with the technique in Section 4.2.2, 13% with the technique in Section 4.2.3, and 6% with techniques in Sections 4.2.2 and Section 4.2.4 together. If we combine all of these individual improvements to estimate the collective improvement, then the computation time is about 0.45% of that of the original DICA, which is similar to the computation time result with the enhanced DICA in which all these techniques are implemented.

Table 4.2 compares the computation times between the enhanced DICA and the optimized traffic light algorithm for all five traffic volume cases. From the results shown in the table, we note that the computation time for the optimized traffic light algorithm gradually increases as the traffic volume increases. However, since the optimized traffic light algorithm has $\mathcal{O}(1)$ computational complexity, its computation time cannot be affected by the number of vehicles around an intersection. Thus, roughly speaking, one can say that the computation time of the optimized traffic light for a particular traffic volume case is, in fact, the time required for the simulation software SUMO to run a simulation with the number of vehicles for that particular traffic volume case. Therefore, the actual computation time of the enhanced DICA for a particular traffic volume case can be roughly approximated by subtracting the computation time of the optimized traffic light for the case from the computation time of the enhanced DICA presented in the table. For example, for the traffic volume with 500 vehicles, the actual computation time for the enhanced DICA can be approximated as $0.031(= 0.058 - 0.027)$ hours which is 1.86 minutes. Note that this 1.86 minutes is the computation time taken by the algorithm to handle 500 vehicles. Thus this in turn implies that it takes only 0.2232 seconds to handle each vehicle. An exception to this approximation is the case with 100

Table 4.2: Computation time comparison between enhanced DICA and optimized traffic light

Traffic volume (Number of vehicles per 10 minutes)	100	200	300	400	500
Optimized Traffic light (h)	0.014	0.017	0.020	0.024	0.027
Enhanced DICA (h)	0.011	0.024	0.026	0.042	0.058

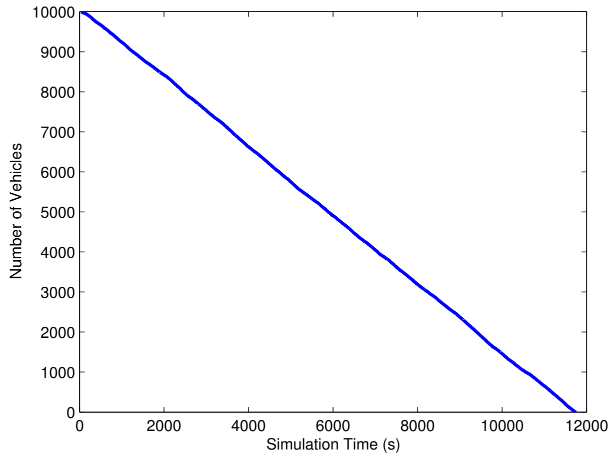


Figure 4.4: The number of vehicles which wait to cross the intersection over time.

vehicles traffic volume case where the computation time for optimized traffic light takes a longer time than that of the enhanced DICA. The reason for this result can be understood by considering the fact that, in such a low traffic volume situation, the average number of vehicles to be simulated by SUMO at each simulation time step is smaller in the enhanced DICA case since vehicles are crossing an intersection much faster without waiting at an intersection under the enhanced DICA than the optimized traffic light as shown in Section 4.3.2.

Liveness and Safety

Although we have theoretically shown the liveness of DICA, it is better to have simulation results that support the theory. Since the simulation in this section is only a verification, we run a simulation with 10, 000 vehicles instead of giving a

Table 4.3: Average trip time comparison between major roads and minor roads in unbalanced traffic

Traffic volume (Number of vehicles per 10 minutes)	100	200	300	400	500
Average trip time on major roads (s)	6.17	6.60	7.38	8.15	10.15
Average trip time on minor roads (s)	6.21	6.57	7.38	7.90	9.63

restriction on the simulation time. The simulation ends after all 10, 000 vehicles have exited the simulation scene. We recorded the number of vehicles that are waiting to cross the intersection at each simulation time step and plot the number profile in Figure 4.4. As shown in the figure, the number of vehicles drops to zero in almost a linear way within a finite time which demonstrates that every vehicle was able to cross the intersection eventually that proves the Proposition 1 in Section 4.1.

We performed a set of simulations for the case of an unbalanced traffic situation where the number of vehicles on the minor roads is only 30% of that on major roads to demonstrate the fairness of the DICA. To show the fairness of the algorithm, we recorded the average trip times for major roads and minor roads for every traffic volume. As shown in Table 4.3, the average trip time of the minor roads is about the same as that of the major roads. This shows that cases in which some vehicles cannot get confirmed or will experience a long delay before being confirmed do not exist, which demonstrates that the DICA is starvation free.

To validate the safety property (i.e., collision freeness) of the DICA through simulation, we computed the inter-vehicle distance between every pair of vehicles within an intersection at every second during the simulation. As each vehicle is represented as a polygon, a 5 m long and 1.8 m wide rectangle more precisely, we obtained this data based on an algorithm of the shortest distance calculation between two polygons. A histogram of the recorded inter-vehicle distances is shown in Figure 4.5. Clearly, the inter-vehicle distance must be less than or equal to zero if two vehicles are in a collision and must be positive otherwise. As one can see

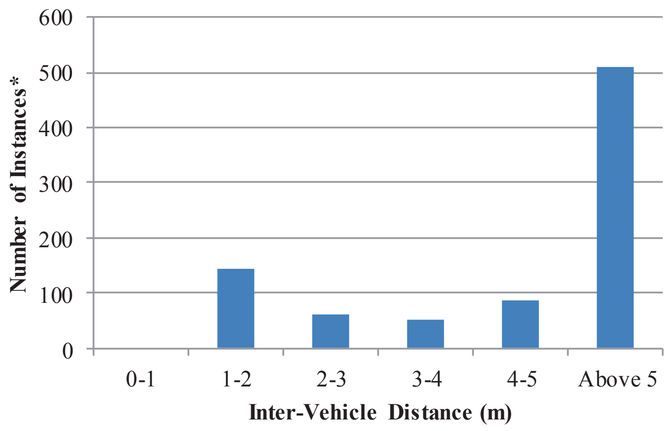


Figure 4.5: Histogram of the inter-vehicle distance within the intersection. (* An instance means a situation when a pair of vehicles are separated by the calculated inter-vehicle distance.)

from the figure, there is no instance observed throughout the entire simulation with less than a 1 m inter-vehicle distance, which is a clear indication that there is no collision inside the intersection. Note that Figure 4.5 demonstrates the safety of the DICA. The safety problem pertaining to the robustness of the DICA that vehicles cannot follow a confirmed DTOT correctly will be studied in our future work.

Control Performance

The overall traffic control performance of the enhanced DICA is also evaluated and compared with that of the optimized traffic light algorithm based on the performance measures introduced in Section 3.5.2. A comparison of the performance between the enhanced DICA and the optimized traffic light control algorithm is shown in Figure 4.6. From this result, we can see that, as the throughputs of the two algorithms are always similar, the profiles of the average trip time and effective average trip time also show similar trends. The enhanced DICA always performs better than the optimized traffic light for the first four traffic volume cases. In the case of the traffic volume with 500 vehicles, the average trip time performance of

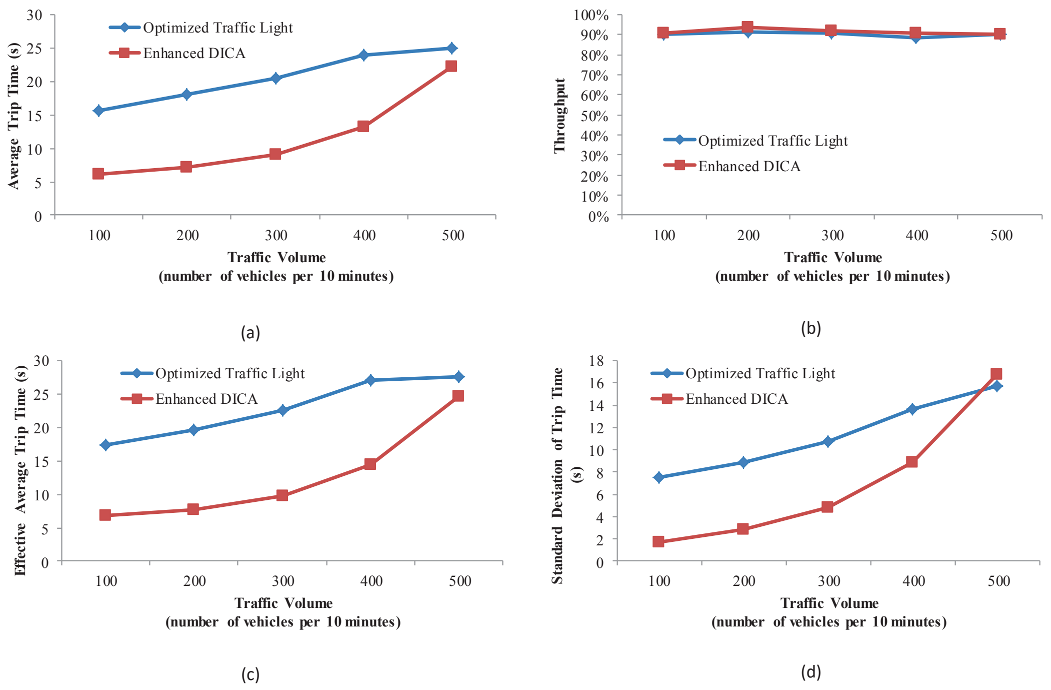


Figure 4.6: Performance comparison between the enhanced DICA and optimized traffic light. (a) average trip time, (b) throughput, (c) effective average trip time, and (d) standard deviation of trip time

the enhanced DICA becomes closer to that of the optimized traffic light. Furthermore, the enhanced DICA has a slightly larger standard deviation of the trip time than the optimized traffic light. In short, the enhanced DICA performs much better than the optimized traffic light from low to medium traffic volume cases, while its performance becomes worse and closer to the performance of the optimized traffic light for heavy traffic volumes.

We note that this result is mainly due to the fundamental difference between individual vehicle-based traffic coordination algorithms and traffic flow-based coordination algorithms. To see this, we can consider a heavy traffic situation when all incoming roads are congested. In such a situation, we know that most vehicles start to cross an intersection at rest when they are allowed to cross the intersection either by a green light under a traffic light algorithm or confirmation under the proposed

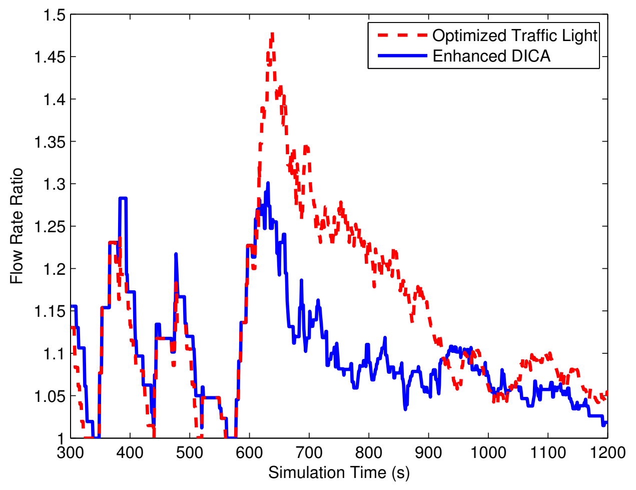


Figure 4.7: Flow rate ratio when traffic volume changes from 100 to 500

DICA. Under traffic light control, if a vehicle is crossing an intersection, then it is highly likely that a few more following vehicles can also cross the intersection without being stopped. However, in the case when vehicles are controlled by an individual vehicle-based coordination algorithm such as our enhanced DICA, it is possible to have a situation where vehicles from different roads are permitted alternatively to cross the intersection, which inevitably results in more frequent stops than the case of traffic light control. This is why the enhanced DICA performs worse and closer to the optimized traffic light in the heavy traffic volume situation. In fact, this result reveals the important point that to achieve the best throughput performance, it is necessary to combine both strategies: an individual vehicle-based coordination in normal traffic volume and a traffic flow based-coordination in congested situations.

Another simulation was performed to validate the transient traffic control performance of the DICA when the traffic volume is changing. We ran a simulation with a simulation time of 20 min during which the traffic volume increased from the case of 100 vehicles to 500 vehicles per 10 min. At each simulation time step, the ratio of the vehicle number generated to the number of vehicles that have exited the intersection, which we call the *flow rate ratio*, was calculated to determine how much congestion could occur and also how long it takes to address the congestion. The flow rate ratio measured during the simulation time is plotted in Figure 4.7. In this figure, if the flow rate ratio is close to 1, then it means that all vehicles approaching an intersection have already crossed the intersection and there are no vehicles waiting to cross at that time. The simulation time starts from 300 s in the figure, as the flow rate ratio needs some time to become stable. From the figure, we can also see that before the increase in the traffic volume, the flow rate ratios of the two algorithms are very similar. After 600 s, at which the traffic volume is changed to 500 vehicles, the flow rate ratio of the optimized traffic light increased considerably. Figure 4.7 shows that the DICA is more resilient to changes in traffic volume than the optimized traffic light.

4.4 Conclusion

In this chapter, We analyzed the computational complexity of the original DICA and enhanced the algorithm so that it can have better overall computational efficiency. Simulation results show that the computational efficiency of the algorithm is improved significantly after the enhancement and the properties of starvation free and safety are guaranteed. We also validated that the overall throughput performance of our enhanced DICA is better than that of an optimized traffic light control mechanism in the case when the traffic is not congested.

CHAPTER FIVE

Reactive DICA: an Approach for Expedited Crossing of Emergency Vehicles

The problem of evacuating emergency vehicles as quickly as possible through autonomous and connected intersection traffic is addressed in this chapter. DICA is augmented to allow emergency vehicles to cross intersections faster and keep the influence on other vehicles' travel as minimum as possible.

5.1 Reactive DICA

The problem we want to solve is how to let EVs which are driven autonomously cross an intersection as soon as possible under the connected and autonomous traffic environment. In the meantime, we aim to keep all other vehicles having similar travel times as when there are no EVs in the traffic. In short, our objective is to evacuate EVs through an intersection as quickly as possible while other vehicles' travel times are minimally affected. Note that for simplicity the term emergency vehicle in this dissertation means an emergency vehicle in the emergency status (i.e. with siren and the lights on). Same assumptions with our previous work [45, 47] are employed in this problem. Overtaking and lane-changing inside the communication region are not allowed which means that vehicles on each lane will keep its lane once it enters the communication region. As an approach to give preference to EVs in autonomous traffic, we give priority to EVs in an intersection crossing traffic by optimizing the sequence of crossing vehicles. Also, since we are augmenting the original DTOT-based intersection control algorithm, the new algorithm will only be used to coordinate vehicles when there is an EV within the communication region of an intersection while the crossing traffic is controlled the same way as before when all vehicles are normal vehicles inside the communication region. Thus, the entering of an EV activates the new algorithm, so we call the augmented DICA the *Reactive DICA* (R-DICA). DICA is only taking care of head vehicles which reduces computational complexity and communication load of the ICA a lot. However, unlike in DICA, more vehicles are needed to be considered in R-DICA in order to allow EVs to cross an intersection as fast as possible. Specifically, all vehicles on the lane of an EV which are ahead of the EV should be included in the set of vehicles whose intersection crossing order are to be optimized. In the sequel, we call all those vehicles as *vehicles on EV's lane*. Thus, the set of vehicles that we need to consider for vehicle ordering are all unconfirmed vehicles on EV's lane and also all confirmed vehicles which are not on EV's lane. All these vehicles can be divided into two types: vehicles whose DTOTs cannot be modified (vehicles who have already entered the intersection or cannot make a stop at the enter line even with maximum deceleration), and vehicles whose DTOTs could be changed (vehicles who are stopping at the enter line of the intersection or are able to make a stop at the enter line, or unconfirmed vehicles who are ahead of the EV). The sequence of vehicles of the latter type is what we can optimize to expedite the crossing of EVs. We define the set of these vehicles as $\mathcal{S}^*$.

Roughly speaking, our approach for a fast crossing of emergency vehicles is to assign the highest priority to them and delay confirmation for all other normal vehicles. Thus, incorporating a priority based ordering of vehicles into the basic DICA framework would achieve this goal. To find such an optimal vehicle ordering, we formulate an optimization problem based on the *entrance time* of vehicles which is the time a vehicle enters the line of an intersection. Let $\mathcal{P}(\mathcal{S}^*)$ be the set of ordered vehicle sequences (or simply called a *sequence* in the sequel) from the set of vehicles in $\mathcal{S}^*$. Then, if we use T_e^v to represent the entrance time of vehicle v, a reasonable objective function for our optimization problem would be:

$$\min_{\mathcal{P}(\mathcal{S}^*)} T_e^{EV} \tag{5.1}$$

where T_e^{EV} is the entrance time of an EV at an intersection. Thus, to solve this optimization problem, we first need to introduce an approach that determines the entrance time of an EV.

