Decentralized Conflict Resolution for Autonomous Vehicles

J. An

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Decentralized Conflict Resolution for Autonomous Vehicles

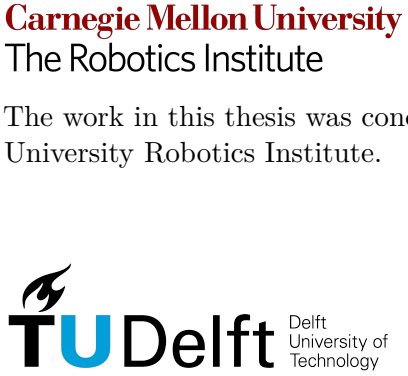
Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

J. An

November 20, 2020

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology



The work in this thesis was conducted at the Intelligent Control Lab of the Carnegie Mellon University Robotics Institute.

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Decentralized Conflict Resolution for Autonomous Vehicles

by

J. An

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: November 20, 2020

Supervisor(s):

dr. C. Liu

dr. G. Giordano

Reader(s):

dr.ir. T. Keviczky

dr. J. Alonso-Mora

dr. L. Ferranti

Abstract

This work presents a decentralized optimization conflict resolution method based on a novel Alternating Directions Method of Multipliers (ADMM) variant and model predictive control (MPC). The variant, titled Online Adaptive Alternating Direction Method of Multipliers (OA-ADMM) aims to unify the application of ADMM to online systems, i.e. systems where fast and adaptive real-time optimization is crucial, into one framework. OA-ADMM introduces two user-designed functions: the similarity function (a forgetting factor between two time steps of the online system) and the adaptation function (adjusting the penalty parameters between updates). The similarity function is what allows OA-ADMM to be applied to online systems where conventional optimization is too slow; the adaptation function allows the user to adjust the online feasibility of the system. We prove convergence in the static case and give requirements for online convergence.

Combining OA-ADMM and MPC allows for robust decentralized motion planning and control that seamlessly integrates decentralized conflict resolution, instead of using separate subsystems or hierarchical optimization. The additional robustness is achieved by using the adaptation function of OA-ADMM as an additional safety measure, allowing the prioritization of certain constraints for (nearly) unsafe states, whilst the similarity function allows optimization at the desired control frequency. This method is compared with convention ADMM in Matlab, resulting in significant improvements in robustness and conflict resolution speed.

Finally, we compare our OA-ADMM and MPC based decentralized conflict resolution method against conventional decentralized conflict resolution methods in the CARLA vehicle simulator. The results show that the OA-ADMM based method has improved performance, safety, robustness, and generality compared with traditional methods. The method also has fewer requirements in terms of prior knowledge (e.g., the geometry of the intersection), making it usable in almost any situation.

Table of Contents

1 Introduction **1**

1-1 Motivation . . . 1

1-2 History and Related work . . . 2

1-3 Scope . . . 3

1-4 Structure . . . 3

I Fundamentals of Conflict Resolution **5**

2 Preliminaries **7**

2-1 Mathematical Notation . . . 7

2-2 State-space Formulation . . . 7

2-3 Kinematic Models . . . 9

2-4 Primal and Dual Decomposition . . . 12

2-5 Centralized and Decentralized Definitions . . . 14

2-6 Vehicle Identification ID . . . 15

2-7 Communication . . . 16

3 Conflict Resolution **21**

3-1 Centralized Conflict Resolution . . . 21

3-2 Decentralized Conflict Resolution . . . 23

3-3 Deadlock Detection . . . 25

3-4 Deadlock Resolution . . . 32

4 Problem Statement **35**

4-1 Environment . . . 35

4-2 Problem Objective . . . 39

II ADMM, variations, and OA-ADMM **43**

5 Alternating Direction Method of Multipliers (ADMM) **45**
5-1 Problem Formulation 45
5-2 Convergence Results 48

6 Adaptive ADMM (A-ADMM) **55**
6-1 Role of the penalty parameter ρ 55
6-2 Residual Balancing 57
6-3 Spectral Penalty Parameter Selection 59

7 Online ADMM (O-ADMM) **65**
7-1 Online ADMM in general 65
7-2 Online distributed motion planning for multi-vehicle system 67
7-3 Linearization and Convexification 70
7-4 Online Feasibility 73

8 Online Adaptive-ADMM (OA-ADMM) **75**
8-1 Problem Formulation 75
8-2 Methodology 76
8-3 Convergence Results 79

III Decentralized Conflict Resolution **87**

9 Decentralized Conflict Resolution using OA-ADMM and MPC **89**
9-1 Methodology 90
9-2 Numerical Results 97
9-3 Discussion 102

10 Simulations **109**
10-1 Simulator Choice 109
10-2 Decentralized Protocols 110
10-3 Benchmarking Tool 115
10-4 Implementation 118
10-5 Simulation Results 126

11 Discussion **129**
11-1 Benchmarking Tool Discussion 129
11-2 OA-ADMM MPC Results Discussion 132
11-3 OA-ADMM MPC vs AMP-IP Results Discussion 132
11-4 OA-ADMM MPC vs TDCR Results Discussion 133
11-5 Summary 134

12 Conclusion **135**
12-1 Summary 135
12-2 Limitations 136
12-3 Future Work 136

Glossary **149**
List of Acronyms 149

List of Figures

1-1 Thesis Structure Diagram 4

2-1 The unicycle model 10
2-2 The bicycle model 11
2-3 Centralized, Decentralized graph examples 16

3-1 Yield graph 26
3-2 Yield Graph Modeling Examples 30

4-1 Environment categories 36
4-2 Homotopy Example 37
4-3 Examples of structured intersections 38
4-4 Special intersection cases 39
4-5 Unmanaged intersection deadlock example 40

5-1 ADMM Steps Diagram 48

6-1 Example of penalty parameter effects on optimality 56

7-1 Online ADMM Steps Diagram 69
7-2 Feasible sets for (linearized) distance function. 72

8-1 OA-ADMM Steps Diagram 78

9-1 Conflict Resolution using OA-ADMM and MPC 89
9-2 OA-ADMM MPC Steps Diagram 92
9-3 Potential behavior of Δ^t and Δ^k 94
9-4 Potential behavior of $\Delta^{t,k}$ 95

9-5 Conflict resolution simulation environment for OA-ADMM and ADMM 99
9-6 Snapshots of OA-ADMM and MPC for four vehicles 100
9-7 Snapshots of OA-ADMM and MPC for eight vehicles 101
9-8 Conflict Resolution Time for O-ADMM 102
9-9 Mean squared violation for O-ADMM 103
9-10 Conflict Resolution Time for OA-ADMM 104
9-11 Mean squared violation for OA-ADMM 105
9-12 Example of a deadlock for O-ADMM MPC 106
9-13 Conflict Resolution Time for successful cases using O-ADMM 107
9-14 Conflict Resolution Time for successful cases using OA-ADMM 108
9-15 Example of the D_{mult} effects. 108

10-1 AMP-IP Flowchart 111
10-2 AMP-IP Deadlock example 112
10-3 TDCR Flowchart 113
10-4 CARLA 4-way Intersection 116
10-5 4 Way Intersection Possible Conflicts 117
10-6 Conflicting component of a trajectory 118
10-7 CARLA 4-way Intersection 2D View 119
10-8 Circular vs capsular hull 120
10-9 Capsule parameters 121
10-10Minimum distance cases between capsules 122
10-11Average delays for the decentralized protocols 127

Preface and Acknowledgments

Given that of the field of conflict resolution for autonomous vehicles is at the intersection of many fields and disciplines, a lot of related works require knowledge outside the scope expected of a Systems and Control MSc student. As a result, the thesis is of considerable length in order to keep the thesis self-contained. Please keep this in mind when reading the initial chapters of this work.

I would like to thank my supervisors dr. C. Liu and dr. G. Giordano for their detailed, yet to the point, feedback during the writing of this thesis. I would also like to thank dr. C. Liu for giving me the opportunity to visit the Intelligent Control Lab at Carnegie Mellon University; I thoroughly enjoyed my time at in Pittsburgh and learned a lot from my fellow lab members at the Intelligent Control Lab.

Delft, University of Technology — J. An
November 20, 2020

Chapter 1

Introduction

1-1 Motivation

Traffic congestion has a significant economic and ecological cost attributed to it. Driving delays have been estimated to cost $160 billion a year in the US alone [1]; on top of the economic cost there is also a negative effect on public health [2], and traffic congestion also has a large impact on carbon dioxide emissions, which could be reduced significantly [3]. Most estimates place fully autonomous vehicles somewhere within the next decade (2020-2030) [4]. It is important that autonomous vehicles do not worsen congestion but reduce it instead; since intersections are a major contributor to traffic delays and accidents, it is crucial that autonomous vehicles are equipped to deal with them efficiently [5].

Studies have shown that there is a big gap between the readiness of connected and autonomous vehicles, and the urban infrastructure [6]. It is therefore not realistic to expect every intersection or other conflict zones to be equipped with the infrastructure required to centrally resolve the conflicts. Because of this, it is necessary that autonomous vehicles are able to resolve these conflicts without any external infrastructure. This is especially true for unmanaged intersections outside urban environments where there are often no traffic lights or signs. Autonomous vehicles, however, face additional difficulties when navigating unmanaged intersections using decentralized policies due to the potential for deadlocks or accidents. These difficulties can be resolved by using communication and conflict resolution protocols among autonomous vehicles.

The goal of this thesis is conduct thorough investigation of the problem and the state of the art, mainly focusing on its limitations, in combination with the proposal of a novel approach aimed to improve performance and reduce the limitations. Specifically, the improvements are made through using optimization to resolve conflicts, whilst simultaneously generalizing the problem to any structured or unstructured environment.

1-2 History and Related work

Initial research into connected vehicle environments started to take off with the proposal of Dedicated Short-Range Communication (DSRC) standards around 2005 [7]. Early research was focused on driver assistant systems, such as adaptive cruise control [8] and collision mitigation brake systems [9]. The desire for higher levels of autonomy led to autonomous lane changes and merge maneuvers [10]. At around the same time, the research began on intersection management, with one of the earliest centralized works being a First Come First Serve (FCFS) reservation-based intersection control in [11] and [12]. Gradual improvements were made on a similar framework like the Look-ahead Intersection Control Policy (LICP), which took potential delays into account [13]. Other improvements were made to the centralized approaches such as in [14], which transformed the problem into a convex space domain problem. The convex formulation was then used by [15] for a centralized MPC approach.

Centralized conflict resolution has the major limitation that, when the central manager fails, the entire environment comes to a halt. Furthermore, the requirement of central conflict resolution hardware at all locations where conflicts can occur has two major issues. First, there is no simple way to identify all physical locations where conflicts can occur. Second, the amount of hardware needed comes at a very high cost, including maintenance. These limitations of centralized approaches resulted in the desire for decentralized approaches for intersection conflict resolution as well. One of the first was the priority-based approach in [16], which utilized the token ring[1] principle from computer networks. Another example is the decentralized-navigation-functions approach for point mass vehicles [17]. It is also possible to use sequential local optimization based on the time until reaching the intersection [18].

More traditional approaches are the decentralized intersection protocols that use various priority policies [19, 20, 21]. An improvement of this framework allows real-time adjustments of the priorities and the utilization of constrained optimal control, allowing for better overall performance [22].

Whilst not incorporating the entire conflict resolution problem, recent development in the field of online multi-agent motion planning is closely related to our work on optimization based decentralized conflict resolution. Online distributed motion planning for multi-agent systems has been proposed in [23], utilizing a single iteration receding horizon approach, with the goal of formation control; [24] expands the method by also considering inter-vehicle collision avoidance through the use of separating hyperplanes. The ADMM variant introduced in [25], intended for autonomous vessels, utilizes a central coordinator and claims to improve the convergence rate through iteratively adding approximated collision avoidance constraint. The authors of [26] introduce a method similar to that of [23, 24], utilizing a linearized collision avoidance constraint and incorporating deadlock-protection. A distributed MPC approach is proposed in [27], incorporating the residual balancing method from [28]. The authors of [29] utilize distributed trajectory optimization based on a decomposition technique that distinguishes itself from ADMM based methods by avoiding communication between agents until convergence. A nonlinear MPC-based approach is proposed in [30], reducing the need for linearized constraints.

[1] Token ring is a principle in which a token is passed around in a ring, where holding the token grants the right to transmit information.

Unlike our proposed approach, the methods in [23, 24, 25, 26, 27, 30] do not utilize an adaptive penalty function or a similarity based online update, making the methods more prone to disturbances in online systems and slower to converge. Whilst [27] utilizes a simple residual balancing adaptation scheme, it does not utilize the penalty parameter for increased robustness. As opposed to our iterative linearization based approach, the method in [30] allows the use of nonlinear constraints at the cost of having to use a non-convex solver.

Adaptive penalty parameters have been explored before in [28], where a residual balancing scheme is proposed along with conditions for convergence. A proximal gradient based method, proposed in [31, 32], claims faster convergence without the requirement for manual tuning.

Unlike the methods in [28, 31, 32], our proposed Online Adaptive Alternating Direction Method of Multipliers (OA-ADMM) framework does not explicitly define an adaptation scheme. We choose to leave the exact adaptation method open to encourage novel uses of the adaptation function, e.g., improving online robustness for motion planning problems or incorporating a priority based conflict resolution method.

1-3 Scope

The main scope of this work includes a short summary on the decentralized conflict resolution problem, along with a thorough analysis of ADMM and some of its variants. This leads to the introduction of our novel Online Adaptive ADMM (OA-ADMM) framework, for which the convergence results are analyzed and proven where possible. The OA-ADMM method is then applied to the decentralized conflict resolution problem to achieve decentralized-optimization-based conflict resolution integrated with motion planning and control.

1-4 Structure

This thesis can be divided into three main parts, each centered around a different topic. Part I has two main purposes: 1. introducing the required background information for the rest of the report in Chapter 2 and Chapter 3, and 2. providing the problem statement in Chapter 4. Chapter 2 includes information on the notation used in the thesis, kinematic models for the vehicles, and a summary of expected prior knowledge. Chapter 3 provides a brief overview of the conflict resolution problem, its complications, and some proposed methods to solve them. Chapter 4 establishes definitions for the environment, and derives the subproblems from the main conflict resolution problem.

Part II contains an in depth overview of Alternating Direction Method of Multipliers (ADMM) and its variations. The variations covered include conventional ADMM in Chapter 5, adaptive ADMM in Chapter 6, online ADMM in Chapter 7, and finally our proposed Online Adaptive ADMM in Chapter 8. Chapter 5 introduces the Alternating Direction Method of Multipliers, covers its convergence results and provides proof for the convergence. Chapter 6 covers the role of the penalty parameter in ADMM, and variants which utilize an adaptive penalty parameter. Chapter 7 then briefly summarizes various methods which utilize ADMM for online systems. Finally, our novel ADMM variant is proposed in Chapter 8, where it is covered in depth, similarly to how ADMM is covered in Chapter 5, providing convergence results and requirements.

The last part, Part III, covers the application of OA-ADMM to decentralized conflict resolution in combination with model predictive control (MPC) in Chapter 9. The method is then compared with other methods in Chapter 10, after which the results are discussed in Chapter 11. Chapter 9 begins with deriving a decentralized formulation of the main problem given in Chapter 4, followed by the design of a similarity function and an adaptation function, along with some initial numerical results. Chapter 10 then compares the method against other decentralized conflict resolution methods such as AMP-IP and TDCR. Chapter 11 concludes this thesis with a discussion of the proposed method, summarizing its advantages and potential pitfalls, along with an in-depth analysis of simulation results.

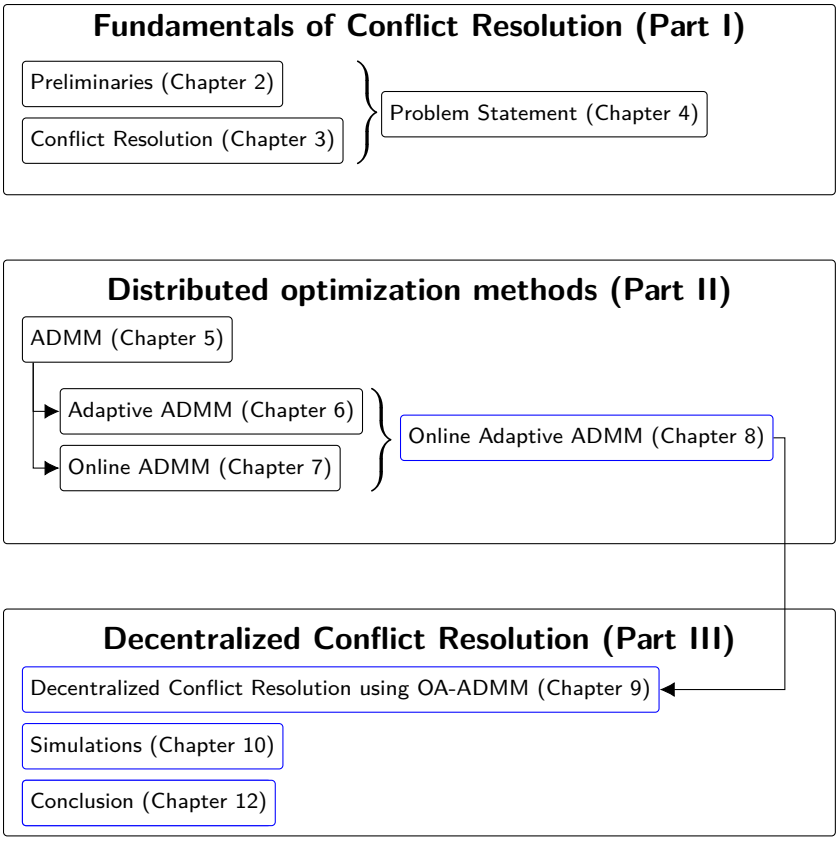


Figure 1-1: Diagram showing the structure of the thesis, novel work is highlighted with blue borders.

Part I

Fundamentals of Conflict Resolution

This part has two main purposes: 1. introducing the required background information for the rest of the report in Chapter 2 and Chapter 3, and 2. providing the problem statement in Chapter 4.

Chapter 2 includes information on the notation used in the thesis, kinematic models for the vehicles, and a summary of expected prior knowledge. Chapter 3 provides a brief overview of the conflict resolution problem, its complications, and some proposed methods to solve them. Chapter 4 establishes definitions for the environment, and derives the subproblems from the main conflict resolution problem.

Chapter 2

Preliminaries

This chapter serves as a short summary on preliminaries required for the rest of the thesis. The concepts explained in this chapter are not in the order of their usage.

2-1 Mathematical Notation

Generally the notation used in the paper are as follow:

- Sets are denoted by a calligraphic symbols, e.g., $\mathcal{S}$
- Complexity related symbols are denoted by Euler script calligraphic symbols, e.g., $\mathcal{P}, \mathcal{O}$
- Matrices are denoted by bold italicized capital letters, e.g., $\boldsymbol{A}$
- Vectors are denoted by bold italicized lower case letters, e.g., $\boldsymbol{a}$
- Scalars are denoted by a regular lower case or capital letter, e.g., a, A. Generally no distinction is made between continuous and discrete scalars.
- Subscripts generally denote the agent or set where the symbol belongs to, e.g., $(\cdot)_i, (\cdot)_{\mathcal{I}}$
- Superscripts are generally used to specify types, scenarios, or iteration, e.g., $(\cdot)^A$ or $(\cdot)^k$

2-2 State-space Formulation

The state-space formulation is a commonly used framework to represent a dynamical system in terms of its inputs (u), outputs (y), and states (x). The system state x conventionally represents the state of a system at a certain time, the inputs represent how you can affect (or control) the system, and finally the output is what you can measure (or observe), and if desired, can also be an arbitrary performance index.

2-2-1 General State-space Formulation

The general formulation for a state-space model is

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{u}(t)) \\ \boldsymbol{y}(t) &= \boldsymbol{h}(t, \boldsymbol{x}(t), \boldsymbol{u}(t)), \end{aligned} \tag{2-1}$$

where $\boldsymbol{f}$ and $\boldsymbol{h}$ describe the state dynamics and output dynamics, respectively, as a function of time, states, and inputs. This general formulation is used for nonlinear system dynamics, as linear system dynamics can be represented in a matrix form.

2-2-2 Linear State-space Formulation

Linear state-space formulation is the most common usage of state-spaces due to the wide availability of analysis techniques, including the analysis of stability, controllability, observability, etc. However, as most physical systems are nonlinear, linearization has to be applied first.

Linearization involves representing a nonlinear system as a linear system that approximates the nonlinear dynamics around a linearization point. This is generally done through a first order Taylor expansion around the equilibrium, however other points can be chosen as well if desired[2]. It should also be noted that the analysis of a linearized system only applies to the linearization itself, which is only valid locally in a neighborhood around the linearization point. The conclusions on the linearized system do not necessarily apply globally for the nonlinear system.

The linear state-space representation has four matrices $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$ and three vectors $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u})$, and in the continuous time it is:

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \\ \dot{\boldsymbol{y}}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t), \end{aligned} \tag{2-2}$$

where $\boldsymbol{x}, \dot{\boldsymbol{x}}$ are the state vector and its derivative; $\boldsymbol{u}$ is the input vector; $\boldsymbol{y}$ is the output vector; $\boldsymbol{A}$ is the system state matrix, $\boldsymbol{B}$ is the input matrix; $\boldsymbol{C}$ is the output matrix; and $\boldsymbol{D}$ is the direct feedforward matrix.

2-2-3 Discrete State-space Formulation

As computers cannot directly evaluate continuous state spaces, a discrete state-space formulation is desired. In this case, system states at the next time step are defined as a function of the current states. In order to find the discrete state-space formulation from the continuous one, discretization has to be performed.

Discretization can be achieved in various ways, either exact or through approximation. Exact discretization involves an analytical derivation using the matrix exponential $(e^{\boldsymbol{A}T})$, where $\boldsymbol{A}$ is the system matrix and T is the step size, and integral operations. Because of the heavy computational requirements involved with matrix exponentials and integral operations, an

[2]One use case of multiple linearizations is for Linear Parameter Varying (LPV) systems, where the system switches or interpolates between linearizations depending on the value of a parameter.

approximate solution is generally used. The most commonly used approximations attempt to approximate the matrix exponential $e^{\boldsymbol{A}T}$, for example Euler's method utilizes the approximation

$$e^{\boldsymbol{A}T} \approx \boldsymbol{I} + \boldsymbol{A}T$$

for small time steps. Another example is the Tustin transform, which is also known as the bilinear transform, which utilizes

$$e^{\boldsymbol{A}T} \approx \left(\boldsymbol{I} + \frac{1}{2}\boldsymbol{A}T\right)\left(\boldsymbol{I} - \frac{1}{2}\boldsymbol{A}T\right)^{-1}$$

as its approximation. The Tustin method is a popular method as the stability properties are preserved with the discretization.