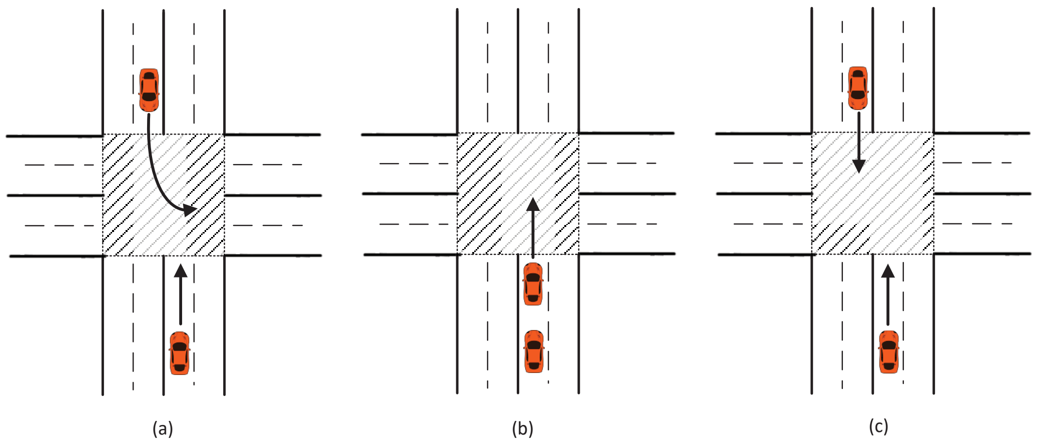


Figure 5.1: Three different situations for separation time.

First, we note that some sequences in $\mathcal{P}(\mathcal{S}^*)$ can be eliminated if we impose some constraints for optimal vehicle ordering. For example, the order of vehicles on EV's lane cannot be altered and hence should be preserved. Also, since all confirmed vehicles $\mathcal{S}^*$ are able to stop before the enter line of an intersection, we can allocate higher priorities for vehicles on EV's lane than those in other lanes. We use $\bar{\mathcal{P}}(\mathcal{S}^*)$ to denote the set of ordered sequences of vehicles satisfying these constraints. Now, let us consider a sequence s in the set $\bar{\mathcal{P}}(\mathcal{S}^*)$. Then, if we consider the first vehicle v

in the sequence s, it is easy to see that the vehicle is always a head vehicle on EV's lane and has a confirmed DTOT. Hence the entrance time of this vehicle v can be determined simply by its $\tau(\mathcal{O}_1^v)$ which is the time when the vehicle v occupies the first occupancy of its DTOT. For any other vehicles which are not the first vehicle in the sequence s, the way to compute its entrance times is a bit different. We need a time interval between any two successive vehicles in a sequence to ensure safety. This time interval is called *separation time* τ_s. In this chapter, as shown in Figure 5.1, we define three separation times for different situations between two vehicles.

$$\tau_s = \begin{cases} \delta_c & v_i \otimes v_j \text{ Figure 5.1 (a), or} \\ \delta_s & v_i \prec v_j \text{ or } v_j \prec v_i \text{ Figure 5.1 (b), or} \\ 0 & v_i \odot v_j \text{ Figure 5.1 (c)} \end{cases} \tag{5.2}$$

where symbols $\odot$ and $\otimes$ are used to represent two vehicles' routes are compatible and conflicting respectively. $v_i \prec v_j$ represents that vehicles v_i and v_j are on a same lane and v_i is following v_j. The separation time's value depends on pavement conditions, vehicle mechanical errors and weather conditions. The focus of this chapter is proposing a coordination algorithm not the determination of these values. Thus, we just approximate the values from current empirical estimations which are widely accepted [106]. Then the expression to compute the entrance time of v^j which is not the first vehicle v^1 in the sequence is:

$$T_e^j = max\{T_a^j, T_e^i + \tau_s\} \tag{5.3}$$

where v^i is the immediate predecessor of v^j in the sequence, T_a^j is the *predicted arrival time* of the vehicle v^j which is the shortest time for the vehicle to arrive at the enter line of an intersection under the constraints of maximum acceleration and speed without considering other vehicles in a traffic. T_e^i is v^i's entrance time and τ_s is the separation time between v^i and v^j. Starting from the second vehicle in sequence, this equation is iteratively used to compute the entrance time of each vehicle in the sequence until the entrance time of the emergency vehicle is computed.

Now the complete form of an optimization problem for optimal vehicle ordering to minimize the entrance time of the EV is formulated as follows:
Given predicted arrival times $T_a^{v_i}$ for all $v_i \in \mathcal{S}^*$, find s^* such that

$$\begin{aligned} s^* \quad &= \quad \min_{s \in \mathcal{P}(\mathcal{S}^*)} T_e^{EV} \\ s.t. \quad & |T_e^i - T_e^j| \geq \begin{cases} 0 & v_i \odot v_j \\ \delta_c & v_i \otimes v_j \\ \delta_s & v_i \prec v_j \text{ or } v_j \prec v_i \end{cases} \\ & T_e^v \geq T_a^v \quad \forall v \in \mathcal{S}^* \end{aligned} \tag{5.4}$$

A naive approach to solve the optimization problem in (5.4) is an exhaustive search in all possible sequences that can be generated from the set $\mathcal{S}^*$. If we suppose that there are n vehicles in $\mathcal{S}^*$ (i.e. $|\mathcal{S}^*| = n$) and there are n_{EV} numbers of vehicles on EV's lane, then there are $n!/n_{EV}!$ sequences in $\mathcal{P}(\mathcal{S}^*)$. However, if n is becoming large, then the computational time and resources required to solve the optimization problem are increasing significantly. Hence it might not be an efficient approach to use an exhaustive search method when we want to solve the problem (5.4) with many vehicles. Such computation issues of the problem present the need to seek heuristic approaches which are good at solving complex problems in a very short time compared with the exhaustive search. Several heuristic optimization approaches like genetic algorithm, ant colony system, artificial neural networks exist in the literature. [107] used permutation encoding scheme and solved the flowshop scheduling problem with an objective of minimizing the makespan. [24] proposed a genetic algorithm to optimize the groups of compatible vehicles in a very short time. [108] and [109] reviewed many researches that genetic algorithms can be used to solve job scheduling problems which can meet our requirements. Thus, we also choose to use Genetic Algorithm (GA) to obtain an optimal sequence of vehicles.

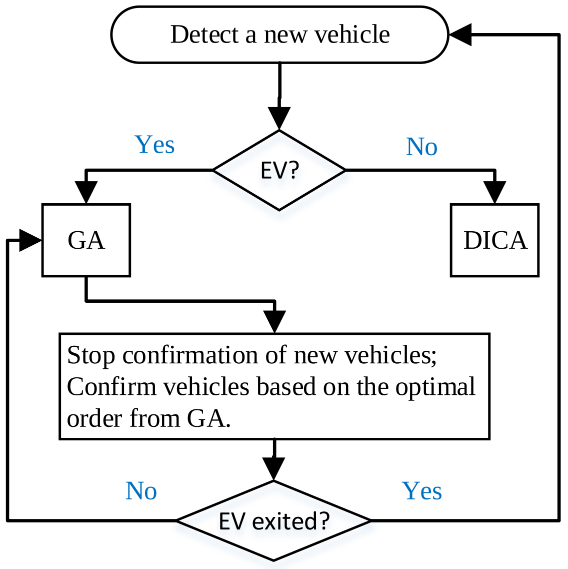

Detect a new vehicle
Yes
EV?
No
GA
DICA
Stop confirmation of new vehicles; Confirm vehicles based on the optimal order from GA.
No
EV exited?
Yes

Figure 5.2: Control flow diagram of the ICA in R-DICA.

The high-level architecture of R-DICA combining GA and DICA is shown in Figure 5.2. R-DICA activates GA when the ICA detects an EV. Then the ICA stop accepting any confirmation of new vehicles which are detected after the EV. All vehicles who belong to $\mathcal{S}^*$ are rearranged to obtain the optimal sequence for the EV's crossing by GA. Then the ICA only confirms vehicles who are already included in the set $\mathcal{S}^*$ until the EV exits the intersection. Once the EV is completely out of the intersection, the ICA switches back to use DICA to manage normal intersection crossing traffic.

5.2 Genetic Algorithm for Vehicle Ordering

In this section, we discuss the details of how GA is used to find the optimal vehicle sequence in (5.4). Genetic Algorithms, which have been widely used to solve problems in computer science, artificial intelligence, information technology and engineering, are techniques of self-organized and self-adapting artificial intelligence mimicking the evolutionary process of creatures in nature [108, 110]. A solution in GA is called an *individual* which is encoded compactly to facilitate the processes of crossover and mutation that are essential in a genetic algorithm. A group of individuals is called a *population* in which some individuals are selected as parents to generate offspring through crossover and mutation. Based on some features of each individual, some individuals survive and others die among all the original population and new individuals. Individuals who correspond or near correct solution have a better chance to survive during evolving since they have high objective values which are called *fitness.* Fitness function should be defined properly to evaluate each individual. As introduced above, solutions in GA evolve to adapt to the objective problem. Optimal or near-optimal solutions are expected to be obtained after a certain number of generations. We propose a GA to solve the complex traffic control problem for emergency vehicles in a short time. Permutation encoding scheme is used in the algorithm. And crossover and mutation operators suitable for permutation encoding scheme are devised. The proposed GA for vehicle ordering is shown in Algorithm 4. A detailed discussion for permutation scheme, crossover, mutation, etc. of the proposed GA is given in the following sections.

In the proposed GA, we first generate a random population $\mathcal{I}$ that contains N_{pop} individuals which are encoded by permutation scheme. The function $feasibilityCheck()$ takes a set of individuals and makes modifications to the infeasible individuals. Feasible individual corresponds to a sequence of vehicles that does not violate the order

Algorithm 4 Genetic Algorithm for Vehicle Ordering

```
1: Generate $N_{pop}$ different individuals randomly for $n$ vehicles in $\mathcal{S}^* \rightarrow \mathcal{I}$
2: $feasibilityCheck(\mathcal{I}) \rightarrow \mathcal{I}$
3: $k = 0$
4: $j = 0$
5: $fitness\_best\_last = 0$
6:
7: while $k < N_{max}$ and $j < N_{noChange}$ do
8:     $crossOver(P_c, \mathcal{I}) \rightarrow \mathcal{I}$
9:     $feasibilityCheck(\mathcal{I}) \rightarrow \mathcal{I}$
10:    $mutation(P_m, \mathcal{I}) \rightarrow \mathcal{I}$
11:    $feasibilityCheck(\mathcal{I}) \rightarrow \mathcal{I}$
12:
13:    $fitness\_best, individual\_best = fitness(\mathcal{I})$
14:    if $fitness\_best > fitness\_best\_last$ then
15:        $j = 0$
16:    else
17:        $j = j + 1$
18:    end if
19:    top $N_{pop}$ individuals $\rightarrow \mathcal{I}$
20:    $k = k + 1$
21: end while
22: Decode $individual\_best$
```

of vehicles on EV's lane. After proper modification, the function returns a set containing individuals which are all feasible. The function $crossOver()$ then perform crossover on randomly selected pairs of individuals from the population $\mathcal{I}$ with a probability P_c to generate new offspring. Then the feasibility of the offspring is checked. Notice that after crossover, the number of individuals is larger than N_{pop}. Mutation on the produced offspring with probability of P_m is done by function $mutation()$. The mutated individuals also need to be checked for feasibility and modified if needed. Based on given conditions, each individual in $\mathcal{I}$ is evaluated by a fitness function $fitness()$ which computes the reciprocal of the entrance time of the emergency vehicle in that individual. The highest fitness value and the corresponding individual are recorded. Notice that the fitness can also be obtained by using other metrics like the exit time of the EV, or the trip time of the EV, etc.

These metrics will give us similar results. We choose the entrance time because we have the predicted arrival time for each vehicle. Thus, it is easy to implement the algorithm. Then we use the top N_{pop} individuals from the original population and offspring to form the new population. If any of the stopping criteria (maximum number of iterations or the best solution is not updated for a certain number of generations) are met, then the algorithm terminates. Otherwise, the algorithm repeats the steps inside the *while* loop.

5.2.1 Chromosome Encoding and Feasibility Check

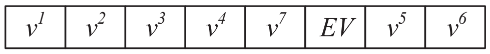


Figure 5.3: An example of the permutation encoding scheme, the left-most vehicle has the highest priority while the right-most one has the lowest priority.

Instead of using the popular binary encoding scheme for genetic algorithms, we choose to use permutation encoding scheme which is more suitable to find an optimal sequence for vehicle ordering. As shown in Figure 5.3, the individual corresponds to a sequence of vehicles which is $\{v^1, v^2, v^3, v^4, v^7, EV, v^5, v^6\}$ with v^1 on the leftmost is the first and the rightmost vehicle v^6 is the last one. Different chromosomes denote different sequences of vehicles. Once an individual is created, it is not always true that the corresponding sequence is a feasible one since vehicles' order on EV's lane cannot be altered. Every newly generated individual should be checked against the sub-sequence of vehicles on EV's lane for feasibility. Figure 5.4 is provided to have a visual impression of the situation when the vehicles' sequence needs to be optimized. In the figure, v^1, v^3 and EV are the vehicles on EV's lane whose order could not be altered. And note that except vehicles on EV's lane, all other vehicles who are not a head vehicle are not part of the sequence. The vehicles from South and West who are not head vehicles are such vehicles that will be confirmed only after EV exits. If an

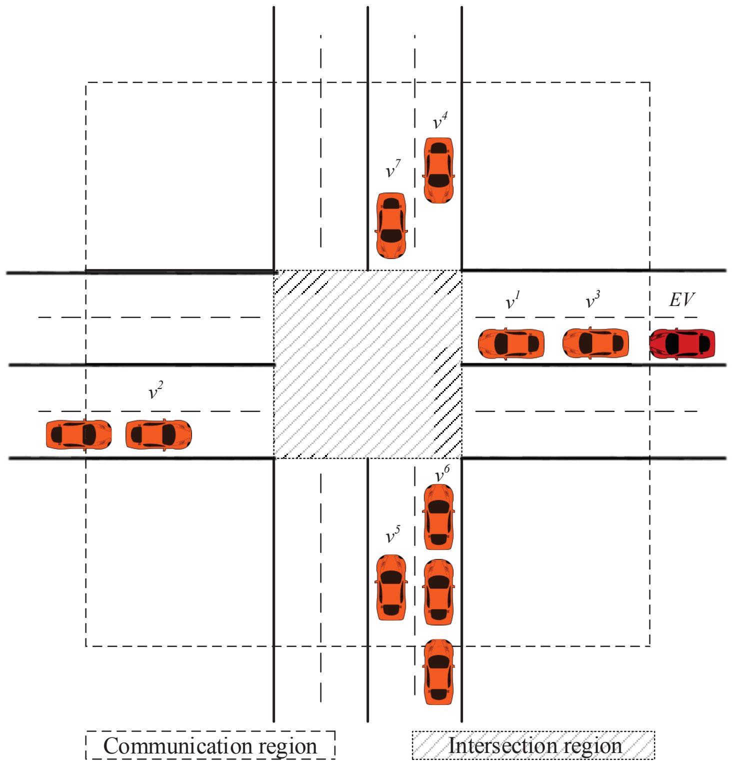


Figure 5.4: Example situation of vehicles whose sequence to be optimized in intersection space, note: vehicles on EV's lane are: v^1, v^3, EV.

individual is not feasible, the corresponding bits of vehicles on EV's lane should be changed to conform to the correct relative order. The function $feasibilityCheck()$ is making the corresponding modifications on an infeasible individual. An example of adjustment according to the sequence of vehicles who are ahead of the EV on the same lane is shown in Figure 5.5.

5.2.2 Crossover and Mutation

Two individuals perform crossover to generate offspring if they are selected to be parents. The offspring inherit features (i.e. gene structures) from their parents. Different encoding scheme has different crossover operator since they have different

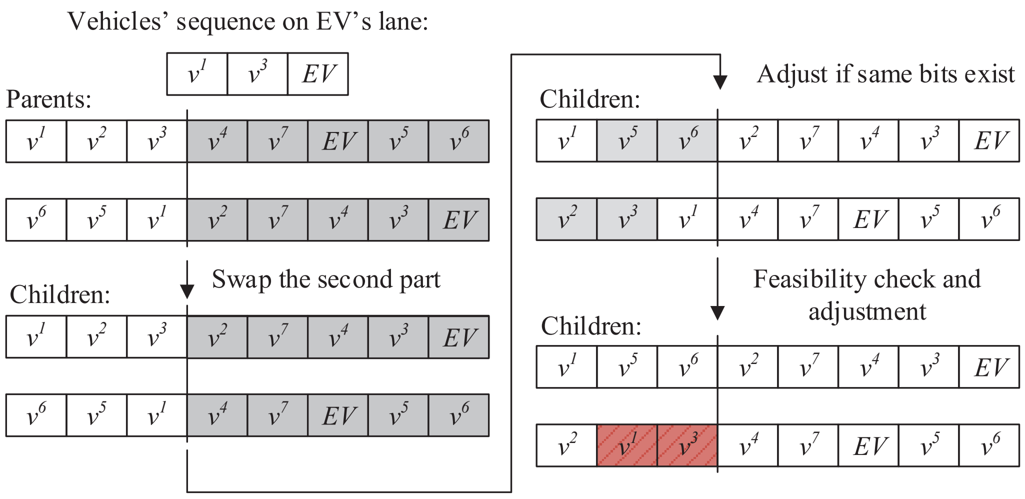


Figure 5.5: An example of one-point crossover. Relative ordering of parents is preserved when the chromosomes are adjusted due to the existence of same bits. Feasibility is checked for the two children based on vehicles' sequence on EV's lane and adjustments are made.

gene structures. For the most popular binary encoding scheme, it is easy to do crossover and mutation since a chromosome only contains binary bits. For our permutation encoding scheme, we choose to apply one-point crossover [110] which is implemented in the function $crossOver()$. As shown in Figure 5.5, the same bits may exist in one chromosome after the parts behind the randomly chosen position are swapped. In the second step in the figure, the two children have same bits $\{v^2, v^3\}$ and $\{v^5, v^6\}$ respectively. To generate correct chromosomes, we adjust the chromosome of one child by swapping those same bits from another child's chromosome while preserving the relative ordering of parents. Note that the new chromosomes may also not be feasible since the order of vehicles on EV's lane in a chromosome may not be the same as the actual order. If this happens, since the order of the vehicles on EV's lane cannot be changed, we manually adjust the relative order of vehicles to be the correct order to have a feasible chromosome. For example in Figure 5.5, we adjust the order of v^1 and v^3 for the second child in the last step. Feasibility check and adjustment are done by the function $feasibilityCheck()$.

Similar to probabilistically selecting two individuals for crossover, we apply mutation on produced chromosomes based on a given probability by the function $mutation()$. Different with binary encoding scheme's mutation which could be done by simply changing the value of a randomly selected bit from 1 to 0 or 0 to 1, our permutation encoding scheme exchanges the bits on two randomly chosen positions to obtain a new chromosome. As shown in Figure 5.6, positions of v^2 and EV are randomly chosen to exchange values and feasibility check based on vehicles' order on EV's lane is performed after mutation.

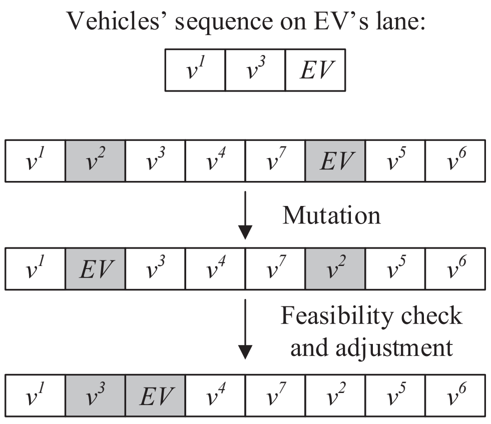

Vehicles' sequence on EV's lane:
v^1 v^3 EV
v^1 v^2 v^3 v^4 v^7 EV v^5 v^6
Mutation
v^1 EV v^3 v^4 v^7 v^2 v^5 v^6
Feasibility check and adjustment
v^1 v^3 EV v^4 v^7 v^2 v^5 v^6

Figure 5.6: An example of mutation. Feasibility is checked for the mutated chromosome based on vehicles' sequence on EV's lane and adjustment is made. v^2 and EV are randomly chosen to swap to perform mutation. EV is swapped with v^3 to conform to vehicles' sequence on EV's lane.

5.2.3 Fitness and Generating New Generation

The reciprocal of the entrance time of the emergency vehicle is defined as the fitness of an individual in our proposed GA. The entrance time of the emergency vehicle can be determined as discussed in Section 5.1. For an individual in a population, the higher the fitness is, the closer the corresponding solution is to the optimal solution. Among all individuals in the population and the offspring produced, the top N_{pop} individuals with respect to the fitness values are selected to form the next generation.

5.2.4 Stopping Criterion

The constant N_{max} represents the number of maximum generations and $N_{noChange}$ represents the number of continuous generations that solutions are not changed. As shown in Algorithm 4, if the best solution is not updated after $N_{noChange}$ generations or the N_{max}th generation has been reached, then the algorithm terminates and stops searching for a better solution.

5.3 Simulation

The performance of the proposed optimization approach for EVs is evaluated against the DICA and a reactive traffic light algorithm which is explained below. All simulations are implemented in the traffic simulator SUMO [103]. The default traffic management for intersections in SUMO is not used and the control algorithms are programmed as Python applications. The TraCI is used for the interaction between the Python applications and SUMO. Configurations for intersections in the simulation and corresponding results are described in this section, followed by discussions on obtained results.

5.3.1 Simulation Setup

Extensive simulations of different traffic volumes are performed on an isolated perfect 4-way intersection where each approach has three incoming lanes and two exit lanes. Most simulation setups in this chapter are similar to that in previous chapters. All roads have the speed limit of $v_m = 70km/h$. The maximal acceleration

(a_{max}) and deceleration (a_{min}) for all vehicles are set to be $2m/s^2$ and $-4.5m/s^2$. In the simulation, for simplicity we used the same size for normal vehicles and EVs that they both have 5 meters length and 1.8 meters width. We let vehicles approach an intersection with different speeds when they enter into the communication region of the intersection to make the simulation more realistic. In detail, when a new vehicle is spawned outside of the communication region, the speed of the vehicle is set with a random value within the range from 40% to 100% of the maximum allowed speed v_m. We set the distance between the enter lines of the communication region and the intersection region as $50m$.

Vehicles are generated randomly on each road with a randomly assigned intersection route. Every generated vehicle has the probability of p_{EV} to be an EV, otherwise it will be a normal vehicle. In our simulation, an emergency vehicle is generated only when there is no such vehicle inside the communication region. To create variations on the traffic pattern, we use several different random seed numbers to generate different traffic patterns and make the simulations reproducible. Table 5.1 summarizes the parameters used for various traffic volumes and patterns that were employed in many of our simulations where p_V corresponds to traffic volumes, p_L, p_S and p_R are the probabilities for a generated vehicle to take Left, Straight, and Right routes respectively. For every traffic volume, we run three simulations with different traffic patterns and then use the averages of these simulation results as the result for each traffic volume case. For the genetic algorithm, we set $N_{pop} = 100, N_{noChange} = 10, P_c = 0.85, P_m = 0.05$ and $N_{max} = 100$.

Simulations were run by 0.05s time step. We terminate each simulation when the simulation time reaches one hour. The definitions of *simulation time* and *computation time* can be found in last chapter.

Table 5.1: Parameters used for various traffic volumes and patterns. ($*$ Expected number of vehicles per 10 minutes.)

Parameter	Value
Traffic volumes$*$	100 / 200 / 300 / 400 / 500
p_V	0.03 / 0.06 / 0.08 / 0.11 / 0.14
p_L	0.20
p_S	0.60
p_R	0.20
p_{EV}	0.02
Random seeds	12 / 21 / 66

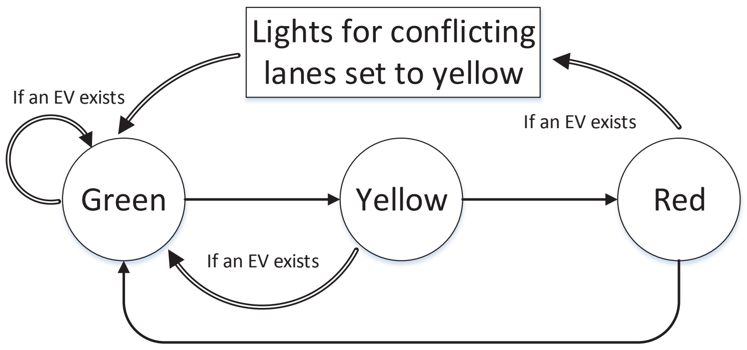


Figure 5.7: Reactive traffic light diagram.

5.3.2 Reactive Traffic Light

To show the effectiveness of the proposed R-DICA for emergency vehicles, a reactive traffic light algorithm for emergency vehicles is implemented and tested. As shown in Figure 5.7, the traffic light for the lane of an EV changes to green as quickly as possible when an EV is detected on the boundary of the communication region. Arrows with single line represent the state transitions (i.e. conventional traffic light algorithm) when there is no EV inside the communication region while arrows with double lines show the actions the algorithm will perform if an EV exists. The conventional traffic light algorithm we used is the default traffic light implemented in SUMO which has 31, 13 and 83 seconds durations for green, yellow and red light phases respectively. As shown in Figure 5.7, if the current status is yellow or green when an EV is detected, the algorithm changes the light back to green or just extends the green light for a fixed amount of time respectively. If the current status of the lane is red when an EV enters the communication region, the algorithm immediately sets the green lights of conflicting lanes to yellow and then the lane of the EV will have green light after the yellow phase of the conflicting lanes. This augmentation of reactive mechanism in traditional traffic light system certainly help an EV to cross an intersection as quickly as possible.

5.3.3 Simulation Results

Performances of three different traffic patterns for all five volume cases are recorded from simulations. Figure 5.8 shows a series of screenshots of simulation employing R-DICA in SUMO when an EV (the vehicle in red) is crossing the intersection from South. In the simulation, normal vehicles in yellow are not confirmed by the ICA while green normal vehicles are the confirmed ones. In Figure 5.8 (a), we can see that R-DICA activated GA algorithm which establishes an optimal order of vehicles to expedite the crossing of the EV. As one can note that in Figure 5.8 (a) and (b) the head vehicle on the right lane of West road is not confirmed which means that this vehicle has a lower priority than the EV. All vehicles whose DTOTs cannot be modified are confirmed vehicles. And head vehicles who have a higher priority than the EV are confirmed. Figure 5.8 (b) and (c) show that the EV is crossing the intersection unhindered while lower priority vehicles are waiting before the intersection. As shown in Figure 5.8, as soon as the EV exits the intersection, all head vehicles get confirmed which means R-DICA operates the same way as DICA. The optimal vehicle-passing sequence from the genetic algorithm ensures the fast crossing of an intersection for the EV.

Computation Time

To show the computational efficiency of R-DICA using GA, we implemented R-DICA in two different versions: One with GA and the other one with the Exhaustive

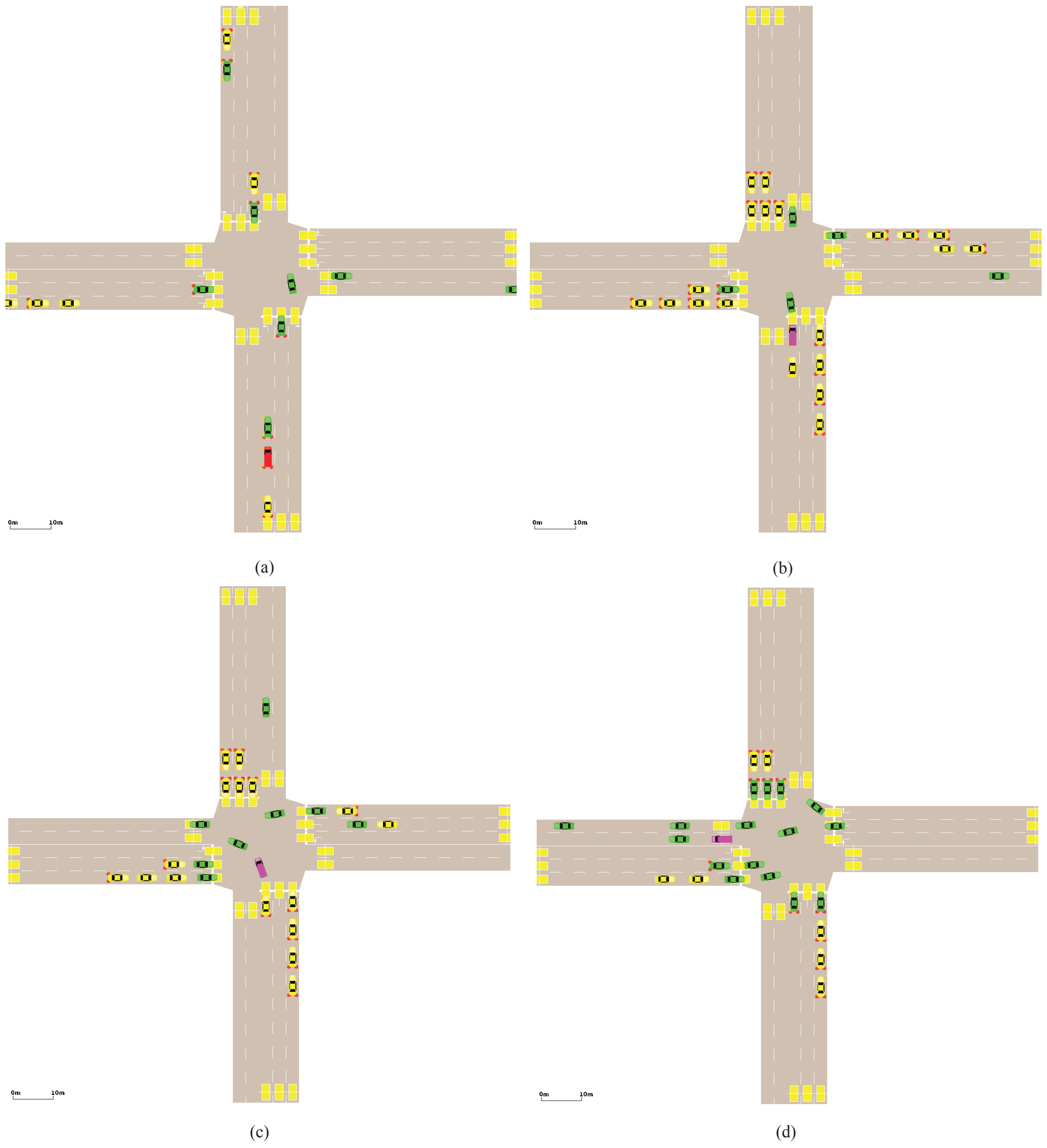
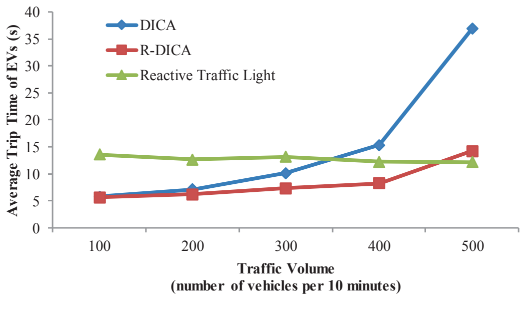


Figure 5.8: A series of screenshots of simulation which illustrates a situation when an EV is crossing the intersection.



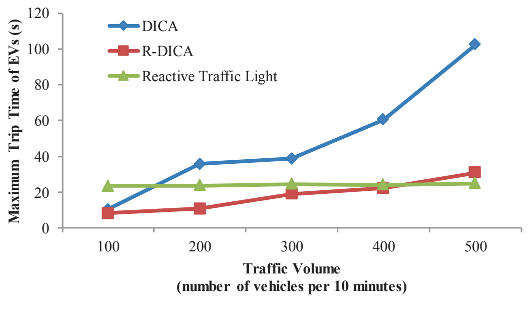


Figure 5.9: Performance comparison of EVs for DICA, R-DICA, and the reactive traffic light: (a) average trip time, (b) maximum trip time

Search (ES) method to solve the optimization problem. Computation times of different volume cases are recorded for both methods. Simulation results are shown in Table 5.2 where 'N/A' means that the computer was not able to complete the simulation due to memory errors.

Table 5.2: Computation time comparison between ES and GA

Traffic volume (Number of vehicles per 10 minutes)	100	200	300	400	500
Computation time of ES (h)	0.05	0.52	N/A	N/A	N/A
Computation time of GA (h)	0.05	0.15	0.35	0.65	0.72

From the result, we can see that R-DICA with GA has definite advantages over R-DICA with ES in terms of computational efficiency. As shown in the table, the exhaustive search method only works for light traffic volumes while it has memory issues for traffics of higher volumes.

Performance of EVs

We obtained the performance measures introduced in Section 3.5.2 to compare the performance of R-DICA with DICA and the reactive traffic light.

As shown in Figure 5.9, the average trip times of EVs in all three algorithms: DICA, R-DICA and the reactive traffic light are compared. For traffic volumes from 100 to 400, R-DICA has the least average trip time of EVs than the other two algorithms. Especially in light traffic volumes, R-DICA reduces the EVs' average travel time by more than 50% from the reactive traffic light. However, the algorithm has a bit longer average trip time for EVs than that of the reactive traffic light in 500 traffic volume. The worst case for EVs' travels is illustrated by the maximum trip time of EVs in (b) of Figure 5.9. The maximum trip time of R-DICA increases and becomes greater than that of the reactive traffic light. Both the average trip time and the maximum trip time of EVs for DICA are increasing along the volumes. One may note that the average trip time and the maximum trip time of the reactive traffic light keep almost the same with the increase of the traffic volume. Through observation of the simulations, part of this is because too many vehicles accumulate before the intersection when the lane of the EV is under red light. At this situation, the EV is not detected and is stopping outside the communication region. When the light for the lane of the EV turns green, the EV accelerates from rest to enter the communication region which results in a higher speed for the EV. Thus, for the heavier traffic volumes, EVs always have a higher speed when detected and are expedited to cross by preference. The trip time within the communication region is then reduced compared with R-DICA.