The discrete linear state-space representation is then written as

$$\begin{aligned} \boldsymbol{x}[k+1] &= \boldsymbol{A}_d\boldsymbol{x}[k] + \boldsymbol{B}_d\boldsymbol{u}[k] \\ \boldsymbol{y}[k] &= \boldsymbol{C}_d\boldsymbol{x}[k] + \boldsymbol{D}_d\boldsymbol{u}[k], \end{aligned} \tag{2-3}$$

with k being the time step. It should be noted that the matrices $\boldsymbol{A}_d, \boldsymbol{B}_d, \boldsymbol{C}_d, \boldsymbol{D}_d$ for a certain system are not the same for the continuous time case and the discrete time case.

2-3 Kinematic Models

Kinematic models allow the derivations of constraints on the vehicle states purely based on the geometry of the vehicle. A consequence of using kinematic models is the inability to model concepts like inertia. The geometric approach also results in the no-slip condition, which implies that the surface of the tires has no relative velocity with respect to the ground. The no-slip condition therefore implies that the wheels are assumed to only have velocity in their longitudinal direction, as no lateral velocity component can be derived from its geometry.

2-3-1 Unicycle Model

The unicycle model, shown in Figure 2-1, is one of the simplest non-linear and non-holonomic vehicle models. The model assumes that the vehicle can be modeled as a single wheel, with its inputs being the forward acceleration and its angle. Additionally a point mass assumption is used with the center of mass (CoM) coinciding with the geometric center of the vehicle. The kinematic constraints for the unicycle model in the general state space formulation are

$$\dot{x}_i = v_i \cos\theta_i \tag{2-4a}$$
$$\dot{y}_i = v_i \sin\theta_i \tag{2-4b}$$
$$\dot{v}_i = a_i \tag{2-4c}$$
$$\dot{\theta}_i = \omega_i, \tag{2-4d}$$

where the state vector is $\boldsymbol{x} = [\text{x y } v\ \theta]^\top$, with x and y being the coordinates of the vehicle, v being the forward velocity, and θ being the orientation of the vehicle measured at its geometric

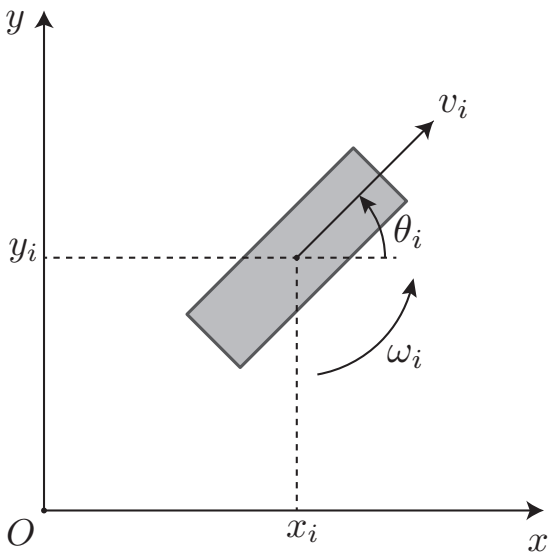


Figure 2-1: The kinematics for a unicycle model. The global coordinates for the geometric center of the unicycle model are x_i and y_i with θ_i being the angle of the wheel relative to its center of rotation. The control inputs for the model are v_i and ω_i for the forward velocity and the angle respectively.

center; the input vector is $\boldsymbol{u} = [a\ \omega]^\top$, with a being the forward acceleration and ω being the angular velocity. The main advantage of the unicycle model is its simplicity whilst still incorporating non-holonomic constraints. Whilst it is more accurate than representing the vehicle as a holonomic vehicle, there is a problem with the assumptions made for a conventional car. Since most conventional cars have some form of front axle Ackermann steering[3], their center of rotation will be varying along the axis of the rear axle [33]. As the unicycle assumes the center of rotation being at the center of the vehicle, it cannot be directly applied to a conventional vehicle.

2-3-2 Bicycle Model

The bicycle model, shown in Figure 2-2, is a much closer approximation of the kinematics of a conventional car which takes into account the inability to directly control the heading of the vehicle. The model makes use of the property that Ackermann steering results in both wheels on the front axis pointing towards the center of rotation (CoR), this allows the front axle to be replaced with a single wheel without changing the kinematics. The rear axle is also replaced by a single wheel to reduce the model from 4 wheels to 2 wheels. The kinematic

[3]Type of steering linkage geometry that aligns the center of rotation of the inside and outside wheels when turning.

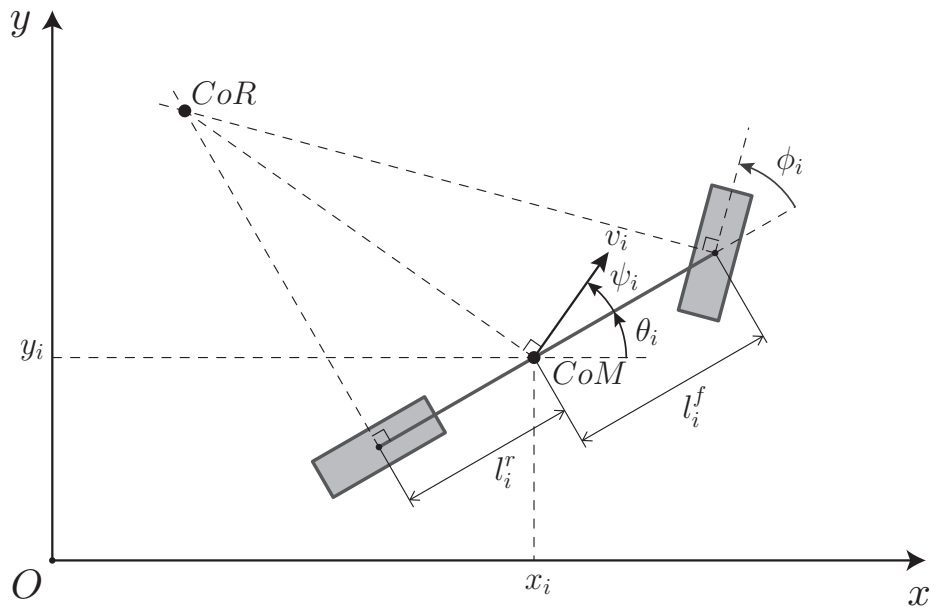


Figure 2-2: The kinematics for a bicycle model. x_i, y_i and θ_i describe the position and orientation of the bicycles center of mass (CoM) in the global coordinate frame, with the distance from the CoM to the axles being l_i^r and l_i^f for the rear and front. The steering angle of the front wheel is described by ϕ_i, which is used to calculate the center of rotation (CoR). Using geometric relations the heading angle ψ_i and velocity v_i at the CoM can be derived.

equations for the bicycle model in the general state space formulation are

$$\dot{x}_i = v_i \cos(\theta_i + \psi_i) \tag{2-5a}$$
$$\dot{y}_i = v_i \sin(\theta_i + \psi_i) \tag{2-5b}$$
$$\dot{v}_i = a_i \tag{2-5c}$$
$$\dot{\theta}_i = \frac{v_i}{l_i^r} \sin \psi_i \tag{2-5d}$$
$$\psi_i = \tan^{-1}\left(\tan \phi_i \frac{l_i^r}{l_i^r + l_i^f}\right), \tag{2-5e}$$

where the state vector is $\boldsymbol{x} = [\text{x y } v\ \theta\ \psi]^\top$, with x and y being the coordinates of the vehicle, v being the forward velocity at its CoM, ψ being the heading angle of the vehicle at the CoM, θ being the vehicle orientation at its CoM; the input vector is $\boldsymbol{u} = [a\ \phi]$, where a is the forward acceleration at the CoM and ϕ is the front wheel steering angle. The main advantage of the bicycle model is the separation of the heading and angle at the CoM, from the orientation of the vehicle and the steering angle, in comparison with the unicycle model in which the three angles are assumed to be identical, allowing for configurations which are not possible for a conventional vehicle. Despite the bicycle model being a kinematic model, at lower speeds it is often as accurate as a linear dynamic model [34]. Since this thesis is predominantly about conflict resolution we shall not delve into dynamic models.

2-3-3 Constraints

In order for the model to be representative of a real vehicle, some constraints should be defined. These can be separated into physical constraints, which are defined by the physical limitations of a vehicle, and safety constraints, which ensure the safety in the environment.

2-3-3-1 Physical Constraints

The physical constraints are based on the physical limitations of the vehicles. These can vary per vehicle type, make, model, year, etc. Load, usage, and slight manufacturing differences can also affect these constraints. The constraint parameters we shall account for are:

- a^{max} (m/s^2): maximum acceleration derived from the maximum engine torque.
- a^{min} (m/s^2): minimum acceleration, or maximum deceleration, derived the maximum braking torque.
- v^{max} (m/s): maximum velocity based on the maximum torque and friction or electronic limiter.
- $\dot{\phi}^{max}$ (rad/s): maximum angular velocity for steering.
- ϕ^{max} (rad): maximum steering angle, defined for the absolute value of the steering angle relative to its natural position in.

2-3-3-2 Safety Constraints

Certain constraints will have to be set for each vehicle to verify the safety of the algorithms. These safety constraints depend on the model, sampling frequency and other uncertainties. In general the safety constraints should guarantee that the vehicle will not collide with any obstacle; some stronger constraints can also be considered to avoid psychological harm to other agents. For example by constraining the maximum speed and accelerations around pedestrians to avoid scaring them. For simplicity the safety constraints shall be defined by a minimum distance (D_{min}) between the edge of the ego vehicle[4] and an obstacle.

2-4 Primal and Dual Decomposition

Primal and dual decomposition involves splitting a problem into separate segments that can be computed separately in order to find the solution for the original problem. Using this approach, the main problem can be distributed among agents to reduce the computational requirements so that it can be performed faster with less computational power. However, there is usually a trade-off between convergence speed, feasibility, and solution accuracy depending on the problem and methods.

2-4-1 Primal Decomposition

Primal decomposition methods decompose the original primal problem into separate sub-problems, which can be solved independently. This requires that the original problem has a

[4]Ego vehicle refers to the vehicle performing the measurements, planning, and control.

complicating variable such that the subproblems become decoupled if that variable is fixed. For example, take the main problem

$$\begin{aligned} \min_{x,c} \quad & \sum_i f_i(x_i) \\ \text{subject to} \quad & x_i \in \mathcal{X}_i^f \ \forall i, \\ & A_i x_i \leq c \ \forall i, \\ & c \in \mathcal{C}^f, \end{aligned} \tag{2-6}$$

where c is a the complicating variable. Decomposing Equation (2-6) using primal decomposition would then result into a lower level decentralized subproblem, and a higher level master problem. The low level problem is

$$\begin{aligned} \min_{x_i} \quad & f_i(x_i) \\ \text{subject to} \quad & x_i \in \mathcal{X}_i^f, \\ & A_i x_i \leq c, \end{aligned} \tag{2-7}$$

which can be computed separately for each agent i. The high level master problem would then be

$$\begin{aligned} \min_{c} \quad & \sum_i f_i^*(c) \\ \text{subject to} \quad & c \in \mathcal{C}^f, \end{aligned} \tag{2-8}$$

where f_i^* is the solution to Equation (2-7). As the subproblems are simply a reformulation of the original problem, the solution satisfies the primal constraints, furthermore the solution to the master problem remains the same as the original problem, making this decomposition truly primal.

2-4-2 Dual Decomposition

In addition to primal decomposition, there is also dual decomposition in which a dual of the original problem is decomposed. The dual decomposition method is a special case of the dual ascent method, requiring that the minimization step in dual ascent[5] can be performed independently and in parallel. For example, given the main problem

$$\begin{aligned} \min_{x} \quad & \sum_i f_i(x_i) \\ \text{subject to} \quad & x_i \in \mathcal{X}_i^f \ \forall i, \\ & \sum_i g_i(x_i) \leq c, \end{aligned} \tag{2-9}$$

where g_i can be any function of x_i, it would be impossible to decompose the problem into a lower level problem by fixing c as done in Equation (2-7). In fact, the satisfaction of the constraint is directly dependent on the solution of the other agents. A simple solution would involve distributing c among agents guaranteeing the satisfaction of the constraints, for

[5] Dual ascent consist of two steps, a minimization step and a dual variable update step.

example $g_i(x_i) \leq \frac{c}{N}$, where N is the total amount of agents. This however does not guarantee that the primal solution can be found, as it is likely that the actual optimum does not have equal distribution. Another possibility involves the use of Lagrange multipliers, with the Lagrangian $\mathcal{L}(x, \lambda) = \sum_i \Big(f_i(x_i) + \lambda^\top g_i(x_i) \Big)$. The low level subproblem can then be written as

$$\begin{aligned} \min_x \quad & f_i(x_i) + \lambda^\top g_i(x_i) \\ \text{subject to} \quad & x_i \in \mathcal{X}_i^f, \end{aligned} \tag{2-10}$$

where λ is the Lagrange multiplier. The accompanying master problem then involves updating λ as follows

$$\begin{aligned} \max_\lambda \quad & \sum_i d_i(\lambda) - \lambda^\top c \\ \text{subject to} \quad & \lambda \geq 0, \end{aligned} \tag{2-11}$$

where $d(\lambda)$ is the Lagrange dual function. The solution of the master problem in Equation (2-11) would then result in the solutions of the accompanying subproblems of Equation (2-10) having no duality gap. It should be noted that decomposition through this method only works if the dual function has strong duality. Furthermore, convergence is only guaranteed at the limit, with run-time solutions having no guarantees of feasibility for original problem. Strong duality can be guaranteed for convex problems using Slater's condition, which requires a problem to be strictly feasible and convex. The mathematical form of this for a convex function $f_i(x)$ would be

$$\begin{aligned} & \exists x \in \mathbf{relint}(\mathcal{D}) \\ \text{subject to} \quad & f_i(x) < 0, \end{aligned} \tag{2-12}$$

where **relint** is the relative interior and $\mathcal{D}$ is the domain. The relative interior is the interior of a set within its affine hull, i.e. $\mathbf{relint}(\mathcal{X}) := \{x \in \mathcal{X} : \exists \epsilon > 0, B_\epsilon(x) \cap \mathbf{aff}(\mathcal{X}) \subseteq \mathcal{X}\}$, where $B_\epsilon(x)$ is an ϵ-ball centered on x and **aff** is the affine hull[6].

When dual decomposition methods are applied in actual systems, the run-time solution is usually projected on the primal feasible set. Actual implementations also frequently use update rules for the Lagrange multiplier instead of optimization to reduce computational costs: for example, instead of Equation (2-11), one could use $\lambda_{k+1} = \lambda_k + \alpha_k \sum_i g_i(x_i)$, where α is the step size.

These are only a few examples of decomposition, whilst in reality the method varies depending on the structure of the original problem. The take-home message however is that primal decomposition is effective, but has hard limitations on the original problem, whereas dual decomposition is more general, but has downsides like infeasible run-time solutions.

2-5 Centralized and Decentralized Definitions

Centralized and decentralized can be used to refer to different concepts depending on the context, field, or author. In general centralized systems refer to systems where there is a single central node which holds authority over all its connected nodes. For example in the

[6]The affine hull of $\mathcal{S}$ is the smallest affine set that fully encloses $\mathcal{S}$, e.g., for three points in $\mathbb{R}^3$, not on one line, the convex hull is a triangle connecting the three points, whereas the affine hull will be the entire plane containing all three points.

case of a team, the team leader has higher authority and therefore, has knowledge of all the team members and decides the actions for the entire team. A decentralized system on the other hand is characterized by the lack of one central node which holds the highest authority. For the same example, a decentralized system would have no team leader, instead the team members would each make their own decisions. Decentralized does not however mean that there is no communication between nodes, or in the case of the example, the team members.

In some literature a distinction is made between decentralized and distributed systems. The exact distinction, however, is not consistent within literature even within the same field. We shall generally use **decentralized** in the context of authority distribution and **distributed** in the context of computational distribution, where authority refers to aspects such as commanding other agents, whereas computational distribution is related to what part of the computation is performed by which agent. We shall also make a distinction between levels of centralization in an attempt to reduce confusion when referring to other literature. Figure 2-3 showcases the difference between a centralized system and a decentralized system together with an example in which both occur. The nodes can represent individual agents, however the nodes do not have to be homogeneous. For example node 1 in Figure 2-3(a) can be a central traffic manager with nodes 2-4 being vehicles. Considering a traffic management environment, a graph as depicted in Figure 2-3(c) can also occur. For example subgroups A and B can represent intersection managers where locally all the decisions are made centrally, the overall traffic management could then be decentralized where each intersection manager only communicates with its neighbors. In terms of graph theory the following definitions can be given based on our examples:

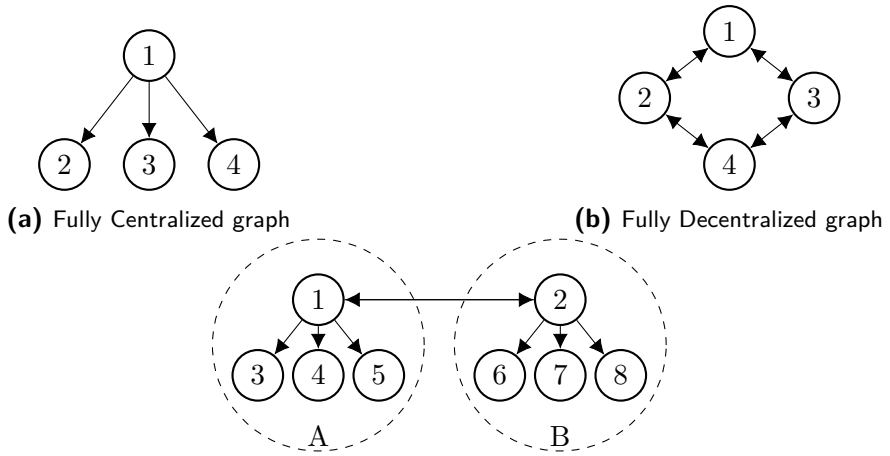
Definition 2-5.1 (Fully Centralized graph)**.** A directed or undirected graph is considered to be fully centralized when there is **a** single node that completely disconnects all other nodes, in the event that the node and its edges are not present.

Definition 2-5.2 (Fully Decentralized graph)**.** A directed or undirected graph is considered to be fully decentralized when there is **no** single node that completely disconnects all other nodes, in the event that the node and its edges are not present.

Definition 2-5.3 (Hybrid graph)**.** A directed or undirected graph is considered to be hybrid when it can be represented as a combination of fully centralized and fully decentralized subgraphs.

2-6 Vehicle Identification ID

In order to resolve ties when determining a priority order, some algorithms rely on a unique vehicle identification number to resolve conflicts when all other methods fail to identify a higher priority. All vehicles must be equipped with such a number and the number must be unique to each vehicle. The uniqueness of the IDs is also crucial for proper communication and decentralization, so that there is no confusion with regards to what information belongs to which vehicle. Such IDs already exist in the form of the Vehicle Identification Number (VIN), which are assigned to all road vehicles with the exception of some military vehicles and other special cases. The VIN implementation might differ per region, however they are all based on either ISO 3779 or ISO 3780 and are compatible [35, 36]. Because of this whenever an algorithm requires the use of a unique vehicle ID, the VIN shall be used.



(a) Fully Centralized graph

(b) Fully Decentralized graph

(c) Hybrid graph

Figure 2-3: Example showcasing the various levels of centralization in a graph form; arrows indicate authority whereas bidirectional arrows indicate equal authority. It can be seen that Figure 2-3(a) is fully centralized as node 1 holds authority over all nodes. The nodes in Figure 2-3(b) on the other hand, all have equal authority with their neighbors, there is no single central node. It is more complicated to classify the graph in Figure 2-3(c). When isolating the nodes and edges for subgraphs A and B, they will both appear to be fully centralized systems; however if you represent each subgraphs as singular node the entire system appears to be fully decentralized.

2-7 Communication

Communication is of crucial importance for both centralized and decentralized systems and its properties have an effect on which methods are feasible and which are not. In this section some important types of communications are introduced together with contemporary standards that define how the communication should occur. We will also introduce a simplified communication protocol for the sake of simplicity and consistency.

2-7-1 Technical Requirements

In general, the requirement for communication in a connected environment is the ability to transmit relevant information to nearby agents within a short time frame. The maximum time delay affects the safety constraints, which in turn affect the speed at which the vehicles can travel. Additionally, time delays can introduce instabilities when not accounted for. Another important aspect is the range of the communication: in general more complete information allows for better solutions. The possible information transmitted is also of high importance, for a decentralized problem communicating vehicle states allows better solutions as there is less safety margin required to compensate for inaccurate estimations from sensor measurements. The desired trajectory and target of other vehicles are also of importance in order to predict future behavior of other vehicles, for example to predict at what time and what area a vehicle will cross an intersection. The communication should also contain some metadata to identify the message and its sender.

2-7-2 Vehicle to Infrastructure communication

Vehicle to Infrastructure (V2I) communication contains all information transfer between vehicles and infrastructure, this is mainly used to send vehicle states and future trajectory to a traffic manager; it can also be used to inform vehicles on things like future signal timings or traffic congestion. An example of how V2I communication can be used is the UR:BAN research project (2012-2016) in Düsseldorf, which aimed to improve safety and efficiency, both in terms of throughput and environment [37]. As part of the project, a central traffic manager has been designed for trucks in urban environments. This manager receives the states and destination of the trucks using V2I. With the data and a traffic model, optimization can be performed on potential signal timing adjustments to reduce the number of trucks stopping. V2I is also used for a decentralized adaptive traffic signal controller in [38]. It should be noted that decentralized is used here in the context of a central traffic manager: each intersection is managed by a central intersection manager, with there being no central traffic manager managing the intersection managers, similarly to the hybrid graph in Figure 2-3(c). The availability of V2I communication allows the intersection manager to have accurate vehicle data, which can be used to reduce the overall queue length. A similar use of V2I is to optimize the signal timings for a cost function directly based on fuel consumption [39].

2-7-3 Vehicle to Vehicle communication

Vehicle to Vehicle (V2V) communication includes all direct messages between vehicles without going through a middleman, e.g. a traffic manager. Because of the lack of a well defined central agent, when utilizing purely V2V communication, there is a requirement for decentralized solutions. It is possible to use V2V with a leader vehicle that acts as a centralized traffic manager, however that brings many complications, such as what happens at an intersection when either multiple or no leader vehicles are approaching at once; or deciding on the leader vehicle; or the vulnerability of a centralized controller.

The proposed uses for V2V are quite varied. An early simple use case is the use of V2V to send a emergency warning message in order to avoid rear-end collisions [40]. Another proposal is to use V2V communication to transfer video data in order to make visual obstructions partially transparent[7] [41]. A more relevant use case is using V2V communication in order to manage intersections, an early example is [19] where vehicle positions and intentions are used to resolve a passing order.

2-7-4 Vehicle to everything communication

Next to V2I and V2V there are also other specific communication types such as: Vehicle to Pedestrian (V2P), Vehicle to Device (V2D), Vehicle to Grid (V2G), etc. All of these belong to the Vehicle to Everything (V2X) umbrella term which includes all communication between a vehicle and any entity. In general all subtypes of V2X are assumed to use the same communication protocols. Two main protocols are Dedicated Short-Range Communication (DSRC), which is based on WLAN, and Cellular Vehicle to Everything (C-V2X), which is based on cellular networks.

[7] In a provided example the front camera video feed from a truck is projected onto the windshield of the vehicle following the truck, making the truck appear as a hollow rectangle tube.

2-7-5 Dedicated Short-Range Communication

DSRC is a type of communication protocol designed to facilitate V2X communication for short ranges. It was originally initialized by the FCC in 1999 when they allocated the 5.850-5.925 GHz band for use of the U.S. Department of Transportation's Intelligent Transportation Systems program [42]. Other countries and regions have also allocated bandwidths for DSRC, for example the European Union has reserved 5.875-5.905 GHz for DSRC use.

Because these standards are defined separately per region, there is no unified standard for DSRC communication. The two main standards are DSRC as used in the US and Intelligent Transport Systems-G5 (ITS-G5) in the EU. Both standards are based on the IEEE 802.11p standard, which is an amendment on the IEEE 802.11 standard, known for its use in Wi-Fi, in order to add Wireless Access in Vehicular Environments (WAVE), which is defined in IEEE P1609.12.

The key change made in IEEE 802.11p is the reduction in overhead found in the traditional Basic Service Set, which is in essence the set of connected agents. This reduction is achieved by allowing an agent to joining a set without the association or authentication process. Next to changes in protocol, the standard also sets performance requirements for receivers to reduce cross-channel interference. More information can be found in [43] and [44]. In order to avoid confusion from the various terms used, protocols based on IEEE 802.11p will be referred to as DSRC protocols.

The performance of the IEEE 802.11p standard has been studied in [45]. The study concluded, using simulations, that the standard allows for up to 2500m transfer range in open air; additionally, the traffic offered should remain below 1000 packets per seconds in order to keep the delay below 100ms, which is commonly used as a maximum delay for V2X communication.

2-7-6 Cellular Vehicle to Intersection

C-V2X is an upcoming type of communication protocol developed as part of the 3rd Generation Partnership Project as a competing standard for IEEE 802.11p using cellular technology. The main benefit of C-V2X systems is the ability to use traditional cellular infrastructure for Vehicle to Network (V2N) and V2I communication, whereas DSRC would require new infrastructure to be installed. Another difference is that C-V2X is synchronous whereas DSRC is not. C-V2X can also achieve more reliable performance at similar ranges due to having a retransmission mechanism and different resource multiplexing [46].

In a study on the two protocols in highway platooning scenarios, the authors concluded from simulations that the improved reliability of C-V2X allows for shorter inter-truck distances compared with DSRC [47]. It should be noted that C-V2X and DSRC are not necessarily exclusive and could be used simultaneously for redundancy or specific communication types.

2-7-7 Vehicle Messages Standards

Similarly to the various protocols there are various standards for the messages being sent over the protocols. As of now the two main message standards are the US Basic Safety Message (BSM) and the EU Cooperative Awareness Message (CAM), these two message types

are based on the SAE J2735 [48] and the ETSI - EN 302 637 - 2 [49] standards respectively. The two standards are not compatible at the moment of writing, however they do achieve the transmission of the same concepts. Since accounting for multiple standards is difficult and brings additional points of failures, it is desired to integrate the two standards into a single international standard. One example of such integration is the proposal to integrate emergency messages for V2X communication [50].

2-7-8 Regional and Industry Decisions

In April 2019 the European Commission has recommended that DSRC would be the standard in the EU for V2X communication [51]. Despite the recommendation by the EC the EU has rejected the proposal in June 2019 in favor of allowing both DSRC and C-V2X [52]. The US has not made a decision yet, however as of now the 5.9 GHz band is reserved for DSRC with C-V2X having no reserved bands. The vehicle industry has also been divided on the topic with Volkswagen, General Motors, and Toyota initially supporting DSRC, and Ford, Daimler and BMW supporting C-V2X [53]. It appears that despite the lack of consensus for a single standard there is still a common desire for a communication standard for V2X.

2-7-9 Hypothetical Generic Communication Standard

Due to the lack of consensus for a single standard, and because this thesis does not intend to analyze the communication problem and only uses it as a tool, we shall not specifically adhere to one of the introduced standards. Instead we assume a Hypothetical Generic Communication Standard (HGCS) which is based on the goals of the actual standards, however it imposes no requirements on conflict resolution solutions based on the specific communication standards' definitions. HGCS includes both a communication protocol and a message standard and has the following features:

1. Vehicles can automatically connect to nearby vehicles based on a defined distance, without any additional delay, and transmit information.
2. All connected vehicles send and receive information synchronously.
3. The protocol has safety features, which prevent malicious attacks.
4. The protocol is reliable, meaning that data will arrive without noise and packet loss will not lead to significant time delays.
5. The maximum range is $r_{max} = 250$ m.
6. Communication occurs at the frequency $f = 10$ Hz.

The maximum range and frequency are based on simulations performed in [54]. In the rest of the thesis, it is assumed that all conflict resolution protocols use this communication standard regardless of the communication protocols it was originally defined with. This allows for a fairer comparison between sub-solutions and is closer to reality; actual implementation will have to use the (inter)nationally defined communication standards instead of a proprietary one.

Chapter 3

Conflict Resolution

The conflict resolution problem can be formulated as a weighted optimization problem where the goal is to minimize the total cost function (J), which has a total delay component (J^D) and a total comfort component (J^C) and a weight (w) for each vehicle. The optimization is subject to safety and feasibility constraints where the vehicle states ($\boldsymbol{x}_i$) should be in their respective safe sets ($\mathcal{X}_i^s$) and feasible sets ($\mathcal{X}_i^f$). Also, $\boldsymbol{U}_i$ is the concatenated control input for vehicle i for all t, N is the total amount of vehicles, D_i is the destination of vehicle i, $\boldsymbol{X}$ is the trajectory of vehicle i. The resulting optimization problem is:

$$\begin{aligned} \min_{\boldsymbol{U_i},...,\boldsymbol{U_N}} \quad & \sum_{i=1}^{N} w_i \left(J_i^D(\boldsymbol{U}_i, D_i) + J_i^C(\boldsymbol{U}_i, ..., \boldsymbol{U}_N) \right) \\ s.t. \quad & \boldsymbol{X}_i \in \mathcal{X}_i^s, \boldsymbol{X}_i \in \mathcal{X}_i^f, \quad \forall i \in \{1, ..., N\} \end{aligned} \tag{3-1}$$

This chapter is separated into five sections, each focusing on a different aspect of conflict resolution, with the intention to provide a comprehensive, yet brief, understanding of the conflict resolution problem. First, Section 3-1 covers centralized conflict resolution approaches; second, the decentralized approaches are covered in Section 3-2; third, the concept of deadlocks and how to detect them is explained in Section 3-3; finally, the methods to resolve deadlocks are covered in Section 3-4.

3-1 Centralized Conflict Resolution

In this section, some centralized conflict resolution methods are shortly summarized. As centralized approaches do not face the issue of deadlocks, the conflict resolution might not fit well into the given subproblems. The methods can be separated into heuristic methods, game theory methods, and optimization based methods.

3-1-1 Heuristic Methods

Heuristic methods are commonly used in intersection management due to their good real-time performance and easy analysis, usually with a trade-off between optimality and generality. As even very simple heuristic methods are better than no conflict resolution methods, the performance gap between heuristic methods and optimization based methods is often acceptable. The most common form of centralized heuristic methods are resource reservation based methods. The main structure of a reservation based approach involves a central intersection manager which allocates resources to approaching vehicles. The resources can range from time slots to enter the intersection to specific trajectories to follow. The principle is that there is no conflict when there is no overlap in allocated resources. Other types of centralized heuristic methods do exist, such as a priority function based method, however these are mainly applied to decentralized conflict resolution. For that reason they will not be mentioned here, but are instead investigated in Section 3-2-1.

3-1-2 Game Theory Based Methods

Game theory based approaches have been proposed for centralized conflict resolution in [55] and [56]. In general, game theory has two categories of strategies, cooperative and competitive. The methods used for intersection conflict resolution generally use cooperative strategies as this leads to better results, e.g shorter delays and/or lower fuel consumption. Game theory based methods generally fall somewhere in between heuristic and optimization based methods. Generally the original problem is simplified to fit in a game theory framework, the simplified problem is then solved using game theory methods to achieve an optimal result.

The framework used in [55] models two distinct types of agents, one for the vehicles around the intersection and one for the central controller, each with their own goal. Information is assumed to be symmetric and communicated through some form of Vehicle to Everything (V2X) protocol. The vehicles are limited to three discrete actions: accelerating, decelerating or maintaining current velocity. The action for the central intersection manager is determining which vehicle agent to optimize at each time step.

The game begins with the central manager selecting a vehicle to be optimized, hereafter referred to as vehicle A. The potential actions vehicle A can take are then fed to all the other vehicles near the intersection. When a vehicle receives such a message it responds with the utility value corresponding to the action it will take. The action is assumed to be optimal for the receiving vehicle. By repeating this process the result of the game is a payoff matrix in which all potential action combinations are explored. The central manager then proceeds to inform the vehicles of which action they should take to minimize the overall utility function.

This method is not too dissimilar from the one used in [56], which models intersection management as a chicken game [57]. In a chicken game, players have the option to either yield or not yield, when only one player yields it loses, if both yield there is a tie, and if neither yield both lose. For an intersection, the loss for both vehicles not yielding is significantly larger than if either would yield; a tie however will lead to deadlocks, which should also be avoided. Based on the possible outcomes, a pay-off matrix is created for all possible action combinations of the players. In the paper an intersection is defined to have two players, one for the north-south lane and one for the east-west lane: each player decides all the actions for

the vehicles within their lanes. This representation however does not include the possibility for turning; to generalize it, the player should represent a group of vehicles that do not have conflicts with each other. This however might lead to issues as some vehicles could be part of multiple groups; the exact generalization for the players should be further investigated.

The central conflict resolution method requires a payoff matrix for all possible scenarios, similar to that of [55]. The matrix is then used to determine the *Nash equilibrium*[8], which is comparable with saddle points in the centralized optimization problem. By running a simulation for each scenario for the nearby vehicles, in combination with a reference payoff matrix, the payoff values for the entire matrix are determined. The reference payoff matrix contains the payoff for certain outcomes, with a large negative payoff if a collision is predicted. The *Nash equilibrium* is found when, for all vehicles, their payoff cannot be improved by unilaterally changing their actions. Deadlocks are avoided in this framework: suppose that the *Nash equilibrium* results in a deadlock with all vehicles yielding, any player could then increase its pay-off by not yielding, which in turn makes the deadlock not a *Nash equilibrium* by definition. It is however not clear how the algorithm acts in the case of a tie: it is likely that a tie breaking algorithm is necessary to arbitrarily choose the solution.

3-1-3 Optimization Based Methods

Optimization based methods attempt to solve the original problem or a dual version of it. These generally achieve the best results, however often have significant requirements in terms of computational power. Depending on the requirements, some optimization methods cannot be applied directly, this is due to the non-convexity of the main problem as described in Section 4-2-1.

3-2 Decentralized Conflict Resolution

In this section decentralized conflict resolution methods are investigated. Decentralized Conflict Resolution is applied when a conflict is detected, the obtained solution found should then be deadlock free. There are two ways to achieve this, by design or by resolving when they occur. Similar to centralized conflict resolution, the decentralized methods can be classified as heuristic methods, game theory based methods, and optimization based methods. In general the vast majority of approaches utilize a heuristic approach, due to the complexity of the game theory and optimization based approaches, which are hard to decentralize. Furthermore, heuristic methods have better real-time performance and are easier to analyze, with the drawback being the lack of optimality. As the primary concern is collision avoidance, this drawback is usually accepted.

3-2-1 Heuristic Methods

Heuristic methods attempt to achieve a good enough solution to the main problem through simplifications and assumptions. In a decentralized framework this generally means using a

[8]The Nash equilibrium is a term used in game theory to describe a solution at which no player can (individually) improve their position by unilaterally changing their strategy.

certain rule or simple priority function to determine when a vehicle can safely cross, or in what order to cross. The two types we shall consider are Resource Reservation based techniques and Priority Order based techniques. Resource reservation-based techniques resolve conflicts by requiring agents to reserve a certain resource, limiting the access to the resource to a single agent. The method to distribute the access is dependent on the implementation, in general, a first come first served method is used to claim access. Methods which apply a resource reservation-based technique include [16], which requires vehicles to reserve a resource called "tokens" in order to pass certain areas on the intersection, and [19, 20, 21], which propose message based methods to communicate which resource the vehicles want to reserve, with the additional use of a priority order to distribute the reservations.

Priority Order based techniques involve the invocation of a passing order through a priority based system. A lower priority agent must allow a higher priority agent to pass before the lower priority agent. The goal is to give each agent a unique priority such that each conflict has a solution based on the priority. One example of such a priority based approach is [58], where sequential optimization is used in the order of the priorities, setting the previously planned trajectories as hard constraints for the next agent.

Besides these two, there are also many separate rule based methods. One example is [59], where a rule set is found for an uncontrolled four-way intersection based on conventional traffic laws. Rule based methods are generally easy to analyze as the amount of configurations are limited; despite that, we shall not explicitly consider them; the reason being that they are generally specifically designed for a certain environment, and therefore not general enough. Furthermore, simple resource reservation or priority order methods can easily outperform rule based methods.

3-2-2 Game Theory Based Methods

Fully Decentralized Game Theory based methods are uncommon in the literature. This could be due to the difficulty to find an equilibrium in a decentralized system, or due to the field being relatively recent. There are however multiple methods that utilize game theory in some minor form; for example, [5] uses game theory to model interactions between vehicles, and [60] to design a vehicle controller based on the predicted behavior of other vehicles. It should be noted that [60] can be considered a decentralized game theory based conflict resolution method, however we shall not investigate it, as it does not provide any guarantees of conflict resolution. The experimental results have shown around 90 % success rate, which is not sufficient to consider it as proper conflict resolution.

The game theory based methods described in Section 3-1-2 do not seem to have decentralized variants yet, as these require a central manager to compute the Nash equilibrium. Potentially, the payoff matrix could be formed through communication, the payoffs could be found by each vehicle analyzing a portion of the matrix. If this can be done, it is then possible for the vehicles to communicate the lowest found overall costs/ highest overall payoff to resolve the conflict. When the payoff matrix cannot be formed an approach that requires a stable *Nash Equilibrium* can be used instead. If the *Nash Equilibrium* is stable, players will naturally converge to that equilibrium as the game progresses; however, playing the game is potentially dangerous, as unstable equilibrium or slow convergence can result in collisions. This approach

will require thorough research on the dynamics of the game in order to apply it to conflict resolution.

Another approach for decentralized game theory based conflict resolution involves a toll based system proposed in [61] and [62]. The method is considered from the perspective of drivers as players in the game, with the goal to determine a reasonable passing order based on the desire of the players. An auction is held in which players near the intersection can bid on passing priority, with the bidding performed by *automatic wallets* configured by the users. There are however multiple issues with this method: the main one is that there is no guarantee that any solution will be close to the solution of the original optimization problem in Equation (3-1). Furthermore, for this method to be considered a decentralized method it would have to perform the auctions through communication. This would rely heavily on secure payment verification to avoid malicious agents from sending fake bids. Additionally, this method is inherently advantageous for wealthy users, whilst it has minimal improvements in terms of total delay time or energy usage.

3-2-3 Optimization Based Methods

Optimization based methods have seen extensive interest through centralized intersection managers, which is explored in Section 3-1-3, however decentralized methods are still lacking. There are however a few methods that do perform some form of decentralized or distributed optimization, for example [58] can be considered to be a hybrid between heuristic and optimization, or [17, 63] in which navigation functions are used for optimal control. We however do not consider these methods to be fully decentralized optimization based conflict resolution, as they do not find the solution to the primal problem of Equation (3-1); instead, these methods use optimization for motion planning and control.

As the focus lies on finding a solution to the primal problem in a decentralized fashion, the use of distributed optimization using decomposition methods is a logical choice. This section will therefore be centered around possible distributed/decentralized optimization techniques which could be used to achieve decentralized conflict resolution.

3-3 Deadlock Detection

Deadlocks can occur in any distributed system and lead towards significant costs in terms of performance and safety. A deadlock occurs when two or more agents cannot proceed with their task due the required resource being held by other agents in the deadlock. This results in each agent being unable to continue until their resource becomes available. Without deadlock resolution strategies, this would continue indefinitely. It is therefore necessary for deadlocks to either be prevented or resolved, which in turn requires the ability to detect deadlocks.

This section begins with Section 3-3-1 that shortly describes the way deadlocks occur in conflict resolution and other systems. Two approaches for detecting deadlocks in a system are investigated, with a graph theory based approach being investigated in Section 3-3-2. Section 3-3-3 then discussed methods to detect deadlocks through cycle detection in a graph, and finally methods to model the system are discussed in Section 3-3-4.

Note that livelocks[9] are not explicitly defined or considered in this thesis as livelocks do not seem to occur in the physical environments investigated. Livelocks can however occur in the communication systems, however as we assume that communication is perfect we do not consider these livelocks.

3-3-1 Deadlock Occurrences

Deadlocks are a common issue when dealing with decentralized systems, regardless of the application. In computer science, deadlocks occur in computer networks or processes due to limited communication or computation resources, with initial research going back as far as 1971 [64, 65, 66, 67]. Other research on deadlocks includes surveys on distributed deadlock detection algorithms [68], and also traffic management [69, 70, 71, 21, 22, 72].

When considering deadlocks specifically for conflict resolution for autonomous vehicles, these usually occur due to unpredictable changes in the system. In this situation, deadlocks can occur at two levels, either high level due to the conflict resolution policy, or low level due to some lower level safety or physical constraints. The two cases are not exclusive, as a low level deadlock can cause or be caused by a high level deadlock in the conflict resolution policy. A common scenario at which this can occur is when four autonomous vehicles attempt to cross a four way intersection at once. If they do not have any conflict resolution, they will likely end up in two possible deadlocks. The first one is a high level deadlock where all vehicles are yielding for each other and no vehicle ends up crossing the intersection. The other deadlock occurs when all vehicles cross at once, ending up in a formation in which no vehicle can continue along its path without violating its safety constraint, resulting in a standstill. Both of these deadlocks lead to the same result, however lower level deadlocks are in general harder to detect. This is due to the blocking being harder to model when compared with yielding. It is therefore important that the modeling method is carefully considered. A proper method would be able to model both high level yielding and low level yielding, and include both of these when analyzing the entire system for deadlocks.

3-3-2 Wait-for/Yield graph

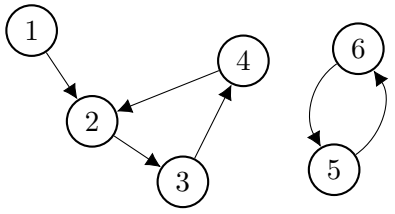


Figure 3-1: Example of a yield graph $\mathcal{Y}$ with two separate deadlocks. Arrows indicate yielding, e.g. node 1 is yielding for node 2. The two deadlocks involve the nodes {2,3,4} and {5,6} respectively

It is possible to model the distributed system in a graph in order to identify deadlocks, an example is shown in Figure 3-1. In the field of computer science such a graph is known as a

[9] A livelock can be seen as a higher order deadlock, with agents changing their states continuously without making any progress.

wait-for graph [73], in this report this type of graph will be referred to as a yield graph ($\mathcal{Y}$).

The graph $\mathcal{Y}$ has a set of nodes $\mathcal{N}$, which represent the individual agents in the system, and a set of edges $\mathcal{E}$, which represent the relations between the agents; an agent i yielding for an agent j is represented by a directed edge from i to j. Nodes and edges can be written as n_i and $e_{i,j} = (n_i, n_j)$ respectively. A cycle in the graph $\mathcal{Y}$ represents a deadlock, a graph without any deadlocks is therefore by definition acyclic.

Theorem 3-3.1. *A graph is deadlock free if it is acyclic.*

To prove this, the definition for a cycle and a deadlock has to be compared, in the context of a yield graph.

Definition 3-3.1 (Cycle)**.** A non-empty directed trail[10] where the first and final node are the same node and also the only repeated node.

Definition 3-3.2 (Deadlock)**.** A state of a multi agent system, where the agents in a deadlock cannot proceed with their task due to resources being held by other agents in the deadlock.

The definition for a deadlock can then be written in the context of a yield graph. In the yield graph ($\mathcal{Y}$), an agent is a node (N_i), and a directed edge represents an agent yielding. For an agent i, being unable to proceed due to other agents in the deadlock implies that i is yielding for another agent in the deadlock. As it cannot be yielding for itself, each node in the yield graph with a deadlock has directed edges pointing to another node in the same deadlock. Furthermore as an agent cannot be in a state of deadlock if its resource is held by an agent outside of the deadlock, there can be no directed edge to a node outside of the deadlock.

For a system to have no outwards edges, and for all nodes to have at least one edge pointing to another node in the same system, at least one node must have both an incoming edge and an outgoing edge. If this were not the case, there would have to be at least one node which does not have a directed edge to another node in the graph, making it not a deadlock by definition.

We have established that there is at least one node, e.g. N_k, in the deadlock which has both an edge going into it, originating from another node, e.g. N_j, in the deadlock, and an outgoing edge, going to another node, e.g. N_l, in the deadlock. Following the outgoing edge of this node (N_k), the following node (N_l) must also have a directed edge going into another node in the graph, e.g. N_j or N_m. As this logic is always true, and the deadlock is assumed to be of finite size, the chain of directed edges must end up at the initial node (N_k). After all, it cannot point towards a node outside of the deadlock, and the initial node is known to have an edge going into it. Therefore any system with a deadlock represented in the yield graph has a cycle by definition. An acyclic yield graph therefore cannot contain a deadlock, it is thereby deadlock-free.