Performance of Normal Vehicles

Comparison of the performance for normal vehicles for all three algorithms is shown in Figure 5.10. From this result, we can see that the throughput and effective average trip time of R-DICA are nearly the same as those of DICA which shows that the performance of normal vehicles is minimally affected by EVs. Both the throughput and effective average trip time of normal vehicles become worse with the increase of traffic volumes. This is consistent with the result in our previous

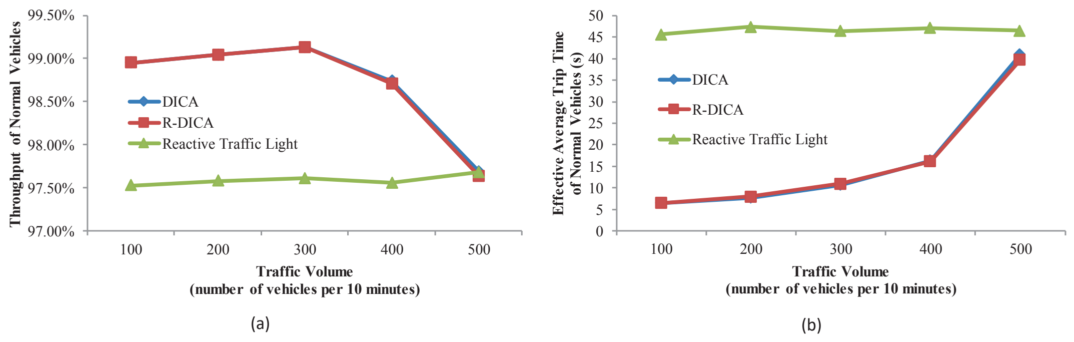


Figure 5.10: Performance comparison of normal vehicles for DICA, R-DICA, and the reactive traffic light: (a) throughput, (b) effective average trip time

work [45, 47]. Also, one can see from Figure 5.10 that the reactive traffic light has steady and worse performance for normal vehicles than the other two algorithms.

To investigate more about the potential negative effects of prioritizing EVs on other normal vehicles, we compare the maximum trip time of normal vehicles for DICA and R-DICA in Table 5.3. The maximum trip time of R-DICA is very close to that of DICA and their difference increases with traffic volumes. This shows that it will bring a more negative effect on normal vehicles to evacuate an EV in congested traffic.

Table 5.3: Comparison of maximum trip times of normal vehicles between DICA and R-DICA

Traffic volume (Number of vehicles per 10 minutes)	100	200	300	400	500
Maximum trip time of normal vehicles: DICA (s)	17.88	35.80	55.37	89.47	132.90
Maximum trip time of normal vehicles: R-DICA (s)	18.62	38.52	61.68	99.15	150.00

5.4 Summary

In this chapter, we have shown that the DICA algorithm can be augmented to allow emergency vehicles to cross intersections faster. A genetic algorithm based approach is proposed as part of the augmented algorithm, called R-DICA, to optimize the sequence of vehicles which gives the emergency vehicle the highest priority and keeps the influence on other vehicles' travel times as minimum as possible. The R-DICA operates the same way as DICA if there is no EV inside the communication region and optimizes vehicle-passing sequence if an EV enters the communication region. A reactive traffic light and DICA algorithms are also implemented for simulation and their results are compared with R-DICA to evaluate the performance of R-DICA. Simulation results show that R-DICA is effective to reduce travel times of EVs and has better performance than the reactive traffic light for normal vehicles. We conclude that the performance of normal vehicles is not noticeably affected based on the simulation results of DICA and R-DICA.

CHAPTER SIX

MICA: Optimal Control of Autonomous and Connected Intersection Traffic

DICA's strategy to obtain a collision-free DTOT for a new vehicle is conservative by delaying the vehicle until all confirmed vehicles can cross the intersection safely. The strategy might introduce an unnecessary delay for a new vehicle i. For example, let's say vehicle i is space-time conflicting with another confirmed vehicle j whose speed is slower. DICA will force vehicle i to slow down waiting for vehicle j to clear the conflicting area. In fact, by proper coordination, vehicle i can cross the conflicting area before vehicle j by accelerating to reduce travel time and have no collisions. Note here, vehicles' arriving order may not be preserved whereas the performance of overall traffic would be improved. Instead of employing the simple delay strategy, this chapter proposes an approach named MICA (Mixed Integer Programming based Intersection Coordination Algorithm) to optimize a new head vehicle's trajectory while potential collisions with confirmed vehicles are avoided.

6.1 Vehicle-Intersection Interaction

Figure 6.1 shows the high-level interaction between a CAV and the ICA in MICA. As introduced in previous chapters, a CAV becomes a head vehicle only when no vehicles exist between the CAV and the enter line of the intersection, or the vehicle which is immediately in front of the CAV has begun to enter the intersection (i.e.,

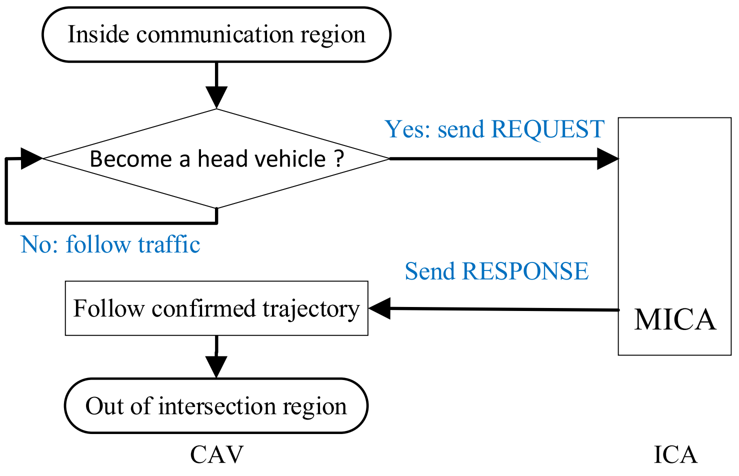


Figure 6.1: Interaction between a CAV and ICA.

has crossed the enter line). If a vehicle becomes a head vehicle, it immediately sends a REQUEST message to the ICA in order to request an intersection crossing trajectory to follow. We call such a trajectory a *confirmed trajectory* which is a sequence of timed states. In DICA, the term DTOT denotes Discrete-Time Occupancy Trajectory which might not always represent the optimal trajectory due to the conservative collision avoidance strategy. With the introduction of MIP optimization, we redefine the term DTOT as Discrete-Time Optimal Trajectory in this chapter since MICA provides the optimal trajectory for every head vehicle. Thus, a confirmed trajectory is also called a *confirmed DTOT* in the sequel. It is assumed that a REQUEST message contains all necessary information such as the vehicle's size, current speed, current position, intended intersection crossing route, etc. for the ICA to generate a confirmed DTOT. Once a confirmed DTOT is generated, the ICA sends the DTOT to the vehicle in a RESPONSE message. When a head vehicle receives its confirmed DTOT, it becomes a *confirmed vehicle*. A vehicle which is not a head vehicle will simply follow other vehicles autonomously. Note that the ICA can check whether a vehicle is a head vehicle or not easily based on the information collected from the detectors installed at the communication region boundary. Therefore, REQUEST messages accidentally from non-head vehicles are simply ignored by the ICA.

6.2 Problem Formulation of MICA

In this section, we introduce how we formulate the trajectory optimization problem for each head vehicle and also the overall MICA algorithm. We solve the problem of generating the optimal crossing trajectory for every vehicle through a mixed integer programming formulation which includes necessary collision avoiding constraints. The formulation ensures every vehicle can cross an intersection as fast as possible while potential collisions among vehicles are safely avoided.

6.2.1 Single Vehicle Trajectory Generation

We first consider the case of trajectory generation for a single vehicle. We assume that there is only one vehicle approaching an intersection. Then in this very simple situation, we aim to generate the DTOT for the vehicle to follow so that it can cross the intersection as quickly as possible while satisfying some constraints like speed and acceleration limits.

Let γ be the intersection crossing route (or path) the vehicle will follow when it crosses an intersection and also let s be the state variable which represents the position of the vehicle along the route γ. We assume that γ is fixed for a given pair of enter and exit lanes across an intersection (i.e., a vehicle will not change lanes during its intersection crossing). We also use h to denote the fixed sampling time, v_{max} to denote the maximum allowed speed for the vehicle, a_{max} and a_{min} to denote the maximum and minimum acceleration rates respectively. Note that, given a fixed h, the DTOT of a vehicle can be represented by a sequence of positions of the vehicle along γ such as $\{s_0, s_1, s_2, \ldots, s_k, \ldots, s_n\}$ for some positive integer number n where s_k represents the position of the vehicle along γ at the kth time step. Here, we implicitly assume that $s_i \leq s_j$ if $i < j$ and this relation will be included as a constraint in our optimization problem formulation. We choose n as a sufficiently large number. Let $\mathbf{s} = [s_0\ s_1\ \cdots\ s_n]^T \in \mathbb{R}^n$. Then the fastest crossing DTOT for the vehicle can be obtained by

$$\max_{\mathbf{s} \in \mathbb{R}^n} s_n \tag{6.1}$$

under several constraints which will be described below.

The main challenge is how to express all constraints as a function of elements of the vehicle's DTOT so that a linear constrained optimization problem can be derived at the end. The first constraint that we should consider is the ordering relation between any pair of s_i and s_j in $\mathbf{s} \in \mathbb{R}^n$ where $i, j \in [0, n]$ and $i \neq j$. Since $\mathbf{s}$ should represent an ordered sequence of positions of the vehicle along γ at each sampling time step, it is clear that we should constrain the values for each variable s_k in $\mathbf{s}$ as follows:

$$s_k^{lb} \leq s_k \leq s_k^{ub} \tag{6.2}$$

where

$$\begin{aligned} s_k^{lb} &= 0 \ \text{ if } k = 0, & s_k^{lb} &= s_{k-1} \ \text{ if } k \neq 0; \\ s_k^{ub} &= s_{k+1} \ \text{ if } k \neq n, & s_k^{ub} &= v_{max} \cdot n \cdot h \ \text{ if } k = n. \end{aligned}$$

Next, vehicles should not drive backward on roads and should obey the speed limit which is enforced by laws. Thus, we have the following constraint for speed:

$$0 \leq v_k \leq v_{max} \tag{6.3}$$

where v_k is the speed of the vehicle at the kth time step. In order to incorporate this constraint into our optimization formulation in (6.1), we rewrite the velocity variable v_k in terms of the positions s_k and s_{k-1} in $\mathbf{s}$ as

$$v_k = \frac{s_k - s_{k-1}}{h} \qquad \text{for } k \in [1, n]. \tag{6.4}$$

In addition to the speed constraint, we also should consider constraints for the acceleration (or deceleration) rate of a vehicle in order to make sure the computed DTOT is dynamically feasible. Let a_k be the acceleration rate of a vehicle at the kth time step. Then, the constraint for acceleration a_k is given by

$$a_{min} \leq a_k \leq a_{max} \qquad \text{for } k \in [1, n]. \tag{6.5}$$

Again, in order to incorporate these constraints into the optimization formulation in (6.1), it is necessary to rewrite a_k in terms of variables s_k in $\mathbf{s}$ as follows:

$$\begin{aligned} a_1 &= \frac{v_1 - v_0}{h} = \frac{s_1 - s_0 - v_0 h}{h^2}, \\ a_k &= \frac{v_k - v_{k-1}}{h} = \frac{s_k - 2s_{k-1} + s_{k-2}}{h^2} \quad \text{for } k \in [2, n] \end{aligned} \tag{6.6}$$

where v_0 is the speed of the vehicle at the 0th time step which we assume is known to the ICA via direct measurement or the V2I message sent from the vehicle to the ICA at that moment.

6.2.2 Mixed Integer Programming for Collision Avoidance

In this section, we generalize the DTOT generation problem to include situations when there are other vehicles also approaching the same intersection. In such situations, we should develop additional constraints and include them in the optimization process so that the vehicle of interest can successfully cross the intersection while potential collisions with other vehicles are safely avoided. In our discussions below, we use the superscript i to indicate the vehicle of interest.

Constraints from space-conflicting vehicles

Suppose that there is a set $\mathcal{C}$ of confirmed vehicles which have arrived at the communication region earlier than vehicle c^i and hence their DTOTs have already been generated and confirmed. Let $\mathcal{C}^*$ be the set of vehicles in $\mathcal{C}$ whose routes are conflicting with vehicle c^i's route γ^i. Now, let us consider another vehicle $c^j \in \mathcal{C}^*$ to derive collision avoidance constraints for c^i's DTOT generation. We first note that, as discussed in our previous work [47], for any pair of intersection crossing routes whose both ends (entering and exiting of an intersection) are different from each other, space conflicting (or overlapping) occurs only at certain small portions of their routes. Also, one can see that these space conflicting short ranges along routes can be determined offline for all pairs of conflicting intersection crossing routes. Thus, to make vehicle c^i avoid any potential collisions with vehicle c^j, it is necessary to ensure that c^i does not occupy the space conflicting area when c^j occupies that area.

Figure 6.2 illustrates an example situation when vehicle c^j is already a confirmed vehicle as vehicle c^i is entering the communication region. In this situation, c^i is the head vehicle on its lane and being considered for confirmation by the ICA (i.e., the DTOT is being generated by the ICA). The two long lines with arrows in the figure represent vehicles c^i and c^j's routes: γ^i and γ^j. We can see that their routes have a conflicting area A (with red diagonal lines) inside the intersection region which is the area that both of the vehicles should not occupy at the same time to avoid collisions. In the figure, $P^i_{in,j}$ denotes the position on γ^i that c^i starts to enter the conflicting area A and $P^i_{out,j}$ denotes the position on γ^i that c^i becomes free from collisions with c^j. Note that we represent the position of a vehicle by the position of the center of the vehicle's front bumper along its route. Thus we choose $P^i_{out,j}$ to be at vehicle c^i's length (L^i) away from the right boundary of A since the length of a vehicle should also be considered for collision avoidance. Note also that the positions $P^i_{in,j}$ and $P^i_{out,j}$ along γ^i are assumed to be known since they can be computed offline for every pair of intersection crossing routes and, in the sequel, we use $\gamma^i|_j$ to denote the conflicting area with c^j on γ^i.

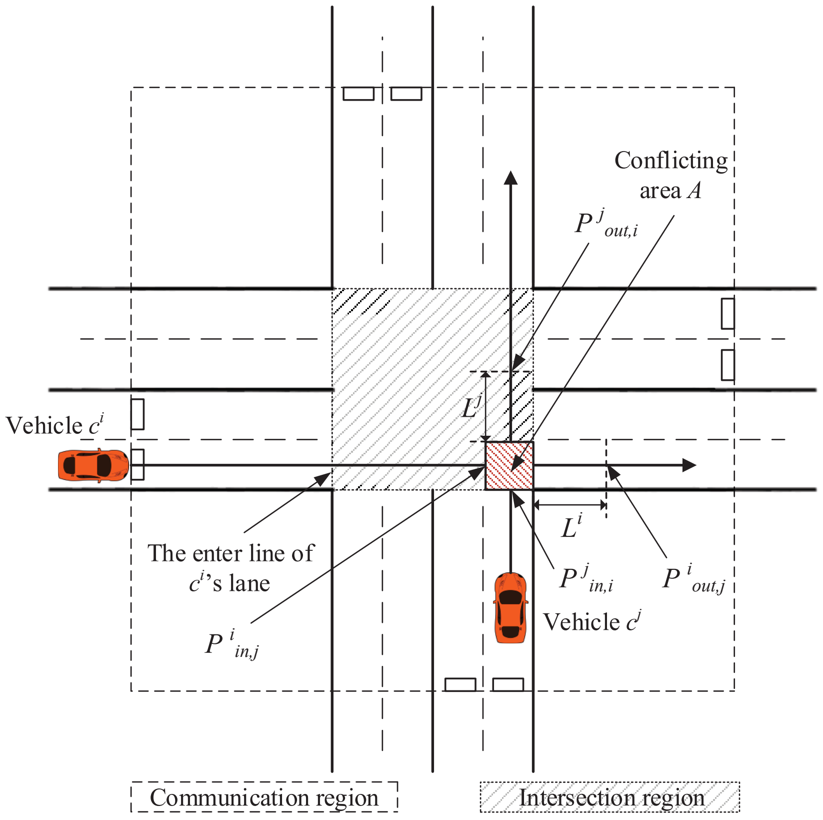


Figure 6.2: Example of conflicting area.

We can see that, for c^i, there are two options to avoid any potential collisions with c^j: (i) exits $\gamma^i|_j$ before c^j arrives at its $\gamma^j|_i$, or (ii) arrives at $\gamma^i|_j$ after c^j exits its $\gamma^j|_i$. Let $p^i_{in,j}$ and $p^i_{out,j}$ be the distances from vehicle c^i's current position to $P^i_{in,j}$ and $P^i_{out,j}$ along γ^i respectively. Then the two options can be expressed in two inequalities:

$$s^i_{\tau(in,j)} \geq p^i_{out,j} \tag{6.7}$$

$$s^i_{\tau(out,j)} \leq p^i_{in,j} \tag{6.8}$$

where s^i_k is the c^i's position on γ^i at the kth time step, $\tau(in, j) \in \{0, 1, \dots, n\}$ is the time step when c^j arrives at position $P^j_{in,i}$ and $\tau(out, j) \in \{0, 1, \dots, n\}$ is the time step when c^j completely leaves $\gamma^j|_i$, i.e., c^j arrives at position $P^j_{out,i}$. Note that $\tau(in, j)$ and $\tau(out, j)$ can be easily computed based on c^j's confirmed DTOT which is known to the ICA.