From the equivalence of a cycle in the yield graph, and a deadlock in a system, two methods can be used to deal with deadlocks. The first is prevention by designing conflict resolution algorithms to never cause cycles, the second method is by using algorithms to resolve deadlocks when they come up. Regardless of the method, both require some form of cycle detection in a graph.

[10] A directed trail is a sequence of nodes and edges with all edges being unique.

3-3-3 Cycle Detection

Due to the equivalence of deadlocks in a system and cycles in a yield graph, as proven in Section 3-3-2, it is necessary to detect these cycles. Despite the clear definition of a cycle it can still be difficult to find these efficiently. This is because a cycle is in essence a permutation of nodes, analyzing each potential cycle therefore has the time complexity $\mathcal{O}(n!)$. For actual intersections, this is likely less of an issue, due to the amount of vehicles being relatively small, and the graph being sparse, i.e. most vehicles will only yield to a few of the vehicles at the intersection.

3-3-3-1 Centralized Cycle Detection

One method for detecting cycles is simply by analyzing each trail in the system. When performed in a centralized fashion, the algorithm can first eliminate all nodes that do not have both an incoming edge and an outgoing edge, as these nodes do not actually contribute to the state of deadlock. Afterwards the algorithm can pick a random edge and follow its trail, branching into multiple trails if a node has multiple outgoing edges. Once a trail arrives at a node that is included in its trail, a cycle is found. Once all the trails and all edges have been fully explored, all the cycles of the system should be detected, provided that the graph is the entire graph of the system. Depending on the implementation, this can be done using a Depth-First Search (DFS) algorithm or a Breadth-First Search (BFS) algorithm,[11] with preference going to a DFS algorithm due it it being able to detect cycles earlier [74]. In reality, the exact search method used is likely going to have little impact on the overall performance due to the graph being sparse and small for a single intersection

3-3-3-2 Decentralized Cycle Detection

In a decentralized system, it is not possible to assume that each individual agent has access to global information without communication. Because of the lack of information, the method in Section 3-3-3-1 cannot be applied directly. When cycle detection has to be performed in a decentralized fashion, there are two options. The first is that all agents communicate their edges propagating the graph information to all its connected agents, so all agents will end up with the full graph, each being able to analyze if they are part of a cycle, similar to the centralized method. The difference is that they will only analyze trails involving themselves, provided that the communication graph is connected. Whilst this distributes the computational burden among the agents, it is not very efficient, as each agent in a cycle will detect the same cycle. This means that the total amount of computations will likely be significantly higher than in the centralized method. Furthermore, depending on the graph size and the communication rate, it can take a long time to form the graph.

Another method involves each agent sending a unique code along its outgoing edges, with the request that any recipients do the same. The message will then propagate along the trails,

[11] Depth-First Search and Breadth-First Search are two algorithms that can be used to explore graphs or trees. As the name suggests Depth-First Search aims to fully explore each branch before moving onto the next one. Breadth-First Search is the opposite where all branches of the same depth are explored before moving onto the next depth level.

and either end up in a dead end or back to the original sender. If the sender receives its own unique code, it can conclude that a cycle exists and the agent is part of it. Furthermore alongside the unique code each agent can attach their identifier such that the trail of the cycle is known. Unique code propagation does however have the same problem as the previous methods, due to the requirement of an additional communication step for each additional node in the trail.

One example of a message based cycle detection algorithm can be found in [75]. The algorithm is based on a node sending out a message along its edges; the recipient then decides upon the action based on previous messages and its state. Nodes can either forward the message or respond to the sender or initial sender, where responses inform other nodes about their status. When the initial node has received all its expected messages, it can evaluate the deadlock. As this algorithm detects deadlocks for a certain initial node, a requirement is needed to decide when the algorithm is executed, e.g. whenever resources are blocked for a certain amount of time. Each execution of the algorithm has a worst case time complexity of $\mathcal{O}(2d)$ and a message complexity[12] of $\mathcal{O}(2e)$, where d is the diameter[13] of the graph, and e is the amount of edges in the graph. Other variations of this approach include [76, 77, 78], with the variations being mostly in terms of improved scalability or reductions in message redundancy.

In terms of working with local information, the effects of having incomplete information on the detection of relevant deadlocks is not well known. More research is needed on the use of local yield graphs, and whether the incomplete information causes any fundamental issues.

3-3-4 System Modeling

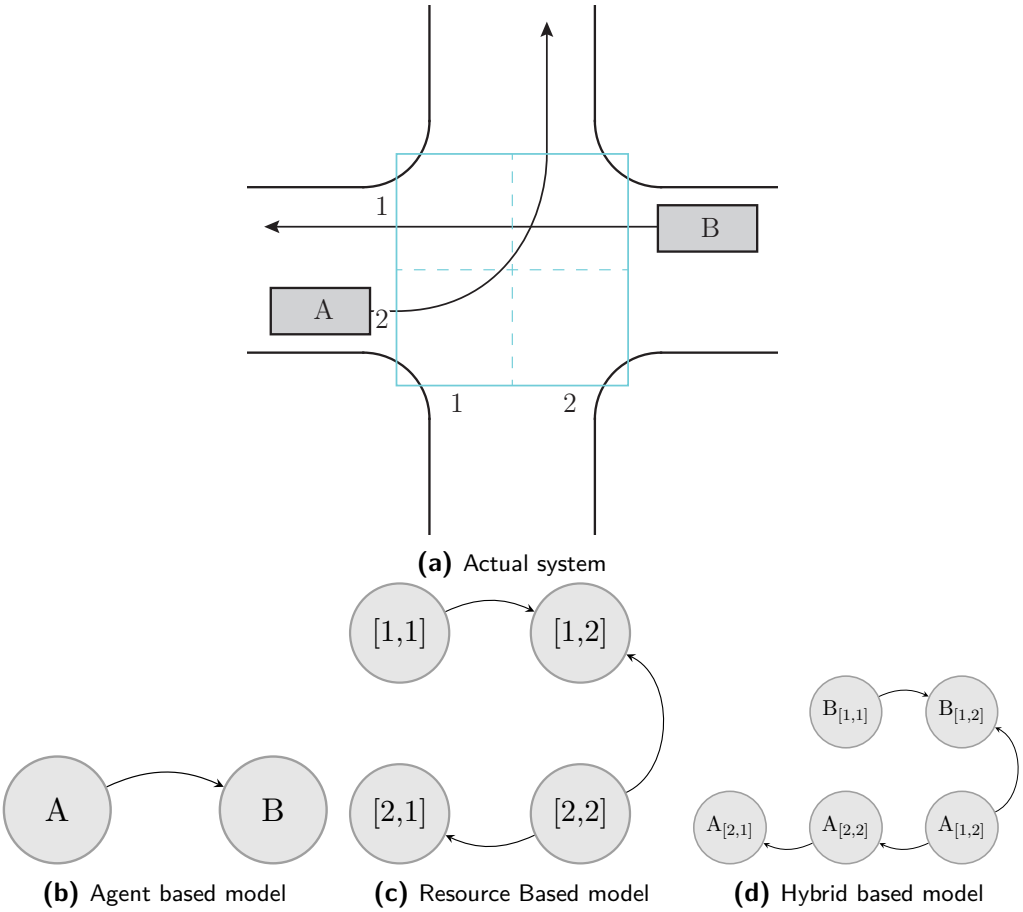
Whilst the detection of deadlocks through detecting cycles in yield graphs is effective, that method is not always directly applicable. For the case of conflict resolution for autonomous vehicles, the high-level deadlocks, as described in Section 3-3-1, can be detected effectively, as the yield graph can be constructed with ease. This is however not the case for the low-level deadlocks, where the edges represent another agent blocking their path. Finding the exact graph is difficult, especially if vehicles are allowed to stray from their path. For this reason it is not uncommon for conflict resolution algorithms to disallow vehicles to change their priority order after entering the intersection, as yielding could result in forming a low-level deadlock. It is therefore important to carefully consider how the system is modeled in the yield graph. Examples of the different system modeling methods are depicted in Figure 3-2.

3-3-4-1 Agent based modeling

Agent based modeling is the standard modeling method used for yield graphs. Each agent is modeled as an individual node and their relations are modeled as the edges. This modeling method is commonly used for computer networks and operating system process management. The simple agent based modeling method is often sufficient for these as deadlocks are often simple in nature, where deadlocks can be resolved by killing a process or forcing a process

[12] The message complexity is similar to time complexity but measures the number messages instead of time.

[13] The diameter of a graph or network is the longest path between any two vertices, when only considering the shortest paths.



(a) Actual system

(b) Agent based model **(c)** Resource Based model **(d)** Hybrid based model

Figure 3-2: Examples of different system modeling methods for the scenario depicted in Figure 3-2(a), where the directed edges in the graph indicate a node yielding. Figure 3-2(b) shows the agent based modeling method, where the nodes model the vehicles; Figure 3-2(c) shows the resource based modeling method, where the nodes model the intersection regions; Figure 3-2(d) shows the hybrid based modeling method, where the nodes model the vehicles with their corresponding intersection regions; see Section 3-3-4-1, Section 3-3-4-2, and Section 3-3-4-3 respectively.

to release some resource. An example of an agent based modeling approach is shown in Figure 3-2(b).

For intersection management a node would represent a vehicle, with the edge representing a vehicle yielding to another vehicle. This can be sufficient if the edges are modeled conservatively enough. An edge should be constructed if a vehicle has to yield for another vehicle, furthermore the yielding vehicle should not be able to enter any conflict zone if it is currently yielding. The modeling of the edges therefore depends on the conflict detection method used in the protocol. When using a spatial conflict detection method, there should be no issues with deadlocks, as that will detect the conflict for any conflict zones the vehicle crosses in its path.

The issue arises when the conflict detection is not conservative enough, for example by using a spatio-temporal conflict detection method. The conflict detection method would not detect a conflict with vehicles crossing the same area before or after the ego vehicle. Because of that the yield graph would not contain any edges between these, resulting in a potentially missed

cycle and improper deadlock detection.

3-3-4-2 Resource based modeling

Resource based modeling models each resource as its own node, with edges modeling relations between the resources. This modeling method is suitable for detecting systematic deadlocks due to resource usage order. For example, if a certain process P requires a resource R_A, which (indirectly) requires the same resource R_A, to continue, a cycle would form in the graph. Such a graph does not take into account the agents directly in its nodes, however their processes can be modeled as a chain of edges in the yield graph. Figure 3-2(c) shows an example of a yield graph modeled using a resource based modeling method.

In the context of intersection management this method can be used to model a conflict region, for example by overlaying a grid on top of the intersection region. The advantage of using resource based modeling is that the method can detect deadlocks that are missed by the agent based method. When the trajectories of the vehicles are known, each trajectory can be represented as a series of nodes yielding for each other. These nodes represent a certain resource, for example when using a grid to divide the regions, a node can represent a certain cell. The edges are then generated based on the trajectory of the vehicles, with each node yielding to the next node in the graph. When using this method, a resource is best considered in the spatio-temporal domain, as a purely spatial domain can lead to over detection of deadlocks. For example when two vehicles in opposite lanes both desire to turn left, if they go around each other their paths will form a cycle in the graph, whilst this is not an issue in reality the vehicles likely cannot occupy the nodes at once.

3-3-4-3 Hybrid based modeling

Hybrid modeling combines the agent based and the resource based modeling, considering both the agent and the resource they require. As this method is a hybrid of the agent based and resource based method it is capable of detecting both deadlocks in the vehicle policies and deadlocks in the system dynamics. It is also possible to achieve the same result by utilizing agent based and resource based graphs independently. One possible implementation of a hybrid based modeling method is shown in Figure 3-2(d).

The method proposed by Y.-T. Lin *et al.* in [72] constructs the nodes based on the agent and the regions it passes through, allowing for the yield graph to account for the physical deadlocks. The hybrid graph is referred to as a *resource conflict graph* in [72], however we shall refer to it as a hybrid graph to distinguish it from the purely resource based graph. The nodes in the hybrid graph are formed based on the agent, and the regions it traverses. In [72] the hybrid graph is constructed from a *timing conflict graph*, which contains nodes of the form $n_{i,a}$, where i represents the agent and a the resource. Furthermore, the *timing conflict graph* defines the following three types of edges:

- Type 1 edges represent dependency based on the vehicle trajectory, an edge $(n_{i,a}, n_{i,b})$ therefore indicates that $n_{i,a}$ has to be traversed before $n_{i,b}$.
- Type 2 edges represent dependency based on identical source lanes, an edge $(n_{i,a}, n_{i,b})$ therefore indicates that $n_{i,a}$ has to be traversed before $n_{i,b}$.

- Type 3 edges are formed bidirectionally to represent conflicts in paths, edges $(n_{i,a}, n_{j,a})$ and $(n_{j,a}, n_{i,a})$ therefore indicate that vehicle i and j have a common resource a in their paths.

Type 1 and 2 edges can be seen as rigid edges, where the dependency they represent cannot be changed by the system without physical changes. Type 3 edges on the other hand are flexible and should be resolved to achieve conflict resolution.

In the hybrid graph a node $n_{i,a,b}$ is formed for each edge $(n_{i,a}, n_{i,b})$ of type 1; additionally edges $(n_{i,a}, n_{i,b})$ and $(n_{i,b}, n_{i,c})$ are combined into one edge $(n_{i,a,b}, n_{i,b,c})$. Furthermore, any other edges between nodes are inherited by the combined nodes in the hybrid graph, with specific rules being explained in Section 3.1 in [72]. The main reason for constructing the hybrid graph from the *timing conflict graph* is that the hybrid graph is both more compact and fully representative. Fully representative implying that cycles in the graph are 1:1 to deadlocks in the physical system.

Whilst the hybrid graph in [72] was constructed from a *timing conflict graph*, it should also be possible to construct such a graph directly. Each node in the hybrid graph in essence represents a trajectory segment for a given agent, with edges representing some dependency, either due to their spatio-temporal dependence in a trajectory, or due to conflict resolution policy. Furthermore, Y.-T. Lin *et al.* construct edges based on the order at which regions are crossed, which is a discrete approximation of the physical system. More research should be conducted on more direct construction methods and the possibility for exact modeling of spatio-temporal trajectory dependency. It is also worth investigating the effects of kinematic and dynamic constraints on the representativeness of the hybrid graph.

3-4 Deadlock Resolution

3-4-1 Centralized Deadlock Resolution

Deadlock resolution is often not considered in centralized methods, as deadlocks can usually be easily prevented. This is often accomplished by limiting the amount of concurrent vehicles at a conflict region to one, avoiding any low level deadlocks. These restrictions come at the cost of performance, and should ideally be avoided. As many centralized methods approach the problem as some form of scheduling problem, there is potential for deadlocks when the schedule is not consistent. For example when a policy is switched whilst a vehicle is crossing, with the new policy having a different order of passing, some vehicles can end up blocking vehicles preceding them in the new schedule. This could be simply avoided if the passing order in the schedule was forced to be consistent, however that will also reduce optimality. The other solution would be taking possible deadlocks into account, which would allow a change in schedule provided the change does not cause deadlocks.

There can be two approaches towards dealing with deadlocks, either by prevention, or by resolution. They are generally similar in that they both require deadlock detection, however prevention rejects solutions if they cause deadlocks, whereas resolution resolves[14] deadlocks for any solution. Deadlock prevention in general can be done by designing a policy that

[14] Resolving a deadlock requires that the new solution no longer has any deadlocks.

cannot cause any deadlocks, such as forcing consistent schedules, or by preemptively applying the detection methods explained in Section 3-3. It should be noted that deadlock prevention methods inherently implies that avoiding deadlocks is more important than the optimality of the solution: it is possible that the global optimum cannot be obtained due to the deadlock prevention mechanism.

Deadlock resolution generally involves analyzing which part of a system is involved in the deadlock. The deadlock can then be resolved by changing the system so that the deadlock is no longer in the system, for example by removing an edge in a yield graph. One example of a resolution method can be found in [70], where deadlocks for a unified transport system are investigated. The used resolution method is a heuristic method where the deadlock is resolved by changing the structure of the yield graph. The method identifies which vehicles are causing the deadlock, and investigates whether any of these vehicles can change their routes. New routes are then computed for these vehicles, removing the deadlock from the system. A method similar to this could also be applied to conflict resolution for autonomous vehicles, where vehicles can be rerouted or forced to yield in order to resolve deadlocks. Another similar method can be found in [79] where the direction of the edges between nodes is swapped in order to resolve deadlocks, and the graph is shrank by eliminating the nodes that cannot be in a cycle. As these methods generally do not consider the effects of the changes made, they are not optimal. One way to improve this would be to determine the impact of the changes on the overall system; however, this would come at the cost of increased computation time.

Optimal deadlock resolution has been explored in [80] for multidatabase systems[15]. The proposed method can be summarized as finding which subsets of nodes in a yield graph need to be terminated in order for the graph to be acyclical, after which the cost of terminating is evaluated for each subset, and finally the lowest cost subset is eliminated. As it is not possible to simply terminate a vehicle, this method cannot directly be applied to our problem. Another potential method has been mentioned by Lin *et al.* in [72], stating that the problem of finding a minimum spanning tree can be applied to deadlock resolution. They suggest the use of Kruskal's algorithm to select edges based on the lowest cost whilst avoiding cycles with the already formed edges [81]. The algorithm involves trimming down the graph to the relevant vehicles, e.g. removing vehicles that are strictly followers of the leader vehicle[16]; afterwards, the edge costs for the remaining nodes is computed based on the delay caused if the edges are not removed. The algorithm then iterates through the edges in descending cost order and determines whether the removal or reversal thereof will resolve the deadlock. This is repeated until the system is deadlock free and optimally resolved, provided that the delay is computed exactly.

3-4-2 Decentralized Deadlock Resolution

Deadlock resolution for decentralized conflict resolution is similar to a centralized deadlock resolution, covered in Section 3-4-1, with the major difference being that decentralized methods lack a central resolution policy. In general there are three possible approaches to achieving

[15] A network of database systems that integrates multiple autonomous local databases into a single accessible database

[16] The leader vehicle would be the vehicle closest to the intersection in a lane, with the vehicles behind it being the followers.

decentralized deadlock resolution: the first involves the construction of the global yield graph through communication, the second involves a constructing a local yield graph, and the last involves a message propagation based method.

As deadlocks have been generalized in Section 3-3, the way deadlocks are relevant in decentralized conflict resolution is similar to centralized conflict resolution, provided that decentralized conflict resolution is solving the problem in Equation (3-1). When this is not the case, for example when utilizing purely local MPC or conflict resolution for a shorter horizon, deadlocks can appear in unobserved areas. When considering distributed optimization deadlocks can furthermore be caused by the infeasible run-time solutions, in addition to the deadlocks occurring in centralized optimization. It is therefore important that the system model used for deadlock detection is able to capture all these elements. Potentially, it would also be possible to integrate deadlock conditions into the optimization constraints; however, that has not been explored yet.

A method involving a full yield graph will work similarly to a centralized method, as each agent will have the full graph available, which is sufficient for centralized methods. This method therefore utilizes centralized deadlock detection, which can be performed separately at each agent. When the full graph is available, agents can also opt for using a centralized deadlock resolution method as described in Section 3-4-1, where each agent would perform the same calculations, with the potential to distribute the local calculations among agents to reduce total computational costs. The full graph could also be used in a heuristic method, where each agent consults the graph to decide whether their new actions will cause deadlocks or not, which would be local deadlock prevention.

The case where each agent only has the graph involving its direct neighbors or another possible partial graph is not very effective. This is due to the possibility of deadlocks occurring in cycles not mapped in any partial graph. A local/partial graph would have to be proven to contain sufficient information about all cycles in the full graph in order to be used for deadlock detection/resolution. One way to achieve this goal would be to trim down a full graph, which would still require the construction of a full graph, or the construction of the partial graph based on message propagation. However, there could be an advantage over a full graph if a search algorithm is used frequently on the partial graph, having a reduction computational costs due to lesser nodes. An approach similar to this can be found in [79] where nodes with zero in-degree or out-degree are removed from the graph.

Deadlock resolution using messages could involve prevention by detecting the cycles and rejecting a change in strategy, or they could be used to identify which agent should change in order to resolve a deadlock. The method to determine which agent should change their strategy can be similar to methods described in Section 3-3. One example of a heuristic message propagation based method is [75], where the *victim* is determined based on the amount of predecessors. A *victim* in this context is the node that has to deviate from its strategy in order to resolve the deadlock. Another example can be found in [82] where the focus is put on taking action as fast as possible; however, as this method is intended for computer processes, there is no regard for the optimality, in terms of conflict resolution, of the solution provided by the deadlock resolution mechanism.

Chapter 4

Problem Statement

In this chapter the decentralized conflict resolution for autonomous vehicles problem is introduced. First, the environment of the problem is defined and its relation to the possible solutions is analyzed in Section 4-1. Afterwards, the objective for the problem is defined in Section 4-2.

4-1 Environment

The environment in which the problem is to be solved has a significant effect on its possible solutions. Current environment definitions in the literature vary from paper to paper ranging from very specific to very vague. For example [83] and [84] define a *semi-structured environment*, where "there is natural topological graph structure (which may or may not be known to the robot *a priori*), but maneuvers off the graph are valid and, in fact, quite frequent" [83]. As examples, they mention parking lots, garages, construction zones, and the areas around shopping centers. *Semi-structured environments* are in contrast to *highly structured environments*, which require a topological graph imposed on the environment with limited deviations allowed, and *unstructured environments* where any path is possible. These definitions for the different types of environments are not ideal, as they rely on some subjective parameters.

Another example is the environment used in [85] for cooperative decentralized conflict resolution. In this case, the environment can be chosen freely; however, the existence of deadlocks cannot be avoided depending on the obstacles in the environment. A completely unrestricted environment would therefore not be suitable for conflict resolution. Most environments used for decentralized conflict resolution, for example in [86, 22, 19, 17], limit their environments to specific conventional intersections.

The goal is to be able to provide general definitions, which are structured like in Figure 4-1, such that each environment type is coupled with its subproblems. This goal requires a definition that can accurately describe the environment based on the problem that has to be solved.

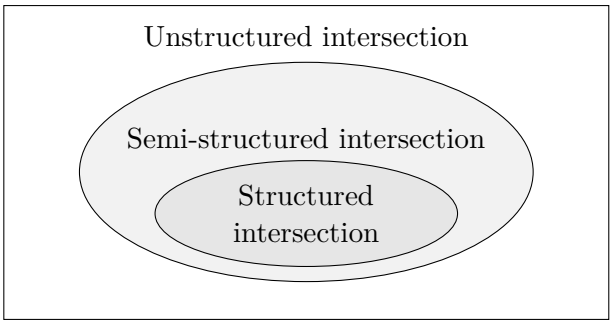


Figure 4-1: Varying levels of structure in environments, semi-structured environments have looser requirements than structured environments

We shall use the concept of homotopy in order to describe the vehicle trajectories in a general environment. Homotopy serves as a general mathematical foundation for the definitions we want to achieve. The use of homotopy for paths is not new and has been used before in for example in path planning [87, 88, 89, 90, 91, 92]. These references mostly focus on using or finding homotopy classes for motion planning, while we instead intend to use the definition of a homotopy class to describe an intersection environment.

4-1-1 Definitions

In order to provide a mathematical definition, first we shall introduce the concept of homotopy as follows:

Definition 4-1.1 (Homotopic)**.** A continuous function f is homotopic with a continuous function g when f can be continuously deformed into g.

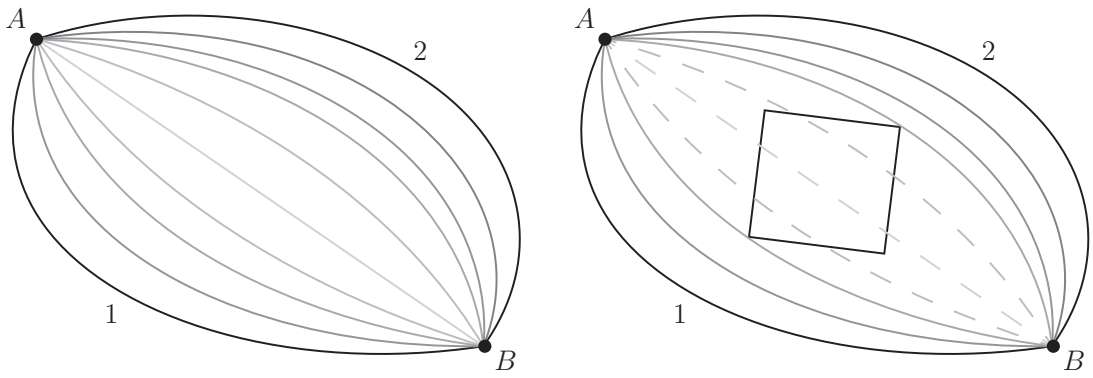
Definition 4-1.2 (Homotopy Class)**.** A homotopy class $\mathcal{H}$ contains all trajectories that are homotopic with each other.

Figure 4-2 showcases how adding an obstacle causes the homotopy class to split in two. Homotopy classes are defined based on their endpoints and their path around the obstacles, for example $\mathcal{H}^i_{(A,B)}$ indicates the ith homotopy class with endpoints A and B. Because a homotopy class is not directional, $\mathcal{H}^i_{(A,B)}$ is equivalent to $\mathcal{H}^i_{(B,A)}$. Following this we attempt to provide a mathematical definition of a (semi-)structured intersection as follows:

Definition 4-1.3 (Semi-structured intersection environment)**.** If an environment has clearly defined inputs and outputs, and there is at least one homotopy class for each input-output pair, which intersects with another homotopy class from another input, then the environment is a semi-structured intersection environment.[17]

Definition 4-1.4 (Structured intersection environment)**.** If an environment has clearly defined inputs and outputs, **with each input-output pair having at most one homotopy class**, and there is at least one homotopy class for each input-output pair, which intersects with another homotopy class from another input, then the environment is a structured intersection environment.[18]

[17] See Figure 4-4 for examples of semi-structured intersection environments.
[18] See Figure 4-3 for examples of structured intersection environments.



(a) One homotopy class **(b)** Two homotopy classes

Figure 4-2: Figure 4-2(a) shows how path 1 can be continuously deformed into path 2, hence there is only one homotopy class. By adding an obstacle in Figure 4-2(b), path 1 can no longer be continuously deformed into path 2: the dashed lines are now in conflict with the obstacle. The result is that there are now two homotopy classes, one above and one under the obstacle.

These two definitions can have similar physical environments, however a purely semi-structured intersection has the added difficulty of additional options in terms of which homotopy class is used. Finding an optimal trajectory would generally require the trajectory information from other vehicles, in a decentralized system this could induce some form of oscillating decision behavior. This issue could also occur in a structured intersection if a homotopy class is wide enough and overtaking is allowed. The potential for the oscillating behavior to occur is intrinsic to certain decentralized systems in which agent decisions are coupled, with the issue often worsened by delays. An example of this type of behavior can also be seen in humans, for example, when two people alternate between moving to the left and right in the attempt to avoid bumping into each other.

4-1-2 Assumptions

To improve the use of the category of a (semi-)structured intersection some assumptions have to be made. The goal of these assumptions is to allow the homotopy classes to somewhat represent a conventional lane. It is also important that the environment bounds to be evaluated are selected sensibly, as that can impact the category of the environment. With that in mind, the following assumptions are made:

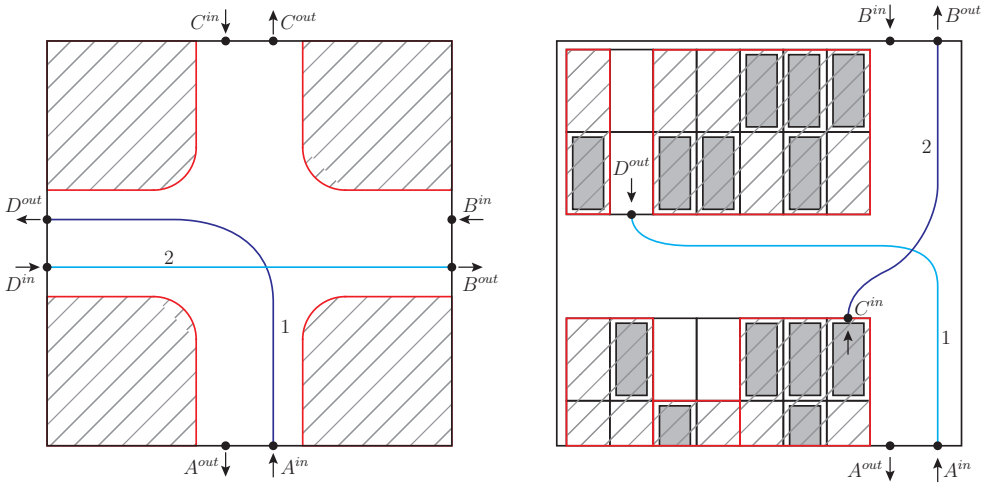
1. Homotopy classes are inflated to account for the physical size of the allowed vehicles, if the inflated class violates any constraints the class is no longer considered to be valid.
2. Traffic rules are modeled as hard constraints for the homotopy classes.
3. Homotopy classes from the same input do not intersect each other after diverging.
4. Vehicles sharing a common path do not overtake each other.

The first assumption is needed to ensure no intersections are identified due to an intersecting homotopy class that cannot be physically traversed. This assumption also ensures that an intersection between vehicles is identified even when their paths do not intersect, whilst the vehicles would collide when actually traversed. The second assumption is to avoid classifying an environment as a (semi-)structured intersection when the path intersection requires an

illegal maneuver; for example performing a u turn on a one-way road. As the definitions do not consider intersections with homotopy classes originating from the same input, it is important that those classes do not intersect with each other after their paths diverge. This in combination with the no-overtaking assumption, ensures that there are no unaccounted intersections.

4-1-3 Examples

For clarity some examples of common traffic scenarios are given and categorized based on the definitions and assumptions made in Sections 4-1-1 and 4-1-2 respectively. It should be noted that this literature review does not cover a method for identifying all homotopy classes and their intersections.

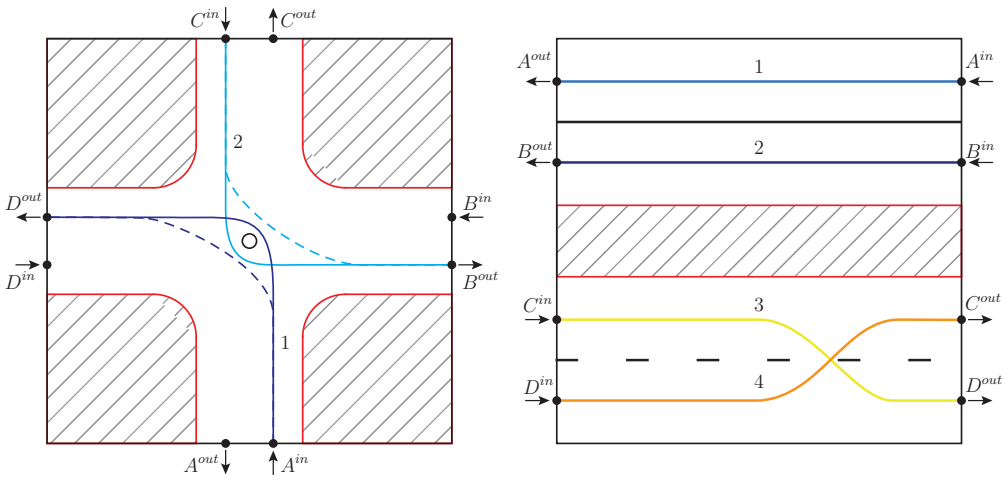


(a) Intersection **(b)** Parking lot

Figure 4-3: Two examples of structured intersections. The inputs of the environment are indicated by capital letters, obstacles are indicated by a hatch pattern. The classical unmanaged intersection in Figure 4-3(a) is an example of a structured intersection: each input-output pair has only one homotopy class and there exists at least one intersection between each homotopy classes, for example the shown paths (A^{in} to D^{out}) and (D^{in} to B^{out}). However, this does rely on the assumption that the homotopy classes are constrained by the lane directions. The parking lot in Figure 4-3(b) is also a structured intersection; this relies on the assumption that overtaking is only allowed from one side, preventing the homotopy class from splitting into two.

Figure 4-3 shows two real life traffic environments, which can be categorized as structured intersections. For the intersection depicted in Figure 4-3(a), it follows from the properties of a conventional unmanaged intersection that the environment is a structured intersection: the inputs and outputs are clearly defined by the in- and outgoing lanes, each input-output pair has at most one homotopy class due to the physical environment, and there is at least one homotopy class intersection for each input-output pair. The parking lot example, depicted in Figure 4-3(b), requires a few additional steps before it can be classified as a structured intersection environment. In this case the parked vehicles are modeled as obstacles which includes some empty spaces; this is not always necessary, however, in some cases the environment can become semi-structured due to a new possible pathway through empty parking lots. Additionally,

when a car intends to leave or park, an input or output should be added, achieving clearly defined input and outputs.



(a) Special Intersection **(b)** Highway

Figure 4-4: Purely semi-structured intersections are generally more complicated than structured ones. This is mainly due to the fact that some input-output pairs have multiple paths to choose from, which can result in different intersections. An example of such an environment is an intersection with an obstacle in the middle of the intersection, like in Figure 4-4(a). Another special case is when dealing with parallel roads, in Figure 4-4(b) it can be seen that path 1 and 2 are not classified as an intersection as expected. When lane changes are allowed the entire stretch of road is now suddenly classified as a (semi)-structured intersection depending on the passing rules.

Figure 4-4 shows two special environments, the first is an intersection with an obstacle in the center. The added obstacles turn the structured intersection into a semi-structured intersection. Because of that, the problem becomes more complicated as vehicles now have the choice between avoiding the obstacle clockwise or counterclockwise. The highway example showcases that allowing lane changes in parallel roads is also classified as a structured intersection, while, when overtaking is allowed on both sides of a vehicle, a semi-structured intersection is obtained. Solutions for semi-structured intersections can therefore also be applied to lane changes, however this might be less effective than solutions designed purely for lane changes.

4-2 Problem Objective

With the environment of the problem clearly defined in Section 4-1, the objective of conflict resolution should also be defined accordingly. The goal is to design a protocol, which will result in improved traffic throughput, without compromising safety and comfort. This problem can be solved using either centralized or decentralized approaches. Centralized solutions, however, rely on infrastructure that is often not available in rural roads, suburbs, parking lots or other traffic environments. Furthermore, relying on a central manager means that an intersection has a single point of failure, where failure results in the entire intersection becoming impassable.

Decentralized solutions are however difficult to optimize due to the possibility of deadlocks and the difficulty of distributing optimization.

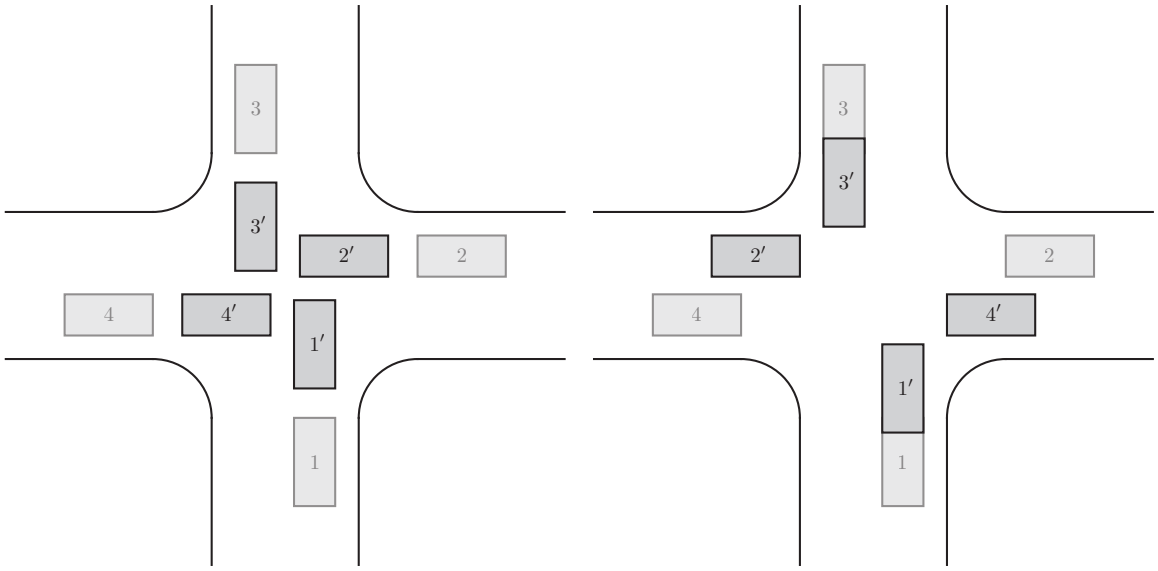


Figure 4-5: The blocks represent vehicles, the lighter blocks of the same number are their initial positions. On the left is an example of a deadlock occurring for an unmanaged intersection with four decentralized agents adopting the same strategy. The deadlock occurs as each vehicle will slowly move forward or stop to wait for the others to pass, in the case that all vehicles are waiting, a standstill will occur. On the right is a possible solution that avoids a deadlock by having vehicles 1 and 3 slow down to yield and thereby improves the traffic flow.

Deadlocks can occur under decentralized conditions if all controllers have the same strategy and the vehicles have the same relative scenarios. For example, in Figure 4-5 four vehicles are approaching an intersection at the same time, when all vehicles adopt the same strategy without any deadlock resolution, they might slowly approach the intersection until they reach a deadlock. Such behavior can be caused by strategies that err on the side of caution: preferring to avoid collisions by yielding instead of accelerating. With a centralized controller, or a deadlock free decentralized controller, for the same scenario the second solution, depicted on the right side of Figure 4-5, might come ahead where vehicles 1 and 3 yield for vehicles 2 and 4, preventing a deadlock, and thereby improving the solution.

4-2-1 System Objective

The conflict resolution problem has been introduced in Equation (3-1). Solving this optimization is difficult for multiple reasons. The constraints depend on the states of the other vehicles, and the overall problem is nonlinear and nonconvex, preventing the use of (for example) CVX[19] or other fast solvers. The optimization should also be solved very quickly (<100ms), making a central optimization based approach unpractical as the problem does not scale well.

[19]CVX is a widely used Matlab-based modeling system for solving convex problems

Due to the coupling in the constraints and objective function, primal decomposition cannot be applied to Equation (3-1). The problem is also shown to be NP-hard in [93].

If we assume that two intersecting homotopy classes only allow one vehicle to cross at once, then the problem can be seen as a combination of a higher level scheduling problem and a lower level trajectory optimization. This is similar to the approach used by [94], which decomposes the problem into an upper time-slot allocation problem and a lower level control policy problem. Another solution is the use of heuristic approaches: a few examples are the First Come First Serve (FCFS) approach [12], the Look-ahead Intersection Control Policy (LICP) approach, which takes into account potential delays [13], and the game-theory-based approach in [56].

However, decomposing the problem into a scheduling and lower level trajectory optimization essentially reduces the maximum efficiency of the problem. After all, both the decomposition method and the heuristic methods do not allow vehicles to change their paths, i.e. trajectory optimization is reduced to speed profile optimization. It is easy to imagine cases where this leads to reduced performance, e.g., if a vehicle cannot adjust its path, it is unable to overtake a slower traveling vehicle, despite having a higher reference velocity, leading to longer overall travel times.

The goal of this thesis is therefore to achieve decentralized conflict resolution, without compromising on the optimal motion planning aspect.

Part II

ADMM, variations, and OA-ADMM

This part contains an in depth overview of Alternating Direction Method of Multipliers (ADMM) and its variations. The variations covered include conventional ADMM in Chapter 5, adaptive ADMM in Chapter 6, online ADMM in Chapter 7, our finally our proposed Online Adaptive ADMM in Chapter 8.

Chapter 5 introduces the Alternating Direction Method of Multipliers, covers its convergence results and provides proof for the convergence. Chapter 6 covers the role of the penalty parameter in ADMM, and variants which utilize an adaptive penalty parameter. Chapter 7 then briefly summarizes various methods which utilize ADMM for online systems. Finally, our novel ADMM variant is proposed in Chapter 8, where it is covered in depth, similarly to how ADMM is covered in Chapter 5, providing convergence results and requirements.

Chapter 5

Alternating Direction Method of Multipliers (ADMM)

Alternating Direction Method of Multipliers (ADMM) is a distributed optimization method that utilizes an augmented Lagrangian in order to achieve better converge properties similar to the non-separable *method of multipliers* method. ADMM requires relatively mild assumptions in order to achieve convergence, and good results can be achieved with few iterations. The method was reviewed in the context of distributed optimization by Boyd *et al.* in [95].

5-1 Problem Formulation

For a problem of the form

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{aligned} \tag{5-1}$$

For a problem of the form given in Equation (5-1), a regular Lagrangian relaxation takes the following form:

$$\mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^\top(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}), \tag{5-2}$$

where $\boldsymbol{\lambda}$ is the Lagrange multiplier. For the same problem, using the *method of multipliers* requires an augmented Lagrangian $\mathcal{L}_\rho$ with

$$\mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^\top(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) + \frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2, \tag{5-3}$$

where λ is the Lagrange multiplier and ρ is the penalty parameter. The purpose of the augmented Lagrangian is to improve the robustness of the dual ascent method, which involves solving a dual problem using gradient ascent and is also used by dual decomposition explained

in Section 2-4-2. The augmented Lagrangian $\mathcal{L}_\rho$ can also be interpreted as a regular Lagrangian $\mathcal{L}$ on the modified problem

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{aligned} \tag{5-4}$$

which has the same solution as its original problem, where $f(\boldsymbol{x})$ is minimized, as $\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$ is 0 due to the constraints. Using dual ascent on the modified problem would result in the update steps being

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \ \mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{\lambda}_k) \\ \boldsymbol{\lambda}_{k+1} &= \boldsymbol{\lambda}_k + \rho(\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{b}), \end{aligned} \tag{5-5}$$

where $\mathcal{L}_\rho$ and ρ are the augmented Lagrangian and the penalty term instead of the regular Lagrangian $\mathcal{L}$ and the step size α. However, the *method of multipliers* no longer decomposes the problem, due to the augmented Lagrangian $\mathcal{L}_\rho$ being coupled.

ADMM can be seen as a hybrid between dual decomposition and the augmented Lagrangian from the method of multipliers. The intended main problems to be solved using ADMM take the form of

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{z}} \quad & f(\boldsymbol{x}) + g(\boldsymbol{z}) \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} = \boldsymbol{c}, \end{aligned} \tag{5-6}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{z} \in \mathbb{R}^m$ can be independent state vectors, and $\boldsymbol{A} \in \mathbb{R}^{p\times n}$, $\boldsymbol{B} \in \mathbb{R}^{p\times m}$, $\boldsymbol{c} \in \mathbb{R}^p$ contain the (coupled) constraints for f and g. The augmented Lagrangian therefore takes the form

$$\mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + g(\boldsymbol{z}) + \boldsymbol{\lambda}^\top(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c}) + \frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c}\|_2^2, \tag{5-7}$$

The overall ADMM algorithm has a similar structure to that of dual ascent/decomposition with a parallel minimization step and a multiplier update, having the form

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \ \mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{z}_k, \boldsymbol{\lambda}_k) \\ \boldsymbol{z}_{k+1} &= \arg\min_{\boldsymbol{z}} \ \mathcal{L}_\rho(\boldsymbol{x}_{k+1}, \boldsymbol{z}, \boldsymbol{\lambda}_k) \\ \boldsymbol{\lambda}_{k+1} &= \boldsymbol{\lambda}_k + \rho(\boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{z}_{k+1} - \boldsymbol{c}), \end{aligned} \tag{5-8}$$

where $\boldsymbol{x}_{k+1}$ and $\boldsymbol{z}_{k+1}$ are in this case sequential updates of the states, with $\boldsymbol{\lambda}_{k+1}$ being the new Lagrange multiplier with Lagrange parameter $\rho > 0$. The alternating direction part of ADMM is derived from the way $\boldsymbol{x}$ and $\boldsymbol{z}$ are updated, alternating between the two. Boyd *et al.* also refer to this as a single *Gauss-Seidel* pass [96] over $\boldsymbol{x}$ and $\boldsymbol{z}$, as opposed to the joint minimization in the regular method of multipliers. Furthermore, Boyd *et al.* elaborate that the alternation is what allows ADMM to be decomposed if $f(\boldsymbol{x})$ and $g(\boldsymbol{z})$ are separable. It should also be noted that the order in which $\boldsymbol{x}$ and $\boldsymbol{z}$ are updated is arbitrary, and can be reversed either in the update or in the problem formulation. There is however a drawback to this version of ADMM as $\boldsymbol{x}_{k+1}$ and $\boldsymbol{z}_{k+1}$ cannot be computed in parallel, reducing the potential gain in computational speed achieved by traditional dual decomposition. The actual impact of the sequential constraint can be minimized if the total state update frequency is faster than the required update rate. Note that Boyd *et al.* do not consider $\boldsymbol{x}_k$ to be a

separate state in the ADMM algorithm, as it is defined to be fully dependent on the $\boldsymbol{z}_{k-1}$ and $\boldsymbol{\lambda}_{k-1}$.