However, the two inequalities are not compatible with each other and no solution exists if we include both of them in the same problem formulation. To capture this logical condition when solving the optimization problem for trajectory generation for c^i, we adopt a binary variable $b^i_j \in \{0, 1\}$ to represent c^i's behavior with respect to c^j and reformulate our optimization problem in (6.1) as a mixed-integer linear programming (MILP) problem. Then, the constraints for collision avoidance with the conflicting vehicle c^j can be represented as

$$s^i_{\tau(in,j)} \geq p^i_{out,j} + M \cdot (b^i_j - 1) \tag{6.9}$$

$$s^i_{\tau(out,j)} \leq p^i_{in,j} + M \cdot b^i_j \tag{6.10}$$

where M is an arbitrarily large number. If $b^i_j = 1$, (6.7) holds which means that c^i is already out of the conflicting area A at the time c^j starts to enter the conflicting area. Similarly, if $b^i_j = 0$, (6.8) holds meaning that c^i is still before the conflicting area at the time c^j exits the conflicting area A. Thus, the binary variable b^i_j ensures that the two equations cannot hold at the same time. The choice of either option for c^i becomes a decision variable in the optimization formulation.

Now we have modeled the constraints from one conflicting vehicle in $\mathcal{C}^*$. Following the same process, we can easily include all other vehicles in $\mathcal{C}^*$ by computing the corresponding times and position parameters related to their conflicting areas with vehicle c^i.

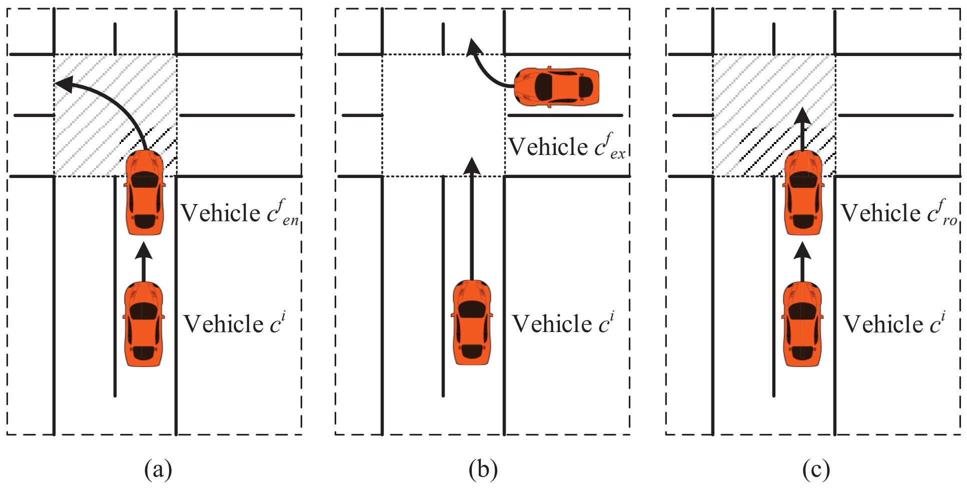


Figure 6.3: Examples of three types of front vehicles (we use the simplest intersection structure to save space).

Constraints from front vehicles

Besides vehicles in $\mathcal{C}^*$ considered above, c^i's immediate front vehicles may also have a direct influence on c^i's crossing behavior since wrong trajectory planning may cause rear-end collisions with its front vehicles. As shown in Figure 6.3, to cross an intersection safely, c^i needs to avoid rear-end collisions with three different types of front vehicles: (i) Figure 3(a), a front vehicle with the same entrance lane (c^f_{en}), (ii) Figure 3(b), a front vehicle with the same exit lane (c^f_{ex}), and (iii) Figure 3(c), a front vehicle with exactly the same route (c^f_{ro}). Below, we discuss how to formulate constraints to the motion of vehicle c^i to avoid collisions with these three types of front vehicles. In the following discussion, note that front vehicles' intersection crossing trajectories are already known to the ICA since they are already confirmed.

First, let us consider the case of collision avoidance with a front vehicle c^f_{en}. If we use $\tau(en, f)$ to denote the time step when c^f_{en} enters the enter line of an intersection and $v^f_{\tau(en,f)}$ is the speed of c^f_{en} at that moment, then, to ensure that vehicle c^i will not collide with c^f_{en} while c^i is approaching the intersection, we constrain the motion of

c^i so that both vehicles are separated in time by at least t_{sep}. t_{sep} can be determined conservatively by assuming that c^i is currently approaching the intersection at its maximum speed v_{max}. Thus we define t_{sep} as

$$t_{sep} = \frac{v_{max} - v^f_{\tau(en,f)}}{|a_{min}|} + \frac{g_{min}}{v_0} + \delta \tag{6.11}$$

where g_{min} is a constant that represents the minimum distance gap between two adjacent vehicles on a same lane regardless of their speeds and δ is another constant that represents a safety margin in time. The first term in (6.11) is the time c^i needs to decelerate from v_{max} to $v^f_{\tau(en,f)}$ with its maximum deceleration rate while the second is the minimum time required to ensure the minimum distance gap g_{min}. Then we can formulate the following constraint to ensure that c^i arrives at the intersection enter line after $\tau(en,f)$ with a certain amount of time margin:

$$s^i_{\tau(en,i)} \leq p^i_{en} \tag{6.12}$$

where $\tau(en,i) = \tau(en,f) + \left\lceil \frac{t_{sep}}{h} \right\rceil$ is the earliest time step at which c^i can arrive at the enter line of the intersection, and p^i_{en} is the position of the enter line on γ^i.

Next, to avoid collisions with a front vehicle c^f_{ex}, we develop a constraint using the same idea that introduces a minimum separation time t_{sep} between c^i and c^f_{ex} when they exit an intersection. Specifically, we make c^i exit the intersection at least t_{sep} time later after c^f_{ex}. In this case, t_{sep} can be obtained in the same way as in (6.11) except that $v^f_{\tau(en,f)}$ must be replaced by $v^f_{\tau(ex,f)}$ in the first term of the equation where $\tau(ex,f)$ is the time step when c^f_{ex} exits the intersection and hence $v^f_{\tau(ex,f)}$ is the speed of c^f_{ex} at that moment. Following the similar process, we can have the following constraint for c^i to avoid collision with c^f_{ex}:

$$s^i_{\tau(ex,i)} \leq p^i_{ex} \tag{6.13}$$

where $\tau(ex,i) = \tau(ex,f) + \left\lceil \frac{t_{sep}}{h} \right\rceil$ is the earliest time step at which c^i can exit the intersection, and p^i_{ex} is the position of the intersection exit on γ^i.

Now, note that if another confirmed vehicle has both the same entrance lane and the same exit lane with vehicle c^i, they actually have the same route. Thus, to avoid collisions with such a front vehicle c^f_{ro}, we can simply combine constraints from the first two types of front vehicles. We include both (6.12) and (6.13) at the same time as constraints in our optimization formulation for vehicle c^i to avoid collisions with this type of front vehicles.

6.2.3 Optimization Problem Formulation

We now present the final form of the proposed optimization problem formulation to generate the DTOT for vehicle c^i that avoids collisions with all confirmed vehicles. Collecting variables for optimization from (6.1), (6.9), and (6.10), we first define the vector of decision variables $\mathbf{x} := \{s^i_1, s^i_2, \cdots, s^i_n, b^i_1, b^i_2, \cdots, b^i_l\}$ assuming that $|\mathcal{C}^*| = l$. Let $\mathbf{x}(k)$ denotes the kth element of $\mathbf{x}$. Then, the final form of the optimization problem is to find $\mathbf{x}^*$ such that

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathbb{R}^n \times \{0,1\}^l}{\operatorname{argmax}} \ \mathbf{x}(n) \tag{6.14}$$

subject to constraints

- Eqs. (6.2), (6.3), and (6.5).
- For vehicle $c^j \in \mathcal{C}^*$, eqs. (6.9) and (6.10).
- For c^f_{en} or c^f_{ro} type front vehicle, eq. (6.12)
- For c^f_{ex} or c^f_{ro} type front vehicle, eq. (6.13)

where v^i_k and a^i_k is defined as in (6.4) and (6.6), respectively.

6.2.4 MICA

Based on the problem formulation introduced above, we now present the overall MICA intersection control framework. As shown in Algorithm 5, we first call the function `getConstraintsFV()` to obtain the constraints due to c^i's front vehicles as explained in Section 6.2.2. Then, the function `getConstraintsSCV()` is called to determine the constraints due to confirmed vehicles whose routes are space conflicting with route γ^i as introduced in Section 6.2.2. After all required constraints for collision avoidance are obtained, MICA computes the DTOT for c^i via the function `MIP()`. With the problem formulation in Section 6.2.3 and the obtained constraints, `MIP()` can be solved using existing optimization solvers like CPLEX, Gurobi, etc. Note that we omit the details on functions `getConstraintsFV()` and `getConstraintsSCV()` and recommend to refer to our earlier work [47] for more details.

Algorithm 5 MICA (Mixed Integer Programming based Intersection Coordination Algorithm)

```
1: Let $\mathcal{C}$ be the set of confirmed vehicles.
2: Let $c^i$ be the vehicle to be considered for confirmation.
3:
4: Call getConstraintsFV($\mathcal{C}, \gamma^i$) $\rightarrow$ constFV
5: Call getConstraintsSCV($\mathcal{C}, \gamma^i$) $\rightarrow$ constSCV
6: Call MIP(constFV, constSCV) $\rightarrow \mathbf{x}^*$
7:
8: Send $\mathbf{x}^*(1, 2, \cdots, n)$ to vehicle $c^i$
```

6.3 Simulation

Simulations were implemented in SUMO [103] to show the performance of MICA. The algorithm was programmed as Python applications which interact with SUMO through its API: TraCI. The mixed integer programming optimization problem was solved using Gurobi optimization software [111].

6.3.1 Simulation Setup

In simulations, we use a four-way intersection that has three entrance lanes and two exit lanes each way. Most simulation setups in this chapter are similar to that in previous chapters. The maximum allowed speed v_{max} for all vehicles is 70 km/h. Vehicles enter the communication region of the intersection with different speeds which are randomly chosen within the range from 40% to 100% of v_{max}. All vehicles have the same rectangular shape and size (5 meters long and 1.8 meters wide). We set $a_{max} = 2\ m/s^2$ and $a_{min} = -4.5\ m/s^2$, respectively. The distance from the enter line of the communication region to that of the intersection region is set as 50 m.

Vehicles were spawned randomly on each entrance lane with an intersection route which is also randomly chosen. We used several different random seeds to generate different traffic patterns and make simulations reproducible. In Table 6.1, the parameters used for generating various patterns and volumes are summarized in which pL, pS and pR are the probabilities for a new generated vehicle to take Left, Straight or Right route respectively. Note, the volume 100 means we generate averagely 100 vehicles for a 10-minute simulation. We use five different traffic volumes $\{100, 200, 300, 400, 500\}$. For every volume, we ran simulations of three different traffic patterns to get the average results as the final result for each traffic volume.

Table 6.1: Parameters used for various traffic volumes and patterns. ($*$ Expected number of vehicles per 10 minutes.)

Parameter	Value
Traffic volumes$*$	100 / 200 / 300 / 400 / 500
p_V	0.03 / 0.06 / 0.08 / 0.11 / 0.14
p_L	0.20
p_S	0.60
p_R	0.20
Traffic patterns: random seeds	1: 12 / 2: 21 / 3: 66

Simulations were run using 0.05 s as the time step and a simulation is terminated when the simulated time reaches 10 minutes.

6.3.2 Simulation Results

The throughput performance of MICA is compared with that of the optimized traffic light algorithm in Section 4.3.1 as well as that of DICA.

For each vehicle, we obtained the performance measures introduced in Section 3.5.2 to show the performance of MICA. Also, we use the Jain's fairness index f [112, 113] to compare the fairness of each algorithm with respect to trip times of crossed vehicles. The Jain's fairness index takes values within $[0, 1]$. If all crossed vehicles have similar trip times, then f is close to 1. Otherwise, if all vehicles experience very different trip times, f is close to 0. If we let η_i be the trip time of vehicle c^i who crossed the intersection in a simulation run and N be the number of crossed vehicles, then we can compute the Jain's fairness index f as follows:

$$f(\vec{\eta} = [\eta_1, \eta_2, \cdots, \eta_N]) = \frac{(\sum_{i=1}^{i=N} \eta_i)^2}{N \sum_{i=1}^{i=N} \eta_i^2} \tag{6.15}$$

The performance comparisons between algorithms are shown in Figure 6.4. The result shows that MICA performs substantially better than others in both perfor-

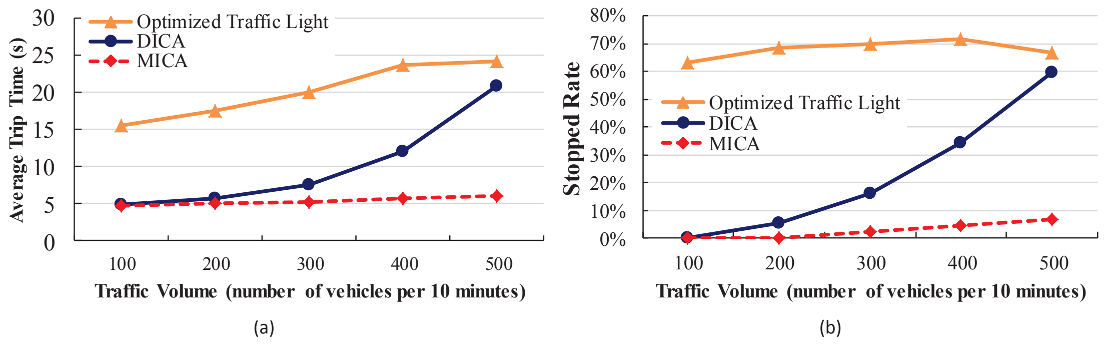


Figure 6.4: Crossed vehicles' performance of the optimized traffic light, DICA and MICA: (a) average trip time, (b) stopped rate.

Table 6.2: Jain's fairness index of the optimized traffic light, DICA and MICA.

Traffic volume (number of vehicles per 10 minutes)	100	200	300	400	500
Optimized Traffic Light	0.80	0.80	0.78	0.75	0.71
DICA	0.97	0.87	0.75	0.60	0.63
MICA	0.98	0.97	0.94	0.90	0.88

mance metrics (average trip time and stopped rate) in all traffic volume cases. In particular, one can see that the average trip time of MICA remains nearly constant while those of other algorithms increase as the traffic volume increases. Also, we can find that most vehicles cross an intersection without making a complete stop under MICA. Even in the highest traffic volume case when more than 60% of the crossed vehicles experienced a stop under the other two algorithms, MICA maintains less than 10% stopped rate for the same situation. Thus, the results show that the capacity of MICA in handling the intersection crossing traffic is much larger than others.

Table 6.2 shows the Jain's fairness index of the three algorithms for all traffic volumes. The index for MICA decreases slightly as the traffic volume increases but still shows much better performance than other algorithms.

Figure 6.5 plots the histograms of the trip times of all crossed vehicles for the optimized traffic light, DICA and MICA for the heaviest volume 500 and traffic pattern 1. Very few vehicles take about 10 seconds more than the most frequent trip time when the simulation employs MICA. DICA's trip times distribute over a much larger range and the optimized traffic light has more vehicles experienced larger trip times.

Figure 6.6 shows that the throughputs of the three algorithms are almost the same for all traffic volumes. We are using random generation of vehicles in the simulations, so the small fluctuations should due to the randomness and the results of light volumes are more influenced. The effective average trip time shares similar

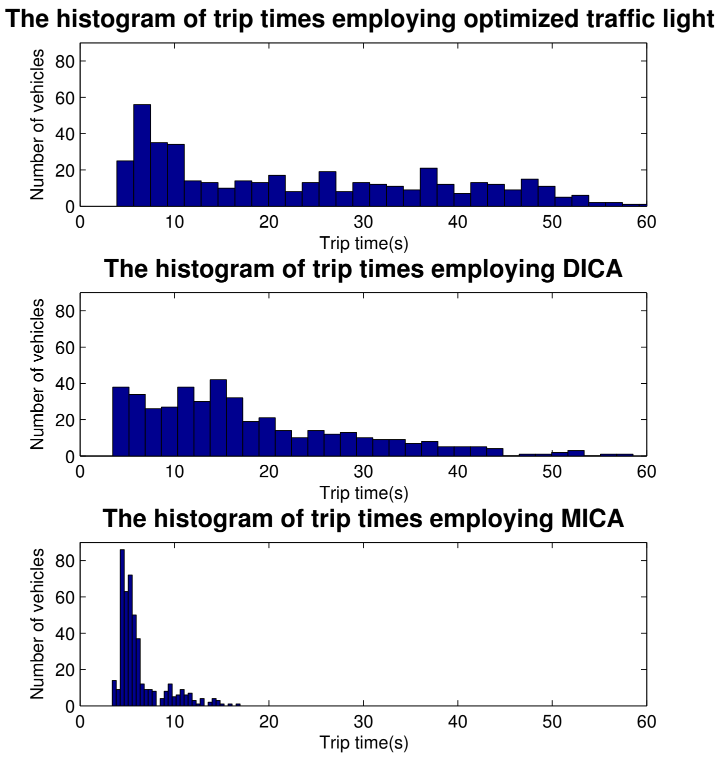
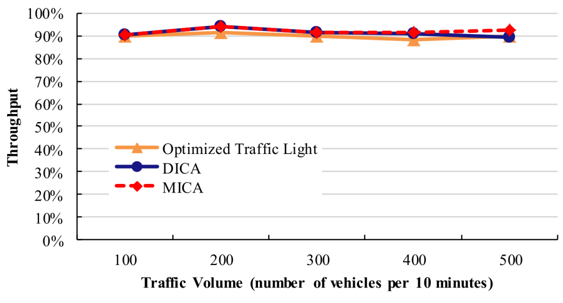


Figure 6.5: Histograms of trip times when employing the optimized traffic light, DICA and MICA for traffic volume 500, pattern 1.

trends with the average trip time in Figure 6.4 because the range of throughput values is small.