Boyd *et al.* also utilize a scaled formulation of ADMM instead of the form in Equation (5-8)

$$\begin{aligned}\boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \; f(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z}_k - \boldsymbol{c} + \boldsymbol{u}_k\|_2^2 \\ \boldsymbol{z}_{k+1} &= \arg\min_{\boldsymbol{z}} \; g(\boldsymbol{z}) + \frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c} + \boldsymbol{u}_k\|_2^2 \\ \boldsymbol{u}_{k+1} &= \boldsymbol{u}_k + \boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{z}_{k+1} - \boldsymbol{c},\end{aligned} \tag{5-9}$$

where the Lagrange multiplier λ is encoded in $\boldsymbol{u} := \frac{1}{\rho}\boldsymbol{\lambda}$ [95]. The scaled form is mainly used to simplify the notation of ADMM, as factor containing $\boldsymbol{\lambda}$ is not contained inside the norm.

5-1-1 Stopping Criteria

In order to define the stopping criteria for ADMM the primal and dual residuals are introduced, with the primal residual at step k being

$$\boldsymbol{r}^{k+1} = \boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z}^{k+1} - \boldsymbol{c}, \tag{5-10}$$

and the dual residual being

$$\boldsymbol{s}^{k+1} = \rho\boldsymbol{A}^\top\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k). \tag{5-11}$$

These residuals are derived from the primal and dual feasibility conditions; primal feasibility requires the ADMM solution to satisfy the original constraint, i.e.

$$\boldsymbol{A}\boldsymbol{x}^\star + \boldsymbol{B}\boldsymbol{z}^\star - \boldsymbol{c} = 0; \tag{5-12}$$

the dual feasibility condition requires that both the x-optimization step and z-optimization step have reached a local minimum, i.e.

$$0 \in \partial f(\boldsymbol{x}^\star) + \boldsymbol{A}^\top\boldsymbol{\lambda}^\star \tag{5-13}$$

$$0 \in \partial g(\boldsymbol{x}^\star) + \boldsymbol{B}^\top\boldsymbol{\lambda}^\star, \tag{5-14}$$

where the subdifferential optimality condition, i.e. $0 \in f(\boldsymbol{x})$ is used. The subdifferential operator ∂ is the set of all derivatives, which for a differentiable function is its gradient.

Depending on the order of the ADMM steps, and at which step the feasibility is evaluated, either Equation (5-13) or Equation (5-14) is satisfied by definition. For the standard ADMM update order we assume that dual feasibility for Equation (5-14) is always satisfied, which means that we only need to check for Equation (5-12) and Equation (5-13). The assumption made is valid as the z-update defines $\boldsymbol{z}^{k+1}$ as the minimizer for $\mathcal{L}_\rho(\boldsymbol{x}^{k+1}, \boldsymbol{z}, \boldsymbol{\lambda}^k)$, i.e.

$$\begin{aligned}0 &\in \partial\mathcal{L}_\rho(\boldsymbol{x}^{k+1}, \boldsymbol{z}, \boldsymbol{\lambda}^k) \\ &= \partial g(\boldsymbol{z}^{k+1}) + \boldsymbol{B}^\top\boldsymbol{\lambda}^k + \rho\boldsymbol{B}^\top(\boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z}^{k+1} - \boldsymbol{c}) \\ &= \partial g(\boldsymbol{z}^{k+1}) + \boldsymbol{B}^\top\boldsymbol{\lambda}^k + \rho\boldsymbol{B}^\top\boldsymbol{r}^{k+1} \\ &= \partial g(\boldsymbol{z}^{k+1}) + \boldsymbol{B}^\top\boldsymbol{\lambda}^{k+1},\end{aligned}$$

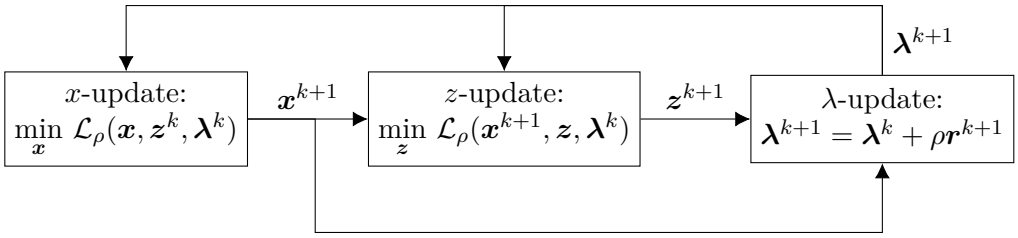


Figure 5-1: Diagram of ADMM steps

where $\boldsymbol{r}^{k+1}$ is the primal residual as defined in Equation (5-10), and the λ-update is used to simplify the last step.

The primal residual given in Equation (5-10) is easily derived from Equation (5-12) by taking the difference between the actual value and zero. The dual residual Equation (5-11) results from the fact that $\boldsymbol{x}^{k+1}$ is a minimizer for $\mathcal{L}_\rho(x, \boldsymbol{z}^k, \boldsymbol{\lambda}^k)$, i.e.

$$\begin{aligned} 0 &\in \partial \mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{z}^k, \boldsymbol{\lambda}^k) \\ &= \partial f(\boldsymbol{x}^{k+1}) + \boldsymbol{A}^\top \boldsymbol{\lambda}^k + \rho \boldsymbol{A}^\top (\boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z}^k - \boldsymbol{c}) \\ &= \partial f(\boldsymbol{x}^{k+1}) + \boldsymbol{A}^\top (\boldsymbol{\lambda}^k + \rho \boldsymbol{r}^{k+1}) + \rho \boldsymbol{A}^\top \boldsymbol{B}(\boldsymbol{z}^k - \boldsymbol{z}^{k+1}) \\ &= \partial f(\boldsymbol{x}^{k+1}) + \boldsymbol{A}^\top \boldsymbol{\lambda}^{k+1} + \rho \boldsymbol{A}^\top \boldsymbol{B}(\boldsymbol{z}^k - \boldsymbol{z}^{k+1}), \end{aligned}$$

which can also be formulated as

$$\rho \boldsymbol{A}^\top \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k) \in \partial f(\boldsymbol{x}^{k+1}) + \boldsymbol{A}^\top \boldsymbol{\lambda}^{k+1}, \tag{5-15}$$

which satisfies the optimality condition Equation (5-13) when $\rho \boldsymbol{A}^\top \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k) = 0$, which is how the dual residual of Equation (5-11) is formed.

Given that the optimality conditions of Equations (5-13) and (5-14) are only guaranteed to be reached as $k \to \infty$ for ADMM, actual implementation utilizes different termination conditions. These conditions are based on the residuals acting as an upper bound on the suboptimality, terminating the algorithm when the residuals are sufficiently small, i.e.

$$\|\boldsymbol{r}^k\|_2 \leq \epsilon_p \quad \text{and} \quad \|\boldsymbol{s}^k\|_2 \leq \epsilon_d, \tag{5-16}$$

where ϵ_p is the primal residual tolerance, and ϵ_s is the dual residual tolerance. Both tolerances should be defined by the user, considering the scaling of the problem and the required accuracy for the application.

5-2 Convergence Results

For completeness some definitions required for the convergence proof are summarized first. The convergence proof requires a saddle point in the Lagrangian ($\mathcal{L}$); a saddle point is a point on a function, in this case $\mathcal{L}$, where the derivatives are all zero in the orthogonal directions, but simultaneously is not a (local) minimum or maximum. The proof further requires a function

to be closed, i.e. the sublevel set $\{x \in \mathrm{dom} f | f(x) \leq c\}$ is a closed set for all $c \in \mathbb{R}$, and proper convex, i.e. its effective domain[20] is nonempty and $f(x) > -\infty, \forall x$

The convergence properties of ADMM, as claimed in [95], is summarized as the following theorem for clarity:

Theorem 5-2.1. *When applying the ADMM algorithm of Equation* (5-8)*, given closed, proper, and convex functions f and g and a saddle point $p^\star$ in the Lagrangian $\mathcal{L}$, it holds that the primal and dual residuals converge to zero, i.e. $\boldsymbol{r}^k \to 0$ and $\boldsymbol{s}^k \to 0$, and that the objective converges to the saddle point, i.e. $p^k \to p^\star$.*

The proof for Theorem 5-2.1 is given in Section 5-2-1 in the form of three lemmas with their corresponding proofs. Furthermore, [97] provides a framework which can be used to prove convergence rates for ADMM and its variants.

The convergence results for ADMM claimed in Theorem 5-2.1 include two requirements. These two assumptions can be restated as:

1. The extended-real-valued separable functions $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ and $g : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ are closed, proper, and convex.
2. The original Lagrangian $\mathcal{L}$ has a saddle point.

The first assumption which requires the function to be closed, proper, and convex implies that the functions f and g are *solvable*, which is necessary for the x and z update steps in ADMM. A function being *solvable* means that there exists a value which minimizes the function, in the case of f (or g) this means that there exists a x (or z) which minimizes the augmented Lagrangian. Boyd *et al.* [95] also stress that the first assumption allows both f and g to be nondifferentiable and to be able to assume a value of $+\infty$. An example provided is that f can be an indicator function for a nonempty convex set $\mathcal{S}$, where $f(x) = 0$ for $x \in \mathcal{S}$, and $f(x) = +\infty$ everywhere else. The first assumption is also summarized using epigraphs, with the epigraph of f being

$$\mathbf{epi} f := \{(x, t) \in \mathbb{R}^n \times \mathbb{R} | f(x) \leq t\}. \tag{5-17}$$

The function f is then satisfying the first assumption when $\mathbf{epi} f$ is a closed nonempty convex set.

The second assumption is mathematically defined as

$$\exists (\boldsymbol{x}^\star, \boldsymbol{z}^\star, \boldsymbol{\lambda}^\star), \text{ where } \mathcal{L}(\boldsymbol{x}^\star, \boldsymbol{z}^\star, \boldsymbol{\lambda}) \leq \mathcal{L}(\boldsymbol{x}^\star, \boldsymbol{z}^\star, \boldsymbol{\lambda}^\star) \leq \mathcal{L}(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}^\star), \forall (\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}). \tag{5-18}$$

The value of $\mathcal{L}(\boldsymbol{x}^\star, \boldsymbol{z}^\star, \boldsymbol{\lambda}^\star)$ is guaranteed to be finite for its saddle points as the first assumption requires f and g to be proper convex functions. The result of this is that $(\boldsymbol{x}^\star, \boldsymbol{z}^\star)$ is a solution to Equation (5-6), with the equality constraint satisfied and the functions f and g being finite. Another result is that $\boldsymbol{\lambda}$ is dual optimal, with strong duality between the primal and dual problems.

The authors further note that ADMM has no further requirements for $\boldsymbol{A}, \boldsymbol{B}$, or $\boldsymbol{c}$, emphasizing that $\boldsymbol{A}$ and $\boldsymbol{B}$ do not have to be full rank.

[20] The effective domain of a function f is a subset of the domain of f, with the additional condition that $f(x) < +\infty, \forall x \in \mathrm{dom} f$.

5-2-1 Convergence Proof

The convergence proof provided in [95] by Boyd *et al.* will be summarized here, note however that Boyd *et al.* refer to [98] and [99] for more sophisticated results on the convergence, including more general penalties and inexact minimization.

The convergence proof follows from the assumptions made in Section 5-2, specifically that f and g are closed, proper, and convex, and that the original Lagrangian $\mathcal{L}$ has a saddle point.

Let the assumed saddle point of $\mathcal{L}$ be at $(\boldsymbol{x}^\star, \boldsymbol{z}^\star, \boldsymbol{\lambda}^\star)$, with

$$V^k = \frac{1}{\rho}\|\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^\star\|_2^2 + \rho\|\boldsymbol{B}(\boldsymbol{z}^k - \boldsymbol{z}^\star)\|_2^2 \tag{5-19}$$

being a Lyapunov-candidate-function, which implies that $V \geq 0$ with $V = 0$ only being true at the saddle point. Boyd *et al.* [95] utilize three inequalities, that, when valid, prove the convergence results claimed in Theorem 5-2.1. We therefore first prove Theorem 5-2.1 using three lemmas which are assumed to be true, afterwards the individual lemmas are proven. The first inequality is related to the Lyapunov-candidate-function, requiring it to decrease with each iteration. This decrease can be compared with the $\dot{V}(\boldsymbol{x}) \leq 0$ condition used in the Lyapunov stability criterion, the first lemma is therefore

Lemma 5-2.1 (Lyapunov decrease)**.** *The Lyapunov-function V from Equation* (5-19) *decreases with each iteration, proportionally to the norm of the residual $\boldsymbol{r}^{k+1}$ and the change between $\boldsymbol{z}^k$ and $\boldsymbol{z}^{k+1}$, i.e.:*

$$V^{k+1} \leq V^k - \rho\|\boldsymbol{r}^{k+1}\|_2^2 - \rho\|\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\|_2^2. \tag{5-20}$$

The result from Lemma 5-2.1 is that $\boldsymbol{\lambda}^k$ and $\boldsymbol{B}\boldsymbol{z}^k$ are bounded given that $V^k \leq V^0$. Rewriting Equation (5-20) gives

$$\rho\left(\|\boldsymbol{r}^{k+1}\|_2^2 + \|\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\|_2^2\right) \leq V^k - V^{k+1},$$

which when iterated from $k = 0$ to $k = \infty$ results in

$$\rho\sum_{k=0}^{\infty}\left(\|\boldsymbol{r}^{k+1}\|_2^2 + \|\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\|_2^2\right) \leq \sum_{k=0}^{\infty}(V^k - V^{k+1}),$$

which can finally be written as

$$\sum_{k=0}^{\infty}\left(\|\boldsymbol{r}^{k+1}\|_2^2 + \|\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\|_2^2\right) \leq V^0.$$

Given that the left hand side is summed from $k = 0$ to $k = \infty$ while being bounded by V^0, it can be concluded that as $k \to \infty$, $\boldsymbol{r}^k \to \boldsymbol{0}$ and $\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k) \to \boldsymbol{0}$; this implies that the both the primal and dual residuals converge to zero which proves the first part of the convergence results claimed in Theorem 5-2.1.

The second lemma required is

Lemma 5-2.2 (Objective suboptimality upper bound)**.**

$$p^{k+1} - p^\star \leq -{\boldsymbol{\lambda}^{k+1}}^\top \boldsymbol{r}^{k+1} - \rho\left(\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right)^\top\left(-\boldsymbol{r}^{k+1} + \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^\star)\right).$$

Given that $\boldsymbol{r}^{k+1} \to \mathbf{0}$, and $\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k) \to \mathbf{0}$, Lemma 5-2.2 can be written as

$$p^{k+1} - p^\star \leq -(\boldsymbol{\lambda}^{k+1})^\top \mathbf{0} - \rho(\mathbf{0})^\top \left(\mathbf{0} + \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^\star)\right),$$

as $k \to \infty$, where the entire right hand side converges to zero given that $\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^\star)$ is bounded. Similarly, the third lemma

Lemma 5-2.3 (Objective suboptimality lower bound)**.** $p^\star - p^{k+1} \leq \boldsymbol{\lambda}^{\star\top} \boldsymbol{r}^{k+1}$.

has its right hand side converge to zero as $\boldsymbol{r}^{k+1} \to \mathbf{0}$ for $k \to \infty$. Lemmas 5-2.2 and 5-2.3 therefore imply that $\lim_{k\to\infty} p^k = p^\star$, meaning that the objective function converges, this proves the second part of the convergence results claimed in Theorem 5-2.1.

Proof for Lemma 5-2.3

Recall that the saddle point of $\mathcal{L}$, which is defined at $(\boldsymbol{x}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\lambda}^\star)$, implies that $\mathcal{L}(\boldsymbol{x}^\star, \boldsymbol{z}^\star, \boldsymbol{\lambda}^\star) \leq \mathcal{L}(\boldsymbol{x}^{k+1}, \boldsymbol{z}^{k+1}, \boldsymbol{\lambda}^\star)$, which is given in Equation (5-18). As the constraint is satisfied in the saddle point, i.e. $\boldsymbol{A}\boldsymbol{x}^\star + \boldsymbol{B}\boldsymbol{z}^\star = \boldsymbol{c}$, $\mathcal{L}(\boldsymbol{x}^\star, \boldsymbol{z}^\star, \boldsymbol{\lambda}^\star) = p^\star$. The objective function in its iterative from is written as $p^{k+1} = f(\boldsymbol{x}^{k+1}) + g(\boldsymbol{z}^{k+1})$, which when combined with $p^\star$ and the saddle point properties results in

$$p^\star \leq p^{k+1} + \boldsymbol{\lambda}^{\star\top} \boldsymbol{r}^{k+1}$$

which is the inequality given in Lemma 5-2.3.

Proof for Lemma 5-2.2

The first step in ADMM minimizes the augmented Lagrangian $\mathcal{L}_\rho$ using $\boldsymbol{x}$, with the minimizer being defined as $\boldsymbol{x}^{k+1}$. As per the first assumption made in Section 5-2 f is closed, proper, and convex, this means that f and $\mathcal{L}_\rho$ are subdifferentiable. The optimality condition for the $\boldsymbol{x}^{k+1}$ optimization step is

$$0 \in \partial \mathcal{L}_\rho(\boldsymbol{x}^{k+1}, \boldsymbol{z}^k, \boldsymbol{\lambda}^k) = \partial f(\boldsymbol{x}^{k+1}) + \boldsymbol{A}^\top \boldsymbol{\lambda}^k + \rho \boldsymbol{A}^\top (\boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z}^k - \boldsymbol{c}), \tag{5-21}$$

which is both a necessary and sufficient condition. Note that Boyd *et al.* elaborate that the equation in Equation (5-21) is the result of 'the basic fact that the subdifferentiable of the sum of a subdifferentiable function and a differentiable function with domain $\mathbb{R}^n$ is the sum of the subdifferentiable and the gradient;'. The ADMM $\boldsymbol{\lambda}$-update step $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \rho(\boldsymbol{r}^{k+1})$, can be written as $\boldsymbol{\lambda}^k = \boldsymbol{\lambda}^{k+1} - \rho(\boldsymbol{r}^{k+1})$ and combined with Equation (5-21) resulting in

$$0 \in \partial f(\boldsymbol{x}^{k+1}) + \boldsymbol{A}^\top \left(\boldsymbol{\lambda}^{k+1} - \rho \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k))\right),$$

(recall that $\boldsymbol{r}^k = \boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z}^{k+1} - \boldsymbol{c}$). This equation implies that the x-minimization step of $\boldsymbol{x}^{k+1}$ is minimizing

$$f(\boldsymbol{x}) + \left(\boldsymbol{\lambda}^{k+1} - \rho \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right)^\top \boldsymbol{A}\boldsymbol{x},$$

with a similar approach applicable to the z-minimization step where $\boldsymbol{z}^{k+1}$ is minimizing

$$g(\boldsymbol{z}) + (\boldsymbol{\lambda}^{k+1})^\top \boldsymbol{B}\boldsymbol{z}.$$

Using these it can be shown that

$$f(\boldsymbol{x}^{k+1}) + \left(\boldsymbol{\lambda}^{k+1} - \rho \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^{k})\right)^{\top} \boldsymbol{A}\boldsymbol{x}^{k+1} \leq f(\boldsymbol{x}^{\star}) + \left(\boldsymbol{\lambda}^{k+1} - \rho \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^{k})\right)^{\top} \boldsymbol{A}\boldsymbol{x}^{\star}, \tag{5-22}$$

with a similar result for z:

$$g(\boldsymbol{z}^{k+1}) + (\boldsymbol{\lambda}^{k+1})^{\top} \boldsymbol{B}\boldsymbol{z}^{k+1} \leq g(\boldsymbol{z}^{\star}) + (\boldsymbol{\lambda}^{k+1})^{\top} \boldsymbol{B}\boldsymbol{z}^{\star}. \tag{5-23}$$

When combining Equation (5-22) and Equation (5-23), we get

$$\begin{aligned} f(\boldsymbol{x}^{k+1}) + \left(\boldsymbol{\lambda}^{k+1} - \rho \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^{k})\right)^{\top} \boldsymbol{A}\boldsymbol{x}^{k+1} + g(\boldsymbol{z}^{k+1}) + (\boldsymbol{\lambda}^{k+1})^{\top} \boldsymbol{B}\boldsymbol{z}^{k+1} \\ \leq f(\boldsymbol{x}^{\star}) + \left(\boldsymbol{\lambda}^{k+1} - \rho \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^{k})\right)^{\top} \boldsymbol{A}\boldsymbol{x}^{\star} + g(\boldsymbol{z}^{\star}) + (\boldsymbol{\lambda}^{k+1})^{\top} \boldsymbol{B}\boldsymbol{z}^{\star}, \end{aligned}$$

which can then be rewritten as

$$p^{k+1} - p^{\star} \leq (\boldsymbol{\lambda}^{k+1})^{\top}(\boldsymbol{A}\boldsymbol{x}^{\star} + \boldsymbol{B}\boldsymbol{z}^{\star} - \boldsymbol{A}\boldsymbol{x}^{k+1} - \boldsymbol{B}\boldsymbol{z}^{k+1}) - \rho \left(\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^{k})\right)^{\top} (\boldsymbol{A}\boldsymbol{x}^{\star} - \boldsymbol{A}\boldsymbol{x}^{k+1}),$$

given that $p^{k} = f(\boldsymbol{x}^{k}) + g(\boldsymbol{z}^{k})$. It is then further simplified using $\boldsymbol{A}\boldsymbol{x}^{\star} + \boldsymbol{B}\boldsymbol{z}^{\star} = c$ and $\boldsymbol{r}^{k+1} = \boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{x}^{k+1} - c$ into

$$p^{k+1} - p^{\star} \leq -(\boldsymbol{\lambda}^{k+1})^{\top}\boldsymbol{r}^{k+1} - \rho \left(\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^{k})\right)^{\top} \left(-\boldsymbol{r}^{k+1} + \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^{\star})\right),$$

which is equal to the inequality given in Lemma 5-2.2.

Proof for Lemma 5-2.1

The proof for Equation (5-20) uses the inequalities of given in Lemmas 5-2.2 and 5-2.3 by combining them into

$$0 \leq -(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{\star})^{\top}\boldsymbol{r}^{k+1} - \rho \left(\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^{k})\right)^{\top} \left(-\boldsymbol{r}^{k+1} + \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^{\star})\right),$$

which is then rewritten and multiplied by 2 in order to form the basis for the proof of Equation (5-20), resulting in

$$\begin{aligned} 2(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{\star})^{\top}\boldsymbol{r}^{k+1} - 2\rho \left(\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^{k})\right)^{\top} \boldsymbol{r}^{k+1} \\ + 2\rho \left(\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^{k})\right)^{\top} \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^{\star}) \leq 0. \end{aligned} \tag{5-24}$$

First, using the Lagrangian update, i.e. $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^{k} + \rho \boldsymbol{r}^{k+1}$, the first term of Equation (5-24), $2(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{\star})^{\top}\boldsymbol{r}^{k+1}$, is rewritten as

$$2(\boldsymbol{\lambda}^{k} - \boldsymbol{\lambda}^{\star})^{\top}\boldsymbol{r}^{k+1} + \rho\|\boldsymbol{r}^{k+1}\|_2^2 + \rho\|\boldsymbol{r}^{k+1}\|_2^2,$$

where $\boldsymbol{r}^{k+1} = \frac{1}{\rho}(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k})$ can be substituted into the first two terms giving

$$\frac{2}{\rho}(\boldsymbol{\lambda}^{k} - \boldsymbol{\lambda}^{\star})^{\top}(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k}) + \frac{1}{\rho}\|(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k})\|_2^2 + \rho\|\boldsymbol{r}^{k+1}\|_2^2,$$

which can be reduced using the fact that $\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k = (\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^\star) - (\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^\star)$

$$\frac{2}{\rho}(\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^\star)^\top(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^\star) - \frac{2}{\rho}\|\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^\star\|_2^2 + \frac{1}{\rho}\|(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^\star) - (\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^\star)\|_2^2 + \rho\|\boldsymbol{r}^{k+1}\|_2^2,$$

where we can use $\|\boldsymbol{u} - \boldsymbol{v}\|_2^2 = \|\boldsymbol{u}\|_2^2 + \|\boldsymbol{v}\|_2^2 - 2\boldsymbol{v}^\top\boldsymbol{u}$ to get

$$\frac{1}{\rho}\left(\|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^\star\|_2^2 - \|\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^\star\|_2^2\right) + \rho\|\boldsymbol{r}^{k+1}\|_2^2. \tag{5-25}$$

Given that Equation (5-25) is the first term of Equation (5-24) rewritten, the current rewritten version of Equation (5-24) is then

$$\frac{1}{\rho}\left(\|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^\star\|_2^2 - \|\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^\star\|_2^2\right) + \rho\|\boldsymbol{r}^{k+1}\|_2^2 - 2\rho\left(\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right)^\top \boldsymbol{r}^{k+1} + 2\rho\left(\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right)^\top \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^\star) \leq 0,$$

where we will simplify the last term using a similar method as before, with the fact that $\boldsymbol{z}^{k+1} - \boldsymbol{z}^\star = (\boldsymbol{z}^{k+1} - \boldsymbol{z}^k) + (\boldsymbol{z}^k - \boldsymbol{z}^\star)$, resulting in

$$\ldots + \rho\left(\|\boldsymbol{r}^{k+1}\|_2^2 - 2\left(\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right)^\top \boldsymbol{r}^{k+1} + 2\|\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\|_2^2\right) + 2\rho\left(\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right)^\top \boldsymbol{B}(\boldsymbol{z}^k - \boldsymbol{z}^\star) \leq 0,$$

where we can combine the norms, simplifying it into

$$\ldots + \rho\|\boldsymbol{r}^{k+1} - \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\|_2^2 + \rho\|\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\|_2^2 + 2\rho\left(\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right)^\top \boldsymbol{B}((\boldsymbol{z}^k - \boldsymbol{z}^\star)) \leq 0,$$

where we can again apply a simple substitution, with $\boldsymbol{z}^{k+1} - \boldsymbol{z}^k = (\boldsymbol{z}^{k+1} - \boldsymbol{z}^\star) - (\boldsymbol{z}^k - \boldsymbol{z}^\star)$, resulting in the last two terms being

$$\ldots + \rho\|\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^\star) - \boldsymbol{B}(\boldsymbol{z}^k - \boldsymbol{z}^\star)\|_2^2 - 2\rho\|\boldsymbol{B}(\boldsymbol{z}^k - \boldsymbol{z}^\star)\|_2^2 + 2\rho\left(\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^\star)\right)^\top \boldsymbol{B}((\boldsymbol{z}^k - \boldsymbol{z}^\star)) \leq 0,$$

which can finally be simplified by combining the norms into

$$\ldots + \rho\left(\|\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^\star)\|_2^2 - \|\boldsymbol{B}(\boldsymbol{z}^k - \boldsymbol{z}^\star)\|_2^2\right) \leq 0.$$

The fully rewritten version of Equation (5-24) is now

$$\frac{1}{\rho}\left(\|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^\star\|_2^2 - \|\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^\star\|_2^2\right) + \rho\|\boldsymbol{r}^{k+1} - \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\|_2^2 + \rho\left(\|\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^\star)\|_2^2 - \|\boldsymbol{B}(\boldsymbol{z}^k - \boldsymbol{z}^\star)\|_2^2\right) \leq 0. \tag{5-26}$$

Now recall that $V^k = \frac{1}{\rho}\|\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^\star\|_2^2 + \rho\|\boldsymbol{B}(\boldsymbol{z}^k - \boldsymbol{z}^\star)\|_2^2$ which allows Equation (5-26) to be written as

$$V^{k+1} \leq V^k - \rho\|\boldsymbol{r}^{k+1} - \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\|_2^2,$$

which when expanded results in

$$V^{k+1} \leq V^k - \rho\|\boldsymbol{r}^{k+1}\|_2^2 - \rho\|\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\|_2^2 + 2\rho(\boldsymbol{r}^{k+1})^\top \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k), \tag{5-27}$$

which satisfies Equation (5-20) if and only if $2\rho(\boldsymbol{r}^{k+1})^\top \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k) \leq 0$. The proof is based on the fact that $\boldsymbol{z}^{k+1}$ is the minimizer of the ADMM z-update step, minimizing $g(\boldsymbol{z}) + (\boldsymbol{\lambda}^{k+1})^\top \boldsymbol{B}z$, it is therefore a given that

$$g(\boldsymbol{z}^{k+1}) + (\boldsymbol{\lambda}^{k+1})^\top \boldsymbol{B}\boldsymbol{z}^{k+1} \leq g(\boldsymbol{z}^k) + (\boldsymbol{\lambda}^{k+1})^\top \boldsymbol{B}\boldsymbol{z}^k,$$

which is essentially stating that the cost function at $\boldsymbol{z}^{k+1}$ is never going to be higher than at $\boldsymbol{z}^k$ for $\boldsymbol{\lambda}^{k+1}$. Similarly one can accurately claim that

$$g(\boldsymbol{z}^k) + (\boldsymbol{\lambda}^k)^\top \boldsymbol{B}\boldsymbol{z}^k \leq g(\boldsymbol{z}^{k+1}) + (\boldsymbol{\lambda}^k)^\top \boldsymbol{B}\boldsymbol{z}^{k+1},$$

as $\boldsymbol{z}^k$ is the minimizer for $\boldsymbol{\lambda}^k$. When combining these two inequalities the result is

$$(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k)^\top \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k) \leq 0,$$

where $\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k$ is the difference between the Lagrange multiplier for the current and next step, which is equal to ρ times the residual $\boldsymbol{r}^{k+1}$, giving

$$\rho(\boldsymbol{r}^{k+1})^\top \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k) \leq 0,$$

which allows Equation (5-27) to always satisfy Equation (5-20), proving Lemma 5-2.1.

Chapter 6

Adaptive ADMM (A-ADMM)

The Adaptive Alternating Direction Method of Multipliers is a variant on conventional ADMM where the penalty parameter ρ is no longer a static variable, and can be dynamic instead. Given that in conventional ADMM the requirement for ρ is that it is a constant larger than zero, a lot of room is left for the choice of this parameter. This chapter is divided in three sections: Section 6-1 examines the role of the penalty parameter in ADMM; Section 6-2 describes one of the earliest adaptive penalty parameter methods; finally, Section 6-3 covers a more advanced adaptive method based on the spectral gradient descent method.

6-1 Role of the penalty parameter ρ

In conventional ADMM, the penalty parameter ρ is introduced through the use of the augmented Lagrangian which is part of the method of multipliers. The penalty parameter has two important points of relevance, the first being in the augmented Lagrangian where ρ acts as a weight factor on the squared L2-norm of the complicating equality constraint, see Equation (5-3). The second point of relevance is the λ update step where ρ acts similarly to a step size for the Lagrange multiplier.

The equations for the original problem and its augmented Lagrangian used by ADMM are repeated here for convenience:

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{z}}{\text{minimize}} \quad & f(\boldsymbol{x}) + g(\boldsymbol{z}) \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} = \boldsymbol{c}, \end{aligned} \tag{6-1}$$

where $f(\boldsymbol{x})$ and $g(\boldsymbol{z})$ are proper convex functions, the augmented Lagrangian version of this problem is

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{z}}{\text{minimize}} \quad & \mathcal{L}_\rho, \\ \text{where} \quad & \mathcal{L}_\rho = f(\boldsymbol{x}) + g(\boldsymbol{z}) + \boldsymbol{\lambda}^\top(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c}) + \frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c}\|_2^2. \end{aligned} \tag{6-2}$$

The incorporation of the equality constraint into the augmented Lagrangian is also known as the method of multipliers and is explained in Section 5-1. The augmented Lagrangian is 'augmented' through the inclusion a squared L-2 norm term proportional to the penalty parameter ρ. This terms also appears in the Lagrange multiplier update, i.e.

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \rho(\boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z}^{k+1} - \boldsymbol{c}). \tag{6-3}$$

In [100], it is claimed that the use of $\boldsymbol{\lambda}^\top(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c})$ in Equation (6-2) allows the exact solution of the original problem, Equation (6-1), to be determined without making ρ tend to infinity, as opposed to ordinary penalization methods where this can cause problems to become ill-conditioned. It is also claimed that the squared norm term improves the convergence properties of the algorithm.

It is trivial to see that the value of ρ affects the weight of the primal residual, i.e. $\boldsymbol{r}^k = \boldsymbol{A}\boldsymbol{x}^k + \boldsymbol{B}\boldsymbol{z}^k - \boldsymbol{c}$, either through Equation (6-3), where the Lagrange multiplier is increased proportionally to the primal residual, or directly in Equation (6-2), where it scales the squared norm of the residual. Given the fact that $\rho > 0$ is required for ADMM to converge [101, 95], it follows that the magnitude of the penalty parameter ρ is inversely proportional to the primal residual obtained in Equation (6-2).

Suppose that the minimum of the unconstrained convex problem $\min_{\boldsymbol{x},\boldsymbol{z}}\ f(\boldsymbol{x}) + g(\boldsymbol{z})$, denoted as $\boldsymbol{x}_U^{min}$, is not equal to the minimum of constrained convex problem in Equation (6-1), denoted as $\boldsymbol{x}_C^{min}$, which is always the case when $\boldsymbol{x}_U^{min} \notin \mathcal{X}^f$, where $\mathcal{X}^f$ is the feasible set of the constrained convex problem. For these problems, it holds that if $f(\boldsymbol{x}), g(\boldsymbol{z})$ are proper convex functions, then the magnitude of ρ is positively correlated to the Euclidean distance between the unconstrained and the constrained minimum. This is the result of the fact that the minimum of the sum of two convex functions is a global minimum, which implies that any variation leads to an increase in cost. One can therefore conclude that $\boldsymbol{x}_C^{min} \geq \boldsymbol{x}_U^{min}$ when the overall problem is convex, and $\boldsymbol{x}_C^{min} > \boldsymbol{x}_U^{min}$ when $\boldsymbol{x}_U^{min} \notin \mathcal{X}^f$, where $\mathcal{X}^f$ is the feasible set of the constrained problem. We have previously shown that the penalty parameter ρ is inversely proportional to the primal residual $\boldsymbol{r}^k$, this implies that the minimum for the augmented Lagrangian, denoted as $\boldsymbol{x}_A^{min}$, moves from $\boldsymbol{x}_U^{min}$ towards $\boldsymbol{x}_C^{min}$ as ρ is increased. This is also visualized in Figure 6-1.

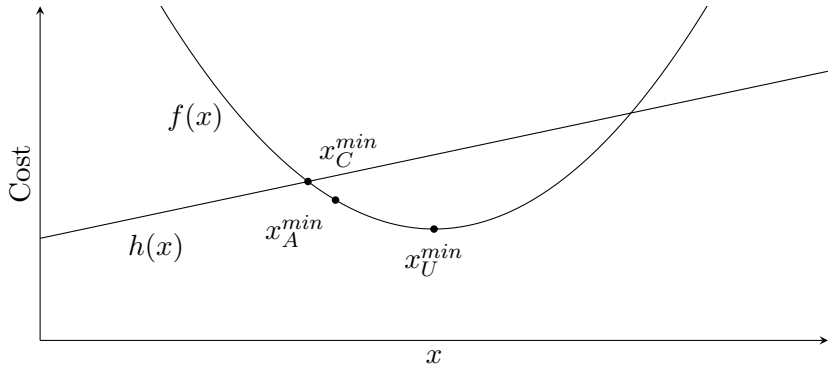


Figure 6-1: Example of the potential effects of the penalty parameter on the optimum. Where $f(x)$ is the unconstrained problem, $h(x)$ is the constraint, x_U^{min} refers to the unconstrained problem, x_C^{min} refers to the constrained problem, and x_A^{min} refers to the problem solved using an augmented Lagrangian.

Briefly summarizing the effects of the penalty parameter:

- Increasing ρ results in an equal or smaller residual r when using the augmented Lagrangian $\mathcal{L}_\rho$.
- Increasing ρ results in an equal or larger cost for the augmented Lagrangian minimum $\boldsymbol{x}_A^{min}$ compared with the unconstrained minimum $\boldsymbol{x}_U^{min}$.

Because of the duality of the value of ρ we cannot simply recommend the choice of very large ρ in the hopes that a good result is obtained. The potential increase in cost is especially an issue as it can cause the problem to become ill-conditioned or result in slow convergence. For certain problems it is also not always desirable to strictly adhere to the (certain) given constraint, e.g. when a minor violation of the constraint results in significantly lower costs whilst maintaining safe operation.

When dealing with offline optimization these type of problems are not very common, due to the fact that there is no strict time/iteration limit for convergence to occur: there would be no need for the constraint to be more conservative than necessary, as there is sufficient time for convergence to occur. The role of ρ becomes more relevant for online optimization, where computation time/power is limited, resulting in the need to use run-time solutions instead of fully converged solutions. Given that run-time solutions of ADMM are not strictly feasible, it is often necessary to utilize more conservative constraints in order to achieve feasible run-time solutions, the topic of online ADMM is further explored in Chapter 7.

Having control over run-time feasibility, optimality and computation time is also highly relevant when the optimization problem is adjusted in real-time. For example, when a problem is solved using a strategy like Convex Feasible Set (CFS) [102, 103], where a problem with a nonconvex feasible set is convexified and solved iteratively. The CFS strategy is also explained in Section 7-3-2. Given the convexification of the nonconvex feasible set $\mathcal{X}_N$, the convex feasible set $\mathcal{X}_C$ will always be a proper subset of the nonconvex feasible set, i.e. $\mathcal{X}_C \subset \mathcal{X}_N$ as per [103]. Allowing the user to design the trade-off between infeasible results and faster computation time, can in certain cases enable CFS, or other similar methods, to be applied online.

The intuition behind this is that infeasible results, which are easier to compute, allows faster computation times, resulting in higher frequency approximations (e.g. convexification or linearization). With higher frequency approximations, the time dependent inaccuracies are reduced as less time passes between approximations. Additionally, it is possible that infeasible results are actually feasible, assuming that the approximated constraints are always proper subsets, leading to faster convergence. However, whether this fact can be effectively exploited using an adaptive penalty parameter requires further research.

6-2 Residual Balancing

The use of a varying penalty parameter ρ for ADMM was first proposed in [28], where an update rule is introduced in order to improve convergence. The residual balancing strategy has an additional penalty parameter update after the λ-update step, with an example of a

ρ-update being

$$\rho^{k+1} = \begin{cases} \tau^{incr}\rho^k & \text{if} \quad \|\boldsymbol{r}^k\|_2 > \mu\|\boldsymbol{s}^k\|_2 \\ \rho^k/\tau^{decr} & \text{if} \quad \|\boldsymbol{s}^k\|_2 > \mu\|\boldsymbol{r}^k\|_2 \\ \rho^k & \text{otherwise,} \end{cases} \tag{6-4}$$

where $\boldsymbol{r}^k$ (resp. $\boldsymbol{s}^k$) is the primal (resp. dual) residual, and $\tau^{incr} > 1$, $\tau^{decr} > 1$, and $\mu > 1$ [28, 95]. Boyd *et al.* also state that typical choices might be $\mu = 10$ and $\tau^{incr} = \tau^{decr} = 2$. The reasoning behind the residual balancing scheme given in Equation (6-4) is to adjust the value of ρ depending on the magnitude of the primal and dual residuals, attempting to achieve a balance between the two. Whilst this method still requires the user to seemingly arbitrarily adjust parameters, the introduction of μ, τ^{incr}, and τ^{decr} allows more robust convergence with easier to interpret parameters. In Equation (6-4), μ limits the factor at which one residual is allowed to exceed the other, τ^{incr} (resp. τ^{decr}) then defines the factor by which the penalty parameter ρ is increased (resp. decreased). When there is no specific reason for τ^{incr} to be different from τ^{decr}, it is advised to use $\tau^{incr} = \tau^{decr}$.

If ADMM is viewed as a dynamic system, the τ used in residual balancing scheme can be interpreted as the derivative of ρ, i.e. $\tau = \dot{\rho}$. For that reason the use of residual balancing can be seen as a move from defining a static ρ towards defining a function $\rho(\cdot)$ (this function is later referred to as ϕ in Chapter 8). The residual balancing scheme therefore changes the burden of choosing the value of ρ into choosing the initial value ρ^0 and its higher order terms, e.g. τ. One thing to note is that the scheme of Equation (6-4) is a simple discontinuous function, where large values of τ can lead to undesired flip-flopping of ρ.

Given that the method is based on the primal and dual residual norms, it is possible for poorly scaled problems that attempting to balance these results in worse results compared with conventional methods. Another caveat is the fact that the initial ρ^0 is still very important for fast convergence depending on the value of τ and μ, if the initial ρ^0 is chosen poorly no significant improvement is made by the adaptation.

6-2-1 Convergence of self-adaptive penalty parameters

An important contribution by He *et al.* [28] is the convergence results for an adaptive penalty parameter. The basis of the convergence analysis in [28] is that observing

$$\lim_{k\to\infty} \left(\|\boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z}^{k+1} - \boldsymbol{c}\|_2^2 + \|\boldsymbol{B}(\boldsymbol{z}^k - \boldsymbol{z}^{k+1})\|_2^2 \right) = 0 \tag{6-5}$$

is sufficient to claim convergence: Equation (6-5) implies that $\boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z}^{k+1} = \boldsymbol{c}$ and $\boldsymbol{z}^k = \boldsymbol{z}^{k+1}$, which for ADMM can only be the case when the solution has converged.

We partially borrow the notation used in [31] for the two conditions used by He *et al.* in [28], with the conditions being the following:

Condition 6-2.1 (Bounded increasing)**.** $\inf\{\rho^k\}_1^\infty > 0$ and $\sum_{k=1}^\infty (\eta^k)^2 < +\infty$, where

$$\eta^k = \sqrt{\max(\frac{\rho^{k+1}}{\rho^k}, 1) - 1}. \tag{6-6}$$

Condition 6-2.2 (Bounded decreasing)**.** $\sum_{k=1}^{\infty}(\theta^k)^2 < +\infty$, where

$$\theta^k = \sqrt{\max(\frac{\rho^k}{\rho^{k+1}}, 1) - 1}. \tag{6-7}$$

Given that η^k (resp. θ^k) essentially measures the increase (resp. decrease) of ρ, the two conditions can be summarized as requiring the ρ to be either bounded increasing, or bounded decreasing for all k. These conditions are combined in Theorem 4.1 from [28], which can be summarized as:

Theorem 6-2.1. *If the adaptive penalty parameter ρ satisfies either Condition 6-2.1 or Condition 6-2.1 and is applied to an otherwise convergent conventional ADMM problem, then the adaptive variant also converges.*

The full proof of this theorem can be found in [28], with the essential part being the fact that both Condition 6-2.1 and Condition 6-2.1 lead towards Equation (6-5), which guarantees convergence.

6-3 Spectral Penalty Parameter Selection

The Adaptive ADMM method using Spectral Penalty Parameters owes part of its name to the spectral gradient methods pioneered by Barzilai and Borwein [104]. The key of these methods is the adaptive selection of a penalty parameter for fast convergence.

The spectral gradient method was proposed in [104], under the name of *Two-Point Step Size Gradient Methods*, as a solution to the poor performance of the traditional steepest-descent method both in terms of convergence rate and the ability to handle ill-conditioned problems.

For a smooth function $f : \mathbb{R}^n \to \mathbb{R}$, the classical gradient descent step has to form of

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \rho \nabla f(\boldsymbol{x}^k),$$

where ρ is the step-size. The standard spectral gradient method proposed by [104] sets $\rho^k = \alpha^k \boldsymbol{I}$, where $\alpha^k \boldsymbol{I}$ is imitating the Hessian of the function f over the previous step, making the method a Quasi-Newton method. The step-size ρ^k is defined using a least squares equation, with the definition for α^k being:

$$\alpha^k = \arg\min_{\alpha \in \mathbb{R}} \|\nabla f(\boldsymbol{x}^k) - \nabla f(\boldsymbol{x}^{k-1}) - \alpha(\boldsymbol{x}^k - \boldsymbol{x}^{k-1})\|_2^2. \tag{6-8}$$

The least squares equation given in Equation (6-8) is in fact a two point approximation of the Hessian used by the secant equation in quasi-Newton methods. As a reminder, the secant equation of quasi-Newton methods is

$$\nabla f(\boldsymbol{x}^k + \Delta \boldsymbol{x}) = \nabla f(\boldsymbol{x}^k) + \boldsymbol{B}\Delta \boldsymbol{x},$$

where $\boldsymbol{B}$ is the Hessian approximation.

Accordingly, the value of α^k is then given by

$$\alpha^k = \frac{\left(\boldsymbol{x}^k - \boldsymbol{x}^{k-1}\right) \cdot \left(\nabla f(\boldsymbol{x}^k) - \nabla f(\boldsymbol{x}^{k-1})\right)}{\left(\nabla f(\boldsymbol{x}^k) - \nabla f(\boldsymbol{x}^{k-1})\right) \cdot \left(\nabla f(\boldsymbol{x}^k) - \nabla f(\boldsymbol{x}^{k-1})\right)}. \tag{6-9}$$

This method forms the basis of the spectral penalty parameter Adaptive ADMM method proposed in [31].