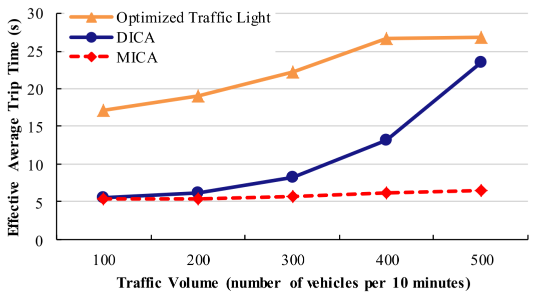


Figure 6.6: Overall performance comparison among the optimized traffic light, DICA and MICA: (a) throughput, (b) effective average trip time.

6.4 Summary

In this chapter, we use MIP to formulate a new head vehicle's optimal control problem. Besides the constraints from vehicles and local laws, constraints from conflicting vehicles and front vehicles are also carefully designed in order to ensure the safety of intersection crossing vehicles. Simulation results show that the proposed MICA achieves substantially higher throughput as well as fairness performance than other intersection control approaches, such as the optimized traffic light and DICA.

CHAPTER SEVEN

MICACO: MICA considering Communication Uncertainties

MICA works very well if we have perfect communication between the ICA and CAVs. However, in real applications, communication uncertainties which typically include packet delay and loss, are unavoidable. In this chapter, we present an approach based on MICA to handle packet delay and loss while ensuring safety and liveness. In this dissertation, packet delay means the amount of time a message needs to travel from a CAV/ICA to another ICA/CAV. And packet loss means some packets may fail to reach their destinations due to the blocking of buildings or message congestions, etc.

7.1 MICACO

7.1.1 New Interaction Mechanism

In order to handle these communication problems, we propose a simple yet effective high-level interaction mechanism based on MICA to ensure CAVs free from collisions under imperfect communication environment. In the new mechanism which we call MICACO (MICA considering COmmunication uncertaities), defaultly a head vehicle which is not confirmed by the ICA will start to decelerate with an acceleration rate that will make it stop at the enter line of the intersection region.

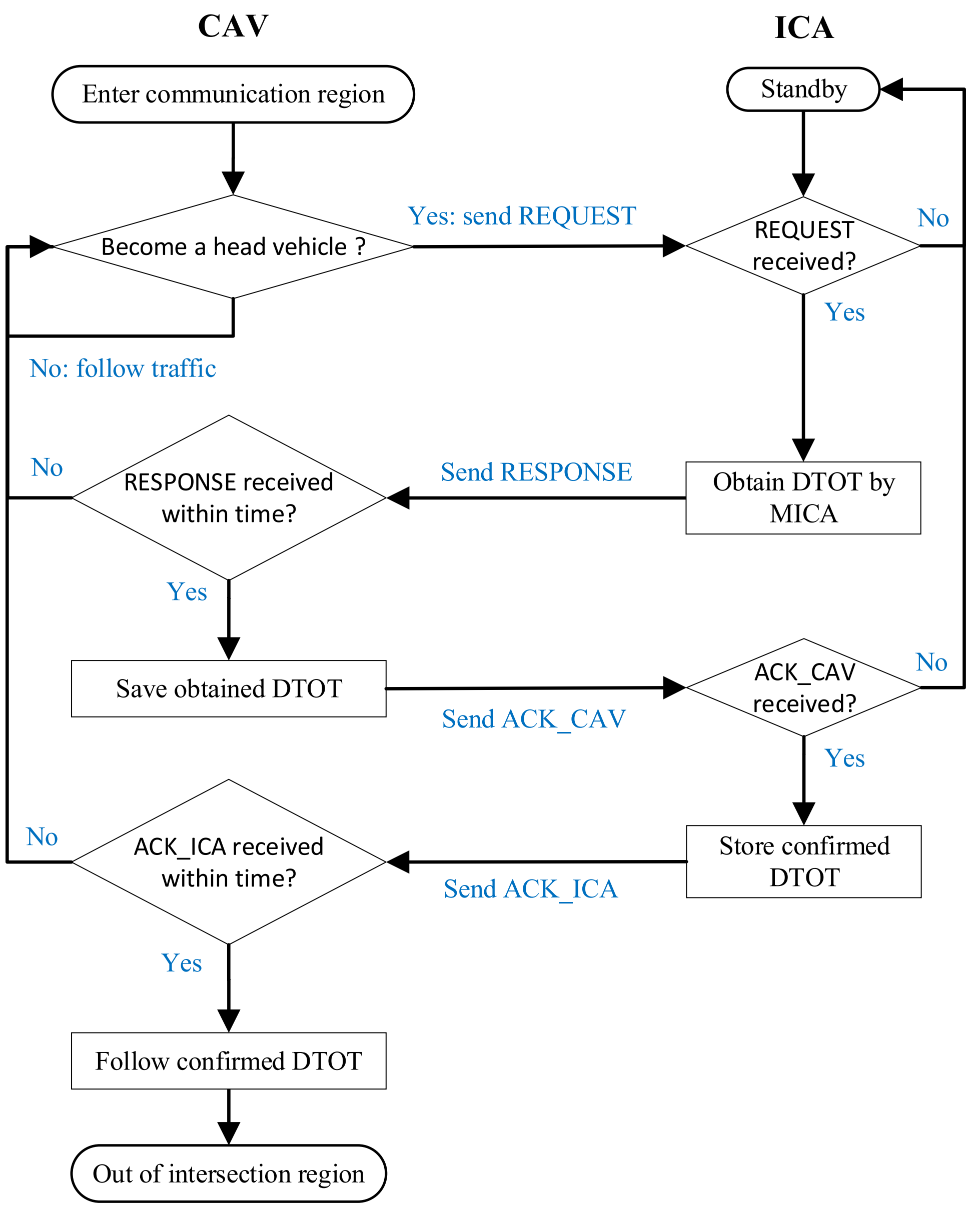


Figure 7.1: Interaction between a CAV and the ICA handling communication problems.

The main idea of MICACO is illustrated through the interaction between a CAV and the ICA in Figure 7.1. Compared with the interaction scheme in Figure 6.1, the new interaction diagram has more components and two more messages ACK_CAV and ACK_ICA to handle delayed and lost messages. Once a vehicle enters the communication region, it will check if it is the head vehicle on its lane. Same as previous interaction mechanism, if a CAV becomes the head vehicle on its lane, it will send a REQUEST message to the ICA to get a confirmed collision-free DTOT to follow. The REQUEST message contains the vehicle's VIN and other information for its crossing. Let's denote the empirical upper bound of the one-way trip time for a message to take between a vehicle and the ICA is h. If the REQUEST message is not delivered successfully to the ICA, the vehicle will decelerate for $2h$ time and realize the failure. Then the vehicle will generate a new REQUEST message and send it again. Otherwise, with the information in the message and other confirmed vehicles' DTOTs, the ICA computes the corresponding DTOT for this vehicle which avoids potential conflicts with all other confirmed vehicles using MICA. The obtained DTOT is sent back to the CAV in the RESPONSE message. Note here, the ICA temporarily saves the generated DTOT and will check against this DTOT when it wants to confirm another new vehicle. If the RESPONSE is received by the CAV within $2h$, the CAV will save the received DTOT and send ACK_CAV message to inform the ICA the reception of the RESPONSE. Otherwise, as we introduced above, the CAV will regenerate a REQUEST and send it. In the meantime, the CAV is still decelerating with an acceleration rate that can make it stop at the enter line. The introduction of ACK_CAV lets the ICA knows that the CAV receives the collision-free DTOT and also stores it. Then those space-times are reserved for the CAV indicating the CAV is confirmed by the ICA. Similarly, if the ACK_CAV is not received by the ICA, the ICA will simply remove the generated DTOT and go back to its Standby state which will be explained later. Lastly, the ICA sends ACK_ICA to indicate the reception of the message ACK_CAV. Then once the CAV receives ACK_ICA, it knows that its collision-free DTOT is confirmed by the ICA and plans itself to follow its DTOT. Note that this DTOT contains the trajectory after the CAV decelerates for $4h$. Even if the CAV receives the DTOT

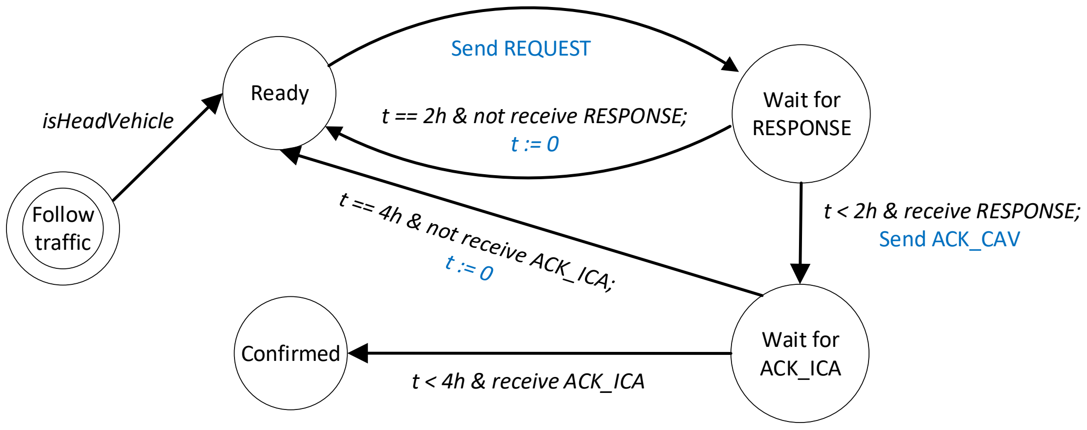


Figure 7.2: State machine of a CAV.

much earlier than $4h$, it will still decelerate and start to follow the DTOT at the time instant after $4h$. If the message ACK_ICA is not received by the CAV within $4h$ time or it is lost, the CAV will realize this at $4h$ and regenerate a REQUEST and send it to the ICA. We will explain the timer reset and other details for the ICA and CAV in their respective state machines in the following sections.

7.1.2 CAV State Machine

The state machine diagram of a CAV is shown in Figure 7.2. Several state transitions are shown in the figure to describe how vehicles will behave where there exist packet delay and loss. Same as before, a CAV will always follow traffic before it becomes a head vehicle. If the CAV is a head vehicle, it enters the state *Ready* which is a transient state that the CAV will send REQUEST and enter another state *Wait for RESPONSE*. In this state, the vehicle is waiting for the RESPONSE message from the ICA. If the message is not lost and the communication delay is within the upper bound (RESPONSE is received by the CAV within $2h$), the CAV sends ACK_CAV and enters the state *Wait for ACK_ICA*. Note here *not lost* means that the message should not be lost in either direction i.e. from the CAV to the ICA and from the ICA to the CAV. Otherwise, it goes back to state *Ready* and

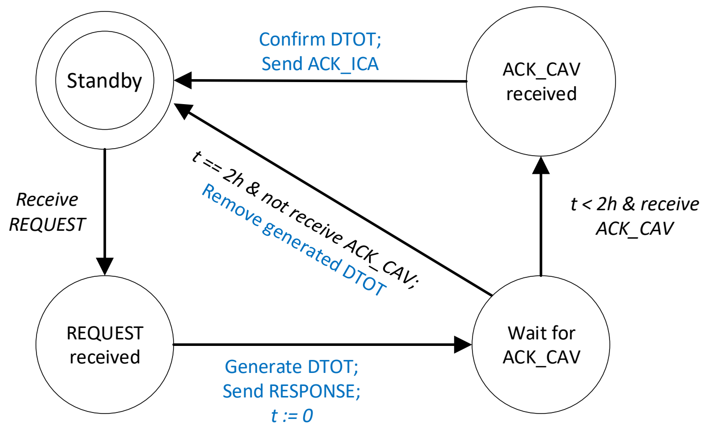


Figure 7.3: State machine of an ICA interaction instance.

sends an updated REQUEST. In the meantime, the timer is reset: $t := 0$. We assume that the communication delay is far larger than the processing time for the ICA to compute a DTOT. Thus the processing time is negligible. In the state *Wait for ACK_ICA*, the vehicle is waiting for the ACK_ICA message from the ICA. If it receives the ACK_ICA in time, it finally enters the state *Confirmed* indicating the completion of interaction with the ICA. Otherwise, it will go back to *Ready* state and restart everything.

7.1.3 ICA State Machine

The state machine diagram of the ICA is shown in Figure 7.3. When no REQUESTs are received, the ICA is in state *Standby*. The ICA will enter state *REQUEST received* once a REQUEST from a CAV arrives at the ICA. Then the ICA generates DTOT for the CAV and sends the RESPONSE back. The ICA starts to count time when it sends the RESPONSE. It is now in state *Wait for ACK_CAV* at which state two transitions may happen. If the message is not lost and the corresponding delay is within the bound i.e. the ICA receives ACK_CAV within $2h$, it enters the state *ACK_CAV received*. Otherwise, the ICA cannot receive the ACK_CAV message in time, it will remove the generated DTOT and become

Table 7.1: Parameters used for various traffic volumes and patterns. ($*$ Expected number of vehicles per 10 minutes.)

Parameter	Value
Traffic volumes$*$	100 / 200 / 300 / 400 / 500
p_V	0.03 / 0.06 / 0.08 / 0.11 / 0.14
p_L	0.20
p_S	0.60
p_R	0.20
Traffic patterns: random seeds	1: 12 / 2: 21 / 3: 66

Standby state again. In state *ACK_CAV received*, the ICA will confirm the collision-free DTOT and send message ACK_ICA to the CAV. Until this point, a complete interaction with a CAV is done.

Note that, Figure 7.3 illustrates the state machine for an ICA interaction instance which only deals with one CAV. If there is another vehicle or multiple vehicles send REQUESTs to the ICA at the same time, the ICA will generate more interaction instances to for each vehicle and destroy those extra instances once the interactions are completed. The ICA will only keep one available instance when there are no requests from CAVs.

7.2 Simulation

Simulations were implemented in the open-source traffic simulator SUMO [103] to show the performance of MICACO. SUMO is highly portable and very popular within the academic community for microscopic road traffic simulations to validate traffic control algorithms. The algorithms were programmed as Python applications which interact with SUMO through its API: TraCI. The mixed integer programming optimization problem was solved using Gurobi optimization solver [111].

7.2.1 Simulation Setup

We used an isolated four-way intersection with three entrance lanes and two exit lanes each way in our simulations. Most simulation setups in this chapter are similar to that in previous chapters. The maximum allowed speed v_{max} for all vehicles is 70 km/h. Before entering the communication region, vehicles are traveling with different constant speeds which are randomly chosen within the range from 40% to 100% of v_{max}. For simple, all vehicles have the same rectangular shape and size (5 meters long and 1.8 meters wide). We set $a_{max} = 2\ m/s^2$ and $a_{min} = -4.5\ m/s^2$, respectively. However, note that MICACO also applies to vehicles with different shapes, sizes and acceleration rates. The distance from the enter line of the communication region to that of the intersection region is set as 50 m.

Vehicles were spawned randomly on each entrance lane with a randomly assigned intersection route. We used several different random seeds to generate different traffic patterns and make simulations reproducible. In Table 7.1, the parameters used for generating various patterns and volumes are summarized in which pL, pS and pR are the probabilities for a new generated vehicle to take Left, Straight or Right route respectively. Note, the volume 100 means we generate averagely 100 vehicles for a 10-minute simulation. For every volume, we ran simulations of three different traffic patterns to get the average result as the final result for each traffic volume. Simulations were run using 0.05 s as the time step and a simulation is terminated when the simulated time reaches 10 minutes. The packet loss rate used was 5% and the one-way delay mean and standard deviation used were $0.1s$ and $25\% * 0.1s$. Simulations were run using 0.05 s as the time step and a simulation is terminated when the simulated time reaches 10 minutes.

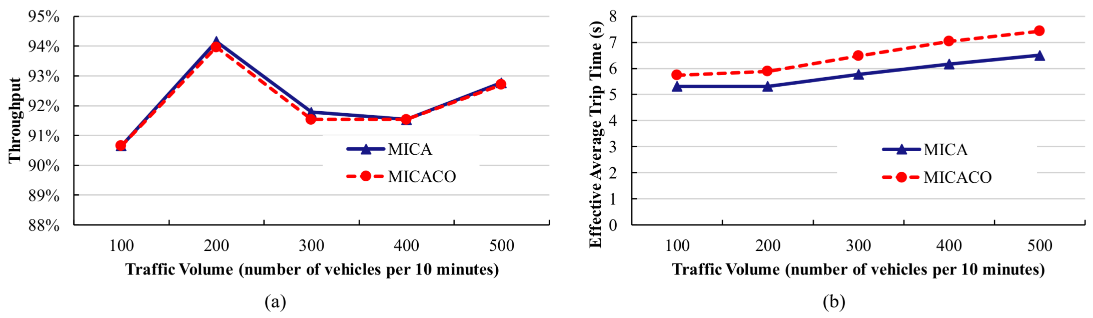


Figure 7.4: Traffic control performance comparison between MICA and MICACO (a) throughput, (b) effective average trip time.

7.2.2 Simulation Results

The traffic control performance of MICACO is compared with that of MICA as well as that of the optimized traffic light algorithm introduced in Section 4.3.1. For each vehicle, we obtained the performance measures introduced in Section 3.5.2 to show the performance of MICACO. Also, we use the Jain's fairness index f introduced in Section 6.3.2 to compare the fairness of each algorithm with respect to trip times of crossed vehicles.

Comparison with MICA

We compare the overall traffic control performance of MICACO with that of MICA to show the effect of considering communication uncertainties on final performance. As shown in Fig. 7.4, with imperfect communication, the throughput keeps almost the same and the effective average trip time increases for all volume situations. Both metrics show similar trends for MICA and MICACO. Since we generated vehicles randomly in the simulations, part of the small fluctuations of throughput should due to the randomness and the results of light volumes are more influenced. Based on the definition of the effective average trip time we know that the average trip time also increases for all volumes situations. However, the percentage of performance drop is relatively small as the maximum decrease percentage of the effective average trip time is about 13%.

Comparison with the Optimized Traffic Light Algorithm

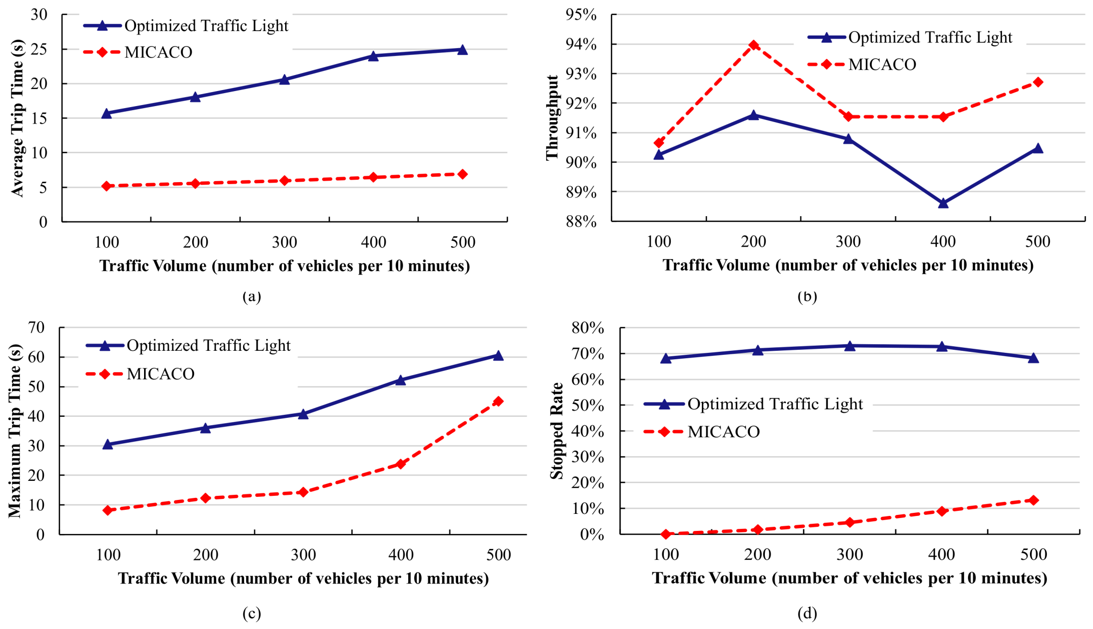


Figure 7.5: Comparison of traffic control performance between the optimized traffic light algorithm and MICACO: (a) average trip time, (b) stopped rate.

The performance comparisons between the optimized traffic light algorithm and MICACO are shown in Figure 7.5. The results show that even though MICACO's performance drops a bit from MICA's performance to handle packet delay and loss, it still performs substantially better than the optimized traffic light algorithm for all performance metrics (average trip time, throughput, maximum trip time and stopped rate) in all traffic volume cases. This is because even though packet delay and loss bring some delays for an approaching head vehicle, the final DTOT computed through MIP is still the optimal trajectory for the vehicle to follow. Compared with the optimized traffic light algorithm, there are not unnecessary waiting times for vehicles under the control of MICACO. In particular, we can see that the average

Table 7.2: Jain's fairness index of the optimized traffic light algorithm and MICACO.

Traffic volume (number of vehicles per 10 minutes)	100	200	300	400	500
Optimized Traffic Light	0.80	0.80	0.78	0.75	0.71
MICACO	0.98	0.95	0.91	0.87	0.86

trip time of MICACO remains nearly constant while that of the optimized traffic light algorithm increases as the traffic volume increases. And in the worst case, the optimized traffic light algorithm takes a vehicle averagely 3 times longer than MICACO to cross the intersection. One can find that although there are small fluctuations of throughput due to randomness, MICACO achieved slightly better throughput than the optimized traffic light across all traffic volume cases. In terms of maximum trip time, both algorithms show similar trends that it took a vehicle longer and longer time to cross in the worst case when there were more and more vehicles. When more vehicles enter the communication region, it is more likely that a vehicle will experience multiple delays and packet losses which result in longer waiting time before crossing that contributes to the longer total trip time. Lastly, Figure 7.5 (d) shows that most vehicles crossed the intersection without making a complete stop under MICACO. Even in the highest traffic volume case, when more than 60% of the crossed vehicles experienced a stop under the optimized traffic light algorithm, MICACO maintains less than 15% stopped rate for the same volume case. Thus, the results show that the capacity of MICACO in handling the intersection crossing traffic is much larger than the traditional traffic light.

Table 7.2 shows the Jain's fairness index of the two algorithms for all traffic volumes. The fariness index for both algorithms decreases slightly as the traffic volume increases but MICACO shows much better performance than the other algorithm. For light traffic volume cases, the fairness index of MICACO is very close to 1 which means that vehicles were able to cross the intersection almost unhindered leading to very similar trip times.

7.3 Summary

Based on MICA, this chapter proposed MICACO which ensures safety and obtains the optimal trajectory for each vehicle while considering imperfect communication like packet delay and loss. MICACO uses simple interaction mechanism and two more message types ACK_CAV and ACK_ICA to ensure the successful delivery of messages between a CAV and an ICA. Extensive simulation results show that the proposed MICACO achieves substantially higher throughput as well as fairness performance than the optimized traffic light algorithm although it performs a bit worse than MICA due to the handling of communication uncertainties.

CHAPTER EIGHT

Conclusion and Future Work

This dissertation has presented several control algorithms for connected and autonomous intersection traffic. These algorithms ranging from basic DICA, reactive DICA for emergency vehicles' expedited crossings, to MICA which gives every vehicle an optimal trajectory and MICACO which handles communication uncertainties. All algorithms were validated through extensive simulations. This chapter summarizes the dissertation briefly and gives potential future work.

8.1 Conclusion

Chapter 1 made a brief introduction for the dissertation and Chapter 2 explored existing researches in related fields. In Chapter 3, we developed an intelligent intersection control algorithm DICA employing the concept DTOT. The chapter introduced the concept of DTOT by which ICA is able to manage limited intersection space at a more accurate and efficient way. Theoretical analysis shows that DICA is free from deadlocks and starvation problems. Simulation results show that our algorithm achieves less Effective Average Trip Time compared with the Concurrent intersection control algorithm.

In Chapter 4, we analyzed the computational complexity of the original DICA and enhanced the algorithm so that it can have better overall computational efficiency. The enhancement was done through several computational techniques like determining conflicting spaces offline, employing the bisection method in time-conflict checking, etc. Simulation results show that the computational efficiency of the algorithm is improved significantly after the enhancement and the properties of starvation free and safety are guaranteed. We also validated that the overall throughput performance of our enhanced DICA is better than that of an optimized traffic light control mechanism in the case when the traffic is not congested.

In Chapter 5, we have shown that the DICA algorithm can be augmented to allow emergency vehicles to cross intersections faster. A genetic algorithm based approach is proposed as part of the augmented algorithm, called R-DICA, to optimize the sequence of vehicles which gives the emergency vehicle the highest priority and keeps the influence on other vehicles' travel times as minimum as possible. The R-DICA operates the same way as DICA if there is no EV inside the communication region and optimizes vehicle-passing sequence if an EV enters the communication region. A reactive traffic light and DICA algorithms are also implemented for simulation and their results are compared with R-DICA to evaluate the performance of R-DICA. Simulation results show that R-DICA is effective to reduce travel times of EVs and has better performance than the reactive traffic light for normal vehicles. We conclude that the performance of normal vehicles is not noticeably affected based on the simulation results of DICA and R-DICA.

In Chapter 6, we use MIP to formulate a new head vehicle's optimal control problem. Besides the constraints from vehicles and local laws, constraints from conflicting vehicles and front vehicles are also properly modeled. Great improvements from DICA are shown in simulation results which implies that the algorithm MICA is able to reduce intersection congestions effectively.

In Chapter 7, we propose MICACO which ensures safety and obtains the optimal trajectory for each vehicle while considering imperfect communications like packet delay and loss. MICACO has more interaction rules between a CAV and an ICA, and two more messages ACK_CAV and ACK_ICA to handle delayed and lost messages.

Extensive simulation results show that the proposed MICACO achieves substantially higher throughput as well as fairness performance than the optimized traffic light algorithm although it performs a bit worse than MICA due to the handling of communication uncertainties.

8.2 Future Work

In the future, assumptions like the accurate prediction of DTOT can be relaxed and methods to deal with car failures will be studied to make the algorithm more applicable to real situations.

In addition to giving priority to special vehicles e.g. emergency vehicles by forming optimal sequence like R-DICA, R-DICA can be enhanced to allow special vehicles' faster crossings through efficient usage of intersection space. For example, ICA may modify both occupancies' positions and times of a vehicle's DTOT in order to form a passage for special vehicles.

We will study algorithms based on MICACO on a network of intersections to have a global optimal performance on a city level. Also, we will work to integrate the grouping strategy used in traffic flow based intersection control algorithms into our MICACO to achieve better performances in more congested situations. MICACO can be generalized to work with mixed traffic where autonomous vehicles and human-driven vehicles coexist. A more interesting and difficult problem could be including pedestrians in the intersection traffic. The validation of MICACO related algorithms will use more professional communication simulation softwares.

8.3 Publications

1. Qiang Lu and Kyoung-Dae Kim, "Autonomous and connected intersection crossing traffic management using discrete-time occupancies trajectory." Applied Intelligence, 2018, DOI: 10.1007/s10489-018-1357-1.
2. Qiang Lu and Kyoung-Dae Kim. "A Mixed Integer Programming Approach for Autonomous and Connected Intersection Crossing Traffic Control." IEEE 88th Vehicular Technology Conference: Chicago, 2018.
3. Qiang Lu and Kyoung-Dae Kim, "A Genetic Algorithm Approach for Expedited Crossing of Emergency Vehicles in Connected and Autonomous Intersection Traffic." Journal of Advanced Transportation, Article ID 7318917 (2017).
4. Qiang Lu and Kyoung-Dae Kim. "Intelligent Intersection Management of Autonomous Traffic Using Discrete-Time Occupancies Trajectory." Journal ofTraffic and Logistics Engineering Vol 4.1 (2016).

REFERENCES

[1] K.-D. Kim and P. R. Kumar, "An mpc-based approach to provable system-wide safety and liveness of autonomous ground traffic," *IEEE Transactions on Automatic Control*, vol. 59, no. 12, pp. 3341–3356, 2014.

[2] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, and Y. Wang, "Review of road traffic control strategies," *Proceedings of the IEEE*, vol. 91, no. 12, pp. 2043–2067, 2003.

[3] R. Horowitz and P. Varaiya, "Control design of an automated highway system," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 913–925, 2000.

[4] DARPA, "The darpa urban challenge," 2007. [Online]. Available: http://archive.darpa.mil/grandchallenge/

[5] K. Bengler, K. Dietmayer, B. Farber, M. Maurer, C. Stiller, and H. Winner, "Three decades of driver assistance systems: Review and future perspectives," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 4, pp. 6–22, 2014.

[6] P. Tientrakool, Y.-C. Ho, and N. F. Maxemchuk, "Highway capacity benefits from using vehicle-to-vehicle communication and sensors for collision avoidance," in *Vehicular Technology Conference (VTC Fall), 2011 IEEE.* IEEE, 2011, pp. 1–5.

[7] L. W. Chen and C. C. Chang, "Cooperative traffic control with green wave coordination for multiple intersections based on the internet of vehicles," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. PP, no. 99, pp. 1–15, 2016.

[8] B. Zhou, J. Cao, X. Zeng, and H. Wu, "Adaptive traffic light control in wireless sensor network-based intelligent transportation system," in *Vehicular technology conference fall (VTC 2010-Fall), 2010 IEEE 72nd.* IEEE, 2010, pp. 1–5.

[9] M. Wiering, "Multi-agent reinforcement learning for traffic light control," in *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*, 2000, pp. 1151–1158.

[10] A. Konar, I. G. Chakraborty, S. J. Singh, L. C. Jain, and A. K. Nagar, "A deterministic improved q-learning for path planning of a mobile robot," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 5, pp. 1141–1153, Sept 2013.

[11] T.-H. Li and S.-J. Chang, "Autonomous fuzzy parking control of a car-like mobile robot," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 33, no. 4, pp. 451–465, 2003.

[12] A. Mammeri, D. Zhou, and A. Boukerche, "Animal-vehicle collision mitigation system for automated vehicles," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 9, pp. 1287–1299, Sept 2016.

[13] A. Colombo and D. Del Vecchio, "Efficient algorithms for collision avoidance at intersections," in *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control.* ACM, 2012, pp. 145–154.

[14] S. Le Vine, A. Zolfaghari, and J. Polak, "Autonomous cars: The tension between occupant experience and intersection capacity," *Transportation Research Part C: Emerging Technologies*, vol. 52, pp. 1–14, 2015.

[15] E. Onieva, U. Hernández-Jayo, E. Osaba, A. Perallos, and X. Zhang, "A multi-objective evolutionary algorithm for the tuning of fuzzy rule bases for

uncoordinated intersections in autonomous driving," *Information Sciences*, vol. 321, pp. 14–30, 2015.

[16] A. Puri, K. Valavanis, and M. Kontitsis, "Statistical profile generation for traffic monitoring using real-time uav based video data," in *Control & Automation, 2007. MED'07. Mediterranean Conference on.* IEEE, 2007, pp. 1–6.

[17] ——, "Generating traffic statistical profiles using unmanned helicopter-based video data," in *Robotics and Automation, 2007 IEEE International Conference on.* IEEE, 2007, pp. 870–876.

[18] A. Puri, "Statistical profile generation of real-time uav-based traffic data," *Ph.D. Dissertation*, 2008.

[19] A. P. Chouhan and G. Banda, "Autonomous intersection management: A heuristic approach," *IEEE Access*, vol. 6, pp. 53 287–53 295, 2018.

[20] M. Khayatian, M. Mehrabian, and A. Shrivastava, "Rim : Robust intersection management for connected autonomous vehicles," in *39th IEEE Real-Time Systems Symposium*, 2018.

[21] J. Lee and B. Park, "Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 81–90, 2012.

[22] J. Wu, A. Abbas-Turki, and A. El Moudni, "Cooperative driving: an ant colony system for autonomous intersection management," *Applied Intelligence*, vol. 37, no. 2, pp. 207–222, 2012.

[23] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

[24] F. Yan, M. Dridi, and A. El Moudni, "An autonomous vehicle sequencing problem at intersections: A genetic algorithm approach," *International Journal of Applied Mathematics and Computer Science*, vol. 23, no. 1, pp. 183–200, 2013.

[25] F. Zhu and S. V. Ukkusuri, "A linear programming formulation for autonomous intersection control within a dynamic traffic assignment and connected vehicle environment," *Transportation Research Part C: Emerging Technologies*, vol. 55, pp. 363–378, 2015.

[26] A. A. Malikopoulos and C. G. Cassandras, "Decentralized optimal control for connected and automated vehicles at an intersection," *arXiv preprint arXiv:1602.03786*, 2016.

[27] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *Journal of artificial intelligence research*, vol. 31, pp. 591–656, 2008.

[28] ——, "Human-usable and emergency vehicle-aware control policies for autonomous intersection management," in *Fourth International Workshop on Agents in Traffic and Transportation (ATT), Hakodate, Japan*, 2006.

[29] D. Carlino, S. D. Boyles, and P. Stone, "Auction-based autonomous intersection management," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, 2013, pp. 529–534.

[30] M. Vasirani and S. Ossowski, "A market-inspired approach for intersection management in urban road traffic networks," *Journal of Artificial Intelligence Research*, vol. 43, pp. 621–659, 2012.

[31] M. W. Levin, H. Fritz, and S. D. Boyles, "On optimizing reservation-based intersection controls," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 3, pp. 505–515, 2017.

[32] M. W. Levin and D. Rey, "Conflict-point formulation of intersection control for autonomous vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 85, pp. 528–547, 2017.

[33] S. R. Azimi, G. Bhatia, R. Rajkumar, and P. Mudalige, "Vehicular networks for collision avoidance at intersections," in *SAE 2011 world congress and exhibition*, 2011.

[34] R. Azimi, G. Bhatia, R. Rajkumar, and P. Mudalige, "Intersection management using vehicular networks," SAE Technical Paper, Tech. Rep., 2012.

[35] S. Azimi, G. Bhatia, R. Rajkumar, and P. Mudalige, "Reliable intersection protocols using vehicular networks," in *Cyber-Physical Systems (ICCPS), 2013 ACM/IEEE International Conference on.* IEEE, 2013, pp. 1–10.

[36] R. Azimi, G. Bhatia, R. R. Rajkumar, and P. Mudalige, "Stip: Spatio-temporal intersection protocols for autonomous vehicles," in *ICCPS'14: ACM/IEEE 5th International Conference on Cyber-Physical Systems (with CPS Week 2014).* IEEE Computer Society, 2014, pp. 1–12.

[37] B. Liu, Q. Shi, Z. Song, and A. El Kamel, "Trajectory planning for autonomous intersection management of connected vehicles," *Simulation Modelling Practice and Theory*, vol. 90, pp. 16–30, 2019.

[38] P. Dai, K. Liu, Q. Zhuge, E. H.-M. Sha, V. C. S. Lee, and S. H. Son, "Quality-of-experience-oriented autonomous intersection control in vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 7, pp. 1956–1967, 2016.

[39] M. A. S. Kamal, J.-i. Imura, T. Hayakawa, A. Ohata, and K. Aihara, "A vehicle-intersection coordination scheme for smooth flows of traffic without using traffic lights," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 3, pp. 1136–1147, 2015.

[40] E. R. Müller, R. C. Carlson, and W. Kraus, "Time optimal scheduling of automated vehicle arrivals at urban intersections," in *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*. IEEE, 2016, pp. 1174–1179.

[41] E. R. Müller, B. Wahlberg, and R. C. Carlson, "Optimal motion planning for automated vehicles with scheduled arrivals at intersections," in *2018 European Control Conference (ECC)*. IEEE, 2018, pp. 1672–1678.

[42] B. Zheng, C.-W. Lin, H. Liang, S. Shiraishi, W. Li, and Q. Zhu, "Delay-aware design, analysis and verification of intelligent intersection management," in *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2017, pp. 1–8.

[43] B. Zheng, M. O. Sayin, C.-W. Lin, S. Shiraishi, and Q. Zhu, "Timing and security analysis of vanet-based intelligent transportation systems," in *Computer-Aided Design (ICCAD), 2017 IEEE/ACM International Conference on*. IEEE, 2017, pp. 984–991.

[44] X. Qian, J. Gregoire, F. Moutarde, and A. De La Fortelle, "Priority-based coordination of autonomous and legacy vehicles at intersection," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2014, pp. 1166–1171.

[45] Q. Lu and K.-D. Kim, “Intelligent intersection management of autonomous traffic using discrete-time occupancies trajectory,” *Journal of Traffic and Logistics Engineering Vol*, vol. 4, no. 1, pp. 1–6, 2016.

[46] ——, “A genetic algorithm approach for expedited crossing of emergency vehicles in connected and autonomous intersection traffic,” *Journal of Advanced Transportation*, 2017.

[47] ——, “Autonomous and connected intersection crossing traffic management using discrete-time occupancies trajectory,” *Applied Intelligence*, 2018.

[48] ——, “A mixed integer programming approach for autonomous and connected intersection crossing traffic control,” in *IEEE 88th Vehicular Technology Conference*. IEEE, 2018.

[49] P. Varaiya, “Smart cars on smart roads: problems of control,” *IEEE Transactions on automatic control*, vol. 38, no. 2, pp. 195–207, 1993.

[50] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, “Predictive active steering control for autonomous vehicle systems,” *IEEE Transactions on control systems technology*, vol. 15, no. 3, pp. 566–580, 2007.

[51] R. Reghelin and L. V. R. de Arruda, “Using a centralized controller to optimize the traffic of intelligent vehicles in a single lane highway provided with a suicide lane,” in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*. IEEE, 2012, pp. 1049–1054.

[52] J. Ruan, M. Fu, Y. Li, and J. Huang, “Study on throttle control of intelligent vehicle longitudinal motion,” in *Vehicular Electronics and Safety, 2005. IEEE International Conference on*. IEEE, 2005, pp. 176–181.

[53] J. E. Naranjo, J. Reviejo, C. González, R. García, and T. de Pedro, "A throttle and brake fuzzy controller: towards the automatic car," in *International Conference on Computer Aided Systems Theory*. Springer, 2003, pp. 291–301.

[54] R. Rajamani, H.-S. Tan, B. K. Law, and W.-B. Zhang, "Demonstration of integrated longitudinal and lateral control for the operation of automated vehicles in platoons," *IEEE Transactions on Control Systems Technology*, vol. 8, no. 4, pp. 695–708, 2000.

[55] H. Kowshik, D. Caveney, and P. Kumar, "Provable systemwide safety in intelligent intersections," *IEEE transactions on vehicular technology*, vol. 60, no. 3, pp. 804–818, 2011.

[56] K.-D. Kim, "Collision free autonomous ground traffic: A model predictive control approach," in *Cyber-Physical Systems (ICCPS), 2013 ACM/IEEE International Conference on*. IEEE, 2013, pp. 51–60.

[57] N. M. Enache, M. Netto, S. Mammar, and B. Lusetti, "Driver steering assistance for lane departure avoidance," *Control engineering practice*, vol. 17, no. 6, pp. 642–651, 2009.

[58] P. G. Gipps, "A model for the structure of lane-changing decisions," *Transportation Research Part B: Methodological*, vol. 20, no. 5, pp. 403–414, 1986.

[59] J. Garcia-Nieto, A. C. Olivera, and E. Alba, "Optimal cycle program of traffic lights with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 6, pp. 823–839, 2013.

[60] O. Younis and N. Moayeri, "Employing cyber-physical systems: Dynamic traffic light control at road intersections," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2286–2296, 2017.

[61] J. L. Fleck, C. G. Cassandras, and Y. Geng, "Adaptive quasi-dynamic traffic light control," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 3, pp. 830–842, 2016.

[62] X. Meng and C. G. Cassandras, "Optimal control of autonomous vehicles for non-stop signalized intersection crossing," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 6988–6993.

[63] K. Dresner and P. Stone, "Multiagent traffic management: An improved intersection control mechanism," in *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*. ACM, 2005, pp. 471–477.

[64] M. Hausknecht, T.-C. Au, and P. Stone, "Autonomous intersection management: Multi-intersection optimization," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 4581–4586.

[65] J. Lee, B. B. Park, K. Malakorn, and J. J. So, "Sustainability assessments of cooperative vehicle intersection control at an urban corridor," *Transportation Research Part C: Emerging Technologies*, vol. 32, pp. 193–206, 2013.

[66] I. H. Zohdy, R. K. Kamalanathsharma, and H. Rakha, "Intersection management for autonomous vehicles using icacc," in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*. IEEE, 2012, pp. 1109–1114.

[67] J. Gregoire, S. Bonnabel, and A. de La Fortelle, "Optimal cooperative motion planning for vehicles at intersections," *arXiv preprint arXiv:1310.7729*, 2013.

[68] S. A. Fayazi, A. Vahidi, and A. Luckow, “Optimal scheduling of autonomous vehicle arrivals at intelligent intersections via milp,” in *American Control Conference (ACC), 2017*. IEEE, 2017, pp. 4920–4925.

[69] S. A. Fayazi and A. Vahidi, “Mixed-integer linear programming for optimal scheduling of autonomous vehicle intersection crossing,” *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 3, pp. 287–299, 2018.

[70] D. Miculescu and S. Karaman, “Polling-systems-based autonomous vehicle coordination in traffic intersections with no traffic signals,” *arXiv preprint arXiv:1607.07896*, 2016.

[71] R. Azimi, G. Bhatia, R. Rajkumar, and P. Mudalige, “V2v-intersection management at roundabouts,” *SAE International Journal of Passenger Cars-Mechanical Systems*, vol. 6, no. 2013-01-0722, pp. 681–690, 2013.

[72] W. Wu, J. Zhang, A. Luo, and J. Cao, “Distributed mutual exclusion algorithms for intersection traffic control,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 1, pp. 65–74, 2015.

[73] Y. J. Zhang, A. A. Malikopoulos, and C. G. Cassandras, “Optimal control and coordination of connected and automated vehicles at urban traffic intersections,” in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 6227–6232.

[74] Y. Zhang, C. G. Cassandras, and A. A. Malikopoulos, “Optimal control of connected automated vehicles at urban traffic intersections: A feasibility enforcement analysis,” in *American Control Conference (ACC), 2017*. IEEE, 2017, pp. 3548–3553.

[75] A. Parker and G. Nitschke, "How to best automate intersection management," in *Evolutionary Computation (CEC), 2017 IEEE Congress on.* IEEE, 2017, pp. 1247–1254.

[76] F. J. Martinez, C.-K. Toh, J.-C. Cano, C. T. Calafate, and P. Manzoni, "Emergency services in future intelligent transportation systems based on vehicular communication networks," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 2, pp. 6–20, 2010.

[77] E. Oliveira and N. Duarte, "Making way for emergency vehicles," in *Proceedings of the 2005 European Simulation and Modelling Conference*, vol. 128. Citeseer, 2005, p. 135.

[78] W. Viriyasitavat and O. K. Tonguz, "Priority management of emergency vehicles at intersections using self-organized traffic control," in *Vehicular Technology Conference (VTC Fall), 2012 IEEE.* IEEE, 2012, pp. 1–4.

[79] E. Andert, M. Khayatian, and A. Shrivastava, "Crossroads: Time-sensitive autonomous intersection management technique," in *Proceedings of the 54th Annual Design Automation Conference 2017.* ACM, 2017, p. 50.

[80] R. Zhang, F. Schmutz, K. Gerard, A. Pomini, L. Basseto, S. B. Hassen, A. Ishikawa, I. Ozgunes, and O. Tonguz, "Virtual traffic lights: System design and implementation," in *IEEE 88th Vehicular Technology Conference.* IEEE, 2018.

[81] K. M. Dresner and P. Stone, "Sharing the road: Autonomous vehicles meet human drivers." in *IJCAI*, vol. 7, 2007, pp. 1263–1268.

[82] R. Verma and D. Del Vecchio, "Semiautonomous multivehicle safety," *IEEE Robotics & Automation Magazine*, vol. 18, no. 3, pp. 44–54, 2011.

[83] L. C. Bento, R. Parafita, S. Santos, and U. Nunes, “Intelligent traffic management at intersections: Legacy mode for vehicles not equipped with v2v and v2i communications,” in *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on.* IEEE, 2013, pp. 726–731.

[84] S. I. Guler, M. Menendez, and L. Meier, “Using connected vehicle technology to improve the efficiency of intersections,” *Transportation Research Part C: Emerging Technologies*, vol. 46, pp. 121–131, 2014.

[85] H. Jiang, Y. Zhang, Y. Chen, C. Zhao, and J. Tan, “Power-traffic coordinated operation for bi-peak shaving and bi-ramp smoothing–a hierarchical data-driven approach,” *Applied energy*, vol. 229, pp. 756–766, 2018.

[86] J. Hao, X. Dai, A. Stroder, J. J. Zhang, B. Davidson, M. Mahoor, and N. McClure, “Prediction of a bed-exit motion: Multi-modal sensing approach and incorporation of biomechanical knowledge,” in *Signals, Systems and Computers, 2014 48th Asilomar Conference on.* IEEE, 2014, pp. 1747–1751.

[87] Y. Gu, H. Jiang, Y. Zhang, and D. W. Gao, “Statistical scheduling of economic dispatch and energy reserves of hybrid power systems with high renewable energy penetration,” in *2014 48th Asilomar Conference on Signals, Systems and Computers.* IEEE, 2014, pp. 530–534.

[88] H. Jiang, J. J. Zhang, D. W. Gao, Y. Zhang, and E. Muljadi, “Synchrophasor based auxiliary controller to enhance power system transient voltage stability in a high penetration renewable energy scenario,” in *2014 IEEE Symposium Power Electronics and Machines for Wind and Water Applications (PEMWA).* IEEE, 2014, pp. 1–7.

[89] J. Hao, X. Dai, Y. Zhang, J. Zhang, and W. Gao, "Distribution locational real-time pricing based smart building control and management," in *2016 North American Power Symposium (NAPS)*, Sept 2016, pp. 1–6.

[90] J. Hao, Y. Gu, Y. Zhang, J. J. Zhang, and D. W. Gao, "Locational marginal pricing in the campus power system at the power distribution level," in *2016 IEEE Power and Energy Society General Meeting (PESGM)*, July 2016, pp. 1–5.

[91] H. Jiang, J. J. Zhang, A. Hebb, and M. H. Mahoor, "Time-frequency analysis of brain electrical signals for behvior recognition in patients with parkinson's disease," in *2013 Asilomar Conference on Signals, Systems and Computers*. IEEE, 2013, pp. 1843–1847.

[92] H. Jiang, Y. Li, Y. Zhang, J. J. Zhang, D. W. Gao, E. Muljadi, and Y. Gu, "Big data-based approach to detect, locate, and enhance the stability of an unplanned microgrid islanding," *Journal of Energy Engineering*, vol. 143, no. 5, p. 04017045, 2017.

[93] H. Jiang, F. Ding, and Y. Zhang, "Short-term load forecasting based automatic distribution network reconfiguration: Preprint," National Renewable Energy Laboratory (NREL), Golden, CO (United States), Tech. Rep., 2017.

[94] J. J. Zhang, D. W. Gao, Y. Zhang, X. Wang, X. Zhao, D. Duan, X. Dai, J. Hao, and F. Wang, "Social energy: mining energy from the society," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 3, pp. 466–482, 2017.

[95] F. He, J. Hao, X. Dai, J. J. Zhang, J. Wei, and Y. Zhang, "Composite socio-technical systems: A method for social energy systems," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 2017, pp. 426–431.

[96] Y. Zhang, J. Zhang, W. Gao, X. Zheng, L. Yang, J. Hao, and X. Dai, "Distributed electrical energy systems: Needs, concepts, approaches and vision," *Acta Automatica Sinica*, vol. 43, no. 9, 9 2017.

[97] K. Dresner and P. Stone, "Learning policy selection for autonomous intersection management," in *The AAMAS 2007 workshop on adaptive and learning agents (ALAg 2007)*, 2007, pp. 34–39.

[98] M. Vasirani and S. Ossowski, "Learning and coordination for autonomous intersection control," *Applied Artificial Intelligence*, vol. 25, no. 3, pp. 193–216, 2011.

[99] L. Li, Y. Lv, and F.-Y. Wang, "Traffic signal timing via deep reinforcement learning," *IEEE/CAA Journal of Automatica Sinica*, vol. 3, no. 3, pp. 247–254, 2016.

[100] E. van der Pol, "Deep reinforcement learning for coordination in traffic light control," *Master's thesis, University of Amsterdam*, 2016.

[101] W. Genders and S. Razavi, "Using a deep reinforcement learning agent for traffic signal control," *arXiv preprint arXiv:1611.01142*, 2016.

[102] R. Zhang, A. Ishikawa, W. Wang, B. Striner, and O. Tonguz, "Partially observable reinforcement learning for intelligent transportation systems," *arXiv preprint arXiv:1807.01628*, 2018.

[103] K. Daniel, E. Jakob, B. Michael, and B. Laura, "Recent development and applications of sumo - simulation of urban mobility," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, December 2012.

[104] D. Cheng, Z. Z. Tian, and C. J. Messer, "Development of an improved cycle length model over the highway capacity manual 2000 quick estimation

method," *Journal of transportation engineering*, vol. 131, no. 12, pp. 890–897, 2005.

[105] T. R. Board, *Highway Capacity Manual*, National Academy of Sciences, Transportation Research Board, Washington, DC, 2000.

[106] X. Chen, L. Li, and Y. Zhang, "A markov model for headway/spacing distribution of road traffic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 4, pp. 773–785, 2010.

[107] T. Murata, H. Ishibuchi, and H. Tanaka, "Genetic algorithms for flowshop scheduling problems," *Computers & Industrial Engineering*, vol. 30, no. 4, pp. 1061–1071, 1996.

[108] E. Hart, P. Ross, and D. Corne, "Evolutionary scheduling: A review," *Genetic Programming and Evolvable Machines*, vol. 6, no. 2, pp. 191–220, 2005.

[109] F. Werner, "Genetic algorithms for shop scheduling problems: a survey," *Preprint*, vol. 11, p. 31, 2011.

[110] K. Dahal, K. C. Tan, and P. I. Cowling, *Evolutionary scheduling.* Springer Science & Business Media, 2007, vol. 49.

[111] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2016. [Online]. Available: http://www.gurobi.com

[112] R. Jain, D.-M. Chiu, and W. R. Hawe, *A quantitative measure of fairness and discrimination for resource allocation in shared computer system.* Eastern Research Laboratory, Digital Equipment Corporation Hudson, MA, 1984, vol. 38.

[113] J. Wu, D. Ghosal, M. Zhang, and C.-N. Chuah, “Delay-based traffic signal control for throughput optimality and fairness at an isolated intersection,” *IEEE Trans. Veh. Technol.*, 2017.