6-3-1 Douglas-Rachford Splitting

As the spectral penalty parameter method is based on the duality between Alternating Direction Method of Multipliers (ADMM) and Douglas-Rachford splitting (DRS) [98], the Douglas-Rachdord splitting (DRS) method is briefly summarized. Douglas-Rachford Splitting is an optimization strategy which splits a main problem of the form

$$\min_{\boldsymbol{s}} q(\boldsymbol{s}) + p(\boldsymbol{s}), \tag{6-10}$$

using the strategy

$$\begin{aligned} \boldsymbol{s}^{k+1} &= \text{prox}_{\gamma,q}(\boldsymbol{t}^k) \\ \boldsymbol{t}^{k+1} &= \boldsymbol{t}^k + \text{prox}_{\gamma,p}(2\boldsymbol{s}^{k+1} - \boldsymbol{t}^k) - \boldsymbol{s}^{k+1}, \end{aligned} \tag{6-11}$$

where $\text{prox}_{\gamma,q}$ is the proximal point method with step size γ for function q, i.e.

$$\text{prox}_{\gamma,q}(\boldsymbol{s}) = \arg\min_{\boldsymbol{z}} \left(q(\boldsymbol{z}) + \frac{1}{2\gamma} \|\boldsymbol{z} - \boldsymbol{s}\|_2^2 \right).$$

The strategy can also be rewritten as

$$\begin{aligned} \boldsymbol{r}^{k+1} &= \text{prox}_{\gamma,q}(\boldsymbol{s}^k + \boldsymbol{l}^k) \\ \boldsymbol{s}^{k+1} &= \text{prox}_{\gamma,p}(\boldsymbol{r}^{k+1} - \boldsymbol{l}^k) \\ \boldsymbol{l}^{k+1} &= \boldsymbol{l}^k + \boldsymbol{s}^{k+1} - \boldsymbol{r}^{k+1}, \end{aligned} \tag{6-12}$$

with $\boldsymbol{l}^k = \boldsymbol{s}^k - \boldsymbol{t}^k$.

When applying DRS to the problem of Equation (5-6), which is the main type of problem solved by ADMM and is repeated here for convenience:

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{z}} \quad & f(\boldsymbol{x}) + g(\boldsymbol{z}) \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} = \boldsymbol{c}, \end{aligned}$$

with its dual reformulation usable by DRS being

$$\max_{\boldsymbol{s}} -\boldsymbol{c}^\top \boldsymbol{s} - f^*(-\boldsymbol{A}^\top \boldsymbol{s}) - g^*(-\boldsymbol{B}^\top \boldsymbol{s}), \tag{6-13}$$

where the $*$ superscript denotes the Fenchel conjugate, e.g. $f^*(\boldsymbol{y}) = \sup_{\boldsymbol{x}} \langle \boldsymbol{x}, \boldsymbol{y} \rangle - f(\boldsymbol{x})$, written in the form of Equation (6-10) gives:

$$\min_{\boldsymbol{s}} \underbrace{\boldsymbol{c}^\top \boldsymbol{s} + f^*(-\boldsymbol{A}^\top \boldsymbol{s})}_{q(\boldsymbol{s})} + \underbrace{g^*(-\boldsymbol{B}^\top \boldsymbol{s})}_{p(\boldsymbol{s})}. \tag{6-14}$$

Applying the DRS method from Equation (6-12) to the dual form results in

$$\begin{aligned}\boldsymbol{r}^{k+1} &= \arg\min_{\boldsymbol{u}} \left(q(\boldsymbol{u}) + \frac{1}{2\gamma}\|\boldsymbol{u} - (\boldsymbol{s}^k + \boldsymbol{l}^k)\|_2^2 \right) \\ \boldsymbol{s}^{k+1} &= \arg\min_{\boldsymbol{u}} \left(p(\boldsymbol{u}) + \frac{1}{2\gamma}\|\boldsymbol{u} - (\boldsymbol{r}^{k+1} - \boldsymbol{l}^k)\|_2^2 \right) \\ \boldsymbol{l}^{k+1} &= \boldsymbol{l}^k + \boldsymbol{s}^{k+1} - \boldsymbol{r}^{k+1},\end{aligned}$$

which is proven to have primal equivalence to the ADMM method in [105]. Furthermore Xu *et al.* [31] state that DRS solves Equation (6-14) by splitting $\boldsymbol{s}$ into $\boldsymbol{s}^k$ and $\hat{\boldsymbol{s}}^k$ according to

$$\begin{aligned}0 &\in \frac{\hat{\boldsymbol{s}}^{k+1} - \boldsymbol{s}^k}{\rho^k} + \partial q(\hat{\boldsymbol{s}}^{k+1}) + \partial p(\boldsymbol{s}^k) \\ 0 &\in \frac{\boldsymbol{s}^{k+1} - \boldsymbol{s}^k}{\rho^k} + \partial q(\hat{\boldsymbol{s}}^{k+1}) + \partial p(\boldsymbol{s}^{k+1}).\end{aligned} \tag{6-15}$$

6-3-2 Methodology

Given that conventional ADMM is a higher level optimization strategy, and therefore does not explicitly define anything closely related to gradient descent, the spectral step size is first applied to the Douglas-Rachford splitting method.

Applying the spectral step size to DRS involves a linear approximation of the partial derivatives of the functions q and p from Equation (6-10), i.e.

$$\begin{aligned}\partial q(\hat{\boldsymbol{s}}) &\approx \alpha^k \hat{\boldsymbol{s}} + \Psi^k \\ \partial p(\boldsymbol{s}) &\approx \beta^k \boldsymbol{s} + \Phi^k,\end{aligned} \tag{6-16}$$

where α^k (resp. β^k) is a local estimates of the curvature of q (resp. p), and Ψ^k, Φ^k have the same dimension as s. Xu *et al.* [31] then propose the following (rewritten):

Theorem 6-3.1. *Let DRS be applied to Equation* (6-14)*, with*

$$\partial q(\hat{\boldsymbol{s}}) = \alpha^k \hat{\boldsymbol{s}} + \Psi^k, \text{ and } \partial p(\boldsymbol{s}) = \beta^k \boldsymbol{s} + \Phi^k$$

Then, $\rho^k = (\alpha\beta)^{-\frac{1}{2}}$ *results in the minimal residual of* $q(\boldsymbol{s}^{k+1})$ *and* $p(\boldsymbol{s}^{k+1})$*.*

For the full proof please see [31].

The A-ADMM method proposed in [31] utilizes an adaptive penalty parameter ρ to replicate spectral gradient descent. This is achieved by using a least squares formulation to estimate the curvature parameters of A-ADMM. Given that the curvature parameters α, β are not know exactly, they are estimated locally using the current iteration k and a previous iteration k_0. The estimation is based on a linear model, i.e.

$$\Delta q^k \approx \alpha \Delta \hat{\boldsymbol{s}}^k + \boldsymbol{a},$$

where $\Delta\hat{s}^k := \hat{\boldsymbol{s}}^k - \hat{\boldsymbol{s}}^{k_0}$, and $\Delta q^k := \partial q(\hat{\boldsymbol{s}}^k) - \partial q(\hat{\boldsymbol{s}}^{k_0})$ Xu *et al.* note that in BB (Brazilai and Borwein) type spectral gradient methods it is common to estimate α using one of two least squares problems:

$$\min_{\alpha} \|\Delta q^k - \alpha \Delta \hat{s}^k\|_2^2,$$

or through

$$\min_{\alpha} \|\alpha^{-1} \Delta q^k - \Delta \hat{\boldsymbol{s}}^k\|_2^2.$$

The solutions for the two problems can be written as

$$\hat{\alpha}_{SD}^k = \frac{\|\Delta \hat{s}^k\|_2^2}{\langle \Delta q^k, \Delta \hat{s}^k \rangle}, \tag{6-17}$$

and

$$\hat{\alpha}_{MG}^k = \frac{\langle \Delta q^k, \Delta \hat{s}^k \rangle}{\|\Delta q^k\|_2^2}, \tag{6-18}$$

where the subscript SD (resp. MG) means *steepest descent* (resp. *minimal gradient*[21]), which follows the notation used in [31, 106]. Similarly the steps for β are

$$\hat{\beta}_{SD}^k = \frac{\|\Delta s^k\|_2^2}{\langle \Delta p^k, \Delta s^k \rangle}, \tag{6-19}$$

and

$$\hat{\beta}_{MG}^k = \frac{\langle \Delta p^k, \Delta s^k \rangle}{\|\Delta p^k\|_2^2}, \tag{6-20}$$

where $\Delta p^k := \boldsymbol{B}(\boldsymbol{z}^k - \boldsymbol{z}^{k_0})$ and $\Delta s^k = \boldsymbol{s}^k - \boldsymbol{s}^{k_0}$. The purpose of having the two estimate options is to utilize both methods through a hybrid scheme proposed by [106] where

$$\hat{\alpha}^k = \begin{cases} \hat{\alpha}_{MG}^k & \text{if} \quad 2\hat{\alpha}_{MG}^k > \hat{\alpha}_{SD}^k \\ \hat{\alpha}_{SD}^k - \frac{1}{2\hat{\alpha}_{MG}^k} & \text{otherwise,} \end{cases} \tag{6-21}$$

which can similarly be applied to the spectral step-size β:

$$\hat{\beta}^k = \begin{cases} \hat{\beta}_{MG}^k & \text{if} \quad 2\hat{\beta}_{MG}^k > \hat{\beta}_{SD}^k \\ \hat{\beta}_{SD}^k - \frac{1}{2\hat{\beta}_{MG}^k} & \text{otherwise.} \end{cases} \tag{6-22}$$

Given that a spectral step-size estimate is based on a linear model, the accuracy of the approximation is widely dependent on the type of functions it is applied to. In the classical BB method the unreliable estimates are dealt with using a line search, Xu *et al.* however note that this is not possible in ADMM, given the lack of distinction between "stable" or "unstable" step-sizes. Xu *et al.* therefore propose to quantify the quality of the curvature estimate through the correlation between the change in dual (sub)gradient and dual variables. If the correlation is high enough the spectral step-size estimate is used. The correlation of α^{k+1} is denoted as

$$\alpha_{cor}^{k+1} = \frac{\langle \Delta \hat{q}^k, \Delta \hat{\lambda}^k \rangle}{\|\Delta \hat{q}^k\| \ \|\Delta \hat{\lambda}^k\|}, \tag{6-23}$$

[21] Minimal gradient is a line search method which minimizes the gradient norm. [106]

and the correlation of β^{k+1} is denoted as

$$\beta_{cor}^{k+1} = \frac{\langle \Delta\hat{p}^k, \Delta\hat{\lambda}^k \rangle}{\|\Delta\hat{p}^k\| \, \|\Delta\hat{\lambda}^k\|}. \tag{6-24}$$

The purpose of using the correlations as a measurement of quality is the linear model assumption, which assumes that the dual (sub)gradient is linear w.r.t to the change in dual variables. If the correlation between the values is high, it is assumed that the linear estimate is accurate. Using these assumptions the update rule for the penalty parameter can be formed:

$$\rho^{k+1} = \begin{cases} \sqrt{\hat{\alpha}^{k+1}\hat{\beta}^{k+1}} & \text{if} \quad \alpha_{cor}^k > \epsilon_{cor}^{k+1} \text{ and } \beta_{cor}^{k+1} > \epsilon_{cor}^{k+1} \\ \hat{\alpha}^{k+1} & \text{if} \quad \alpha_{cor}^k > \epsilon_{cor}^{k+1} \text{ and } \beta_{cor}^{k} \leq \epsilon_{cor}^{k+1} \\ \hat{\beta}^{k+1} & \text{if} \quad \alpha_{cor}^k \leq \epsilon_{cor}^{k+1} \text{ and } \beta_{cor}^{k} > \epsilon_{cor}^{k+1} \\ \rho^k & \text{otherwise,} \end{cases} \tag{6-25}$$

where ϵ_{cor} is a user defined cutoff point that defines what amount of correlation is necessary to be considered a valid estimate.

Chapter 7

Online ADMM (O-ADMM)

Online Alternating Direction Method of Multipliers (O-ADMM) is the application of ADMM to online problems. In online problems it is often necessary to have results at a high frequency, with little time for conventional implementations of ADMM to converge. In practice O-ADMM is not a specific method or variant of ADMM, instead being a group of methods which can reduce computation time, speed up convergence, and guarantee feasible run-time solutions, all of which are necessary for adequate online problems.

The chapter is separated into four main sections: first the application of ADMM to online problems in general is summarized in Section 7-1, second a method utilizing ADMM for online distributed motion planning is covered in Section 7-2, this is followed by Section 7-3 which includes methods to convexify difficult problems such that they can be solved in real-time, and finally Section 7-4 introduces methods to guarantee feasible run-time results.

7-1 Online ADMM in general

Unlike adaptive ADMM, which is covered in Chapter 6, there is no exact definition for what online ADMM is. In general, O-ADMM refers to the use of Alternating Direction Method of Multipliers (ADMM) in an online environment, with online environments usually referring to applications where the time-dependent nature of the system makes pre-computation infeasible. For autonomous driving this is the case, as it is not possible to precompute all potential trajectories for all possible scenarios that autonomous vehicles can encounter. That is not to say that no precomputation can be utilized by online problems. Precomputation, however, is often limited to a small subset of the entire problem, e.g. the use of motion primitives to simplify motion planning optimization problem.

Batch ADMM can be seen as the most simple online implementation of ADMM, it is essentially conventional ADMM being applied to a time-varying system, performing ADMM in batches until convergence. The applicability of batch ADMM is limited to slow systems, as the frequency is limited due to the requirement of convergence. In literature, O-ADMM generally

refer to ADMM variants which do not perform the same steps until convergence, making them more suitable for online optimization.

An early work mentioning the term online ADMM is [107], where online ADMM is proposed in the context of machine learning as a large scale optimization technique. The difference between the O-ADMM from [107] and conventional ADMM is in the x-update, where the function $f(\boldsymbol{x})$ is replaced by a time varying function $f_t(\boldsymbol{x})$, and an additional Bregman divergence[22] term is added. The method actively updates the cost function f_t in real time, instead of requiring the ADMM to be performed until convergence as is the case with batch ADMM. The O-ADMM method from [107] is claimed to have a $O(1/T)$ convergence rate, which is supported with empirical results. This method is however focused on improving the overall convergence rate, and does not take into account additional issues which can arise with online optimization.

Another early work on O-ADMM is [108], which utilizes stochastic optimization methods to facility online implementation of ADMM. This work proposed two variants of O-ADMM: the regularized dual averaging variant, and the online proximal gradient descent variant. The regularized dual averaging variant replaces the function $f(x)$ with a linear function $\bar{\boldsymbol{g}}_t^\top x$, which can be interpreted as a linear approximation of the cost function. This requires the algorithm to first compute a sub-gradient $\boldsymbol{g_t} \in \nabla f(\boldsymbol{x})$, which is then averaged resulting in $\bar{\boldsymbol{g}}_t$. The x-update is modified to cancel out the squared L2-norm term, allowing the $\boldsymbol{x}^{k+1}$ to be written in the form of a projection of x onto a convex set $\mathcal{X}$. This rewriting of the x-update, is what distinguishes the regularized dual averaging variant from [108] with the online ADMM variant proposed in [107]. The online proximal gradient descent variant, as the name suggests, uses proximal gradient descent to simplify the minimization steps. This variant also modifies the x-update, but instead of using the averaged sub-gradient $\bar{\boldsymbol{g}}_t$, it utilizes $\boldsymbol{g}_t$ directly. The proximal gradient descent method also utilizes a proximal term which penalizes the deviance of $\boldsymbol{x}_{t+1}$ from $\boldsymbol{x}_t$, where $\boldsymbol{x}_t$ is the solution $\boldsymbol{x}$ at time t. Both of these methods are claimed to have a convergence rate of $O(1/\sqrt{T})$, with a convergence rate of $O(\log(T)/T)$ for the proximal gradient descent variant when the cost function is strongly convex.

Other works which follow a similar approach to O-ADMM as [107] and [108] are [109] and [110]. These are not analyzed further as they do not sufficiently account for potential issues which can arise with constraint violations.

A more relevant branch of online ADMM is the application of ADMM to online motion planning. Some of the more relevant works are [23], which proposed an online ADMM scheme to achieve online distributed motion planning, [111], which presents a linear MPC scheme using ADMM for online optimization, and [112], which utilizes ADMM and MPC to solve the energy management problem of hybrid electric vehicles. The latter two works will not be explored further as they are intended for single vehicle cases. The online ADMM method from [23] is covered in Section 7-2. Another work worth mentioning is [113], which uses a dual decomposition approach instead of ADMM for distributed online motion planning.

[22]The Bregman divergence is a generalized term used to measure the distance between two points using strictly convex functions, a simple Bregman divergence is the squared Euclidean distance.

7-2 Online distributed motion planning for multi-vehicle system

In [23] a method is proposed for multi-vehicle online distributed motion planning, utilizing ADMM for the distributed part of the problem, and a receding horizon method for the online portion. The motion planning is achieved through trajectory optimization.

7-2-1 Problem Formulation

The main optimization problem is written in the form of

$$\begin{aligned} \min_{\boldsymbol{x}_i(t)} \quad & \sum_{i=1}^{N} J_i(\boldsymbol{x}_i(t)) \\ s.t. \quad & \boldsymbol{x}_i(t) \in \mathcal{X}_i, \\ & g_{i,j}(\boldsymbol{x}_i(t), \boldsymbol{x}_j(t)) = 0, \forall j \in \mathcal{N}_i, \\ & \forall i \in \{1, ..., N\}, \forall t \in [0, T], \end{aligned} \tag{7-1}$$

where $\boldsymbol{x}_i(\cdot)$ represents the trajectory of vehicle i, including its position, velocity, etc.; $\mathcal{X}_i$ is the feasible set of $\boldsymbol{x}_i$, i.e. the set of trajectories which do not violate the system dynamics, the initial and final states, and the environmental collision avoidance constraints. The function $g_{i,j}(\boldsymbol{x}_i(t), \boldsymbol{x}_j(t))$ is the complicating constraint between vehicles i and j, in this case this functions as a formation constraint and is implemented as

$$g_{i,j}(\boldsymbol{x}_i(t), \boldsymbol{x}_j(t)) = \boldsymbol{x}_i(t) - \boldsymbol{x}_j(t) - \Delta \boldsymbol{x}_{ij}$$

in [23]. The objective function used by Van Parys and Pipeleers is the integral of the L1-norm between the trajectory $\boldsymbol{x}_i(t)$ and the desired final states $\boldsymbol{x}_i^T$, i.e.

$$J_i(\boldsymbol{x}_i(t)) = \int_0^T \|\boldsymbol{x}_i(t) - \boldsymbol{x}_i^T\|_1 dt.$$

The approach used for motion planning in [23] involves the use of spline parameterization, which is achieved through the use of B-splines. One of the properties of B-splines is that it is implied that the B-spline is always in the interior of the convex hull of its coefficients. This does however requires that trajectories are approximated using polynomials, which is possible for non-holonomic vehicle models like the bicycle car model. The overall optimization problem now optimizes the spline coefficients $\boldsymbol{\beta}$ instead of the exact trajectory, with the resulting equation being

$$\begin{aligned} \min_{\boldsymbol{\beta}_i} \quad & \sum_{i=1}^{N} J_i(\boldsymbol{\beta}_i) \\ s.t. \quad & \boldsymbol{\beta}_i \in \mathcal{B}_i, \\ & g_{i,j}(\boldsymbol{\beta}_i, \boldsymbol{\beta}_j) = 0, \forall j \in \mathcal{N}_i, \\ & \forall i \in \{1, ..., N\}. \end{aligned} \tag{7-2}$$

7-2-2 Methodology

In order for the problem to be solved in a distributed fashion, the complicating constraint of $g_{i,j}$ needs to be decoupled. This is achieved through ADMM, which requires the rewriting of Equation (7-2) into a form containing a slack vector $\boldsymbol{z}$, which is essentially a copy of the planned B-spline coefficients $\boldsymbol{\beta}$. The introduction of these slack variables results in the problem being reformulated as:

$$\begin{aligned} \min_{\boldsymbol{\beta}_i, \boldsymbol{z}_{ii}, \boldsymbol{z}_{ij}} \quad & \sum_{i=1}^{N} J_i(\boldsymbol{\beta}_i) \\ s.t. \quad & \boldsymbol{\beta}_i \in \mathcal{B}_i, \\ & g_{i,j}(\boldsymbol{z}_{ii}, \boldsymbol{z}_{ij}) = 0, \forall j \in \mathcal{N}_i, \\ & \boldsymbol{\beta}_i = \boldsymbol{z}_{ii}, \boldsymbol{\beta}_j = \boldsymbol{z}_{ij}, \forall j \in \mathcal{N}_i, \\ & \forall i \in \{1, ..., N\}. \end{aligned} \tag{7-3}$$

The addition of the slack vector $\boldsymbol{z}$ allows the problem of Equation (7-3) to be solved through ADMM, as the overall problem now takes the form of Equation (5-6), where the $g_{i,j}(\boldsymbol{z}_i, \boldsymbol{z}_j) = 0$ constraint is now independent of $\boldsymbol{\beta}$, the $\boldsymbol{\beta}_i \in \mathcal{B}_i$ constraint is independent of $\boldsymbol{z}$ and the complicating constraint is $\boldsymbol{\beta}_i = \boldsymbol{z}_i$ and$\boldsymbol{\beta}_j = \boldsymbol{z}_{ij}$, which is of a similar form as the $\boldsymbol{Ax} + \boldsymbol{Bz} = \boldsymbol{c}$ constraint in Equation (5-6). The ADMM Augmented Lagrangian for the overall problem of Equation (7-3) can now be formulated as

$$\mathcal{L}_\rho = \sum_{i=1}^{N} \left(J_i(\boldsymbol{\beta}_i) + \boldsymbol{\lambda}_i^\top (\boldsymbol{\beta}_i - \boldsymbol{z}_{ii}) + \frac{\rho}{2} \|\boldsymbol{\beta}_i - \boldsymbol{z}_{ii}\|_2^2 + \sum_{j \in \mathcal{N}_i} \left(\boldsymbol{\lambda}_{ij}^\top (\boldsymbol{\beta}_j - \boldsymbol{z}_{ij}) + \frac{\rho}{2} \|\boldsymbol{\beta}_j - \boldsymbol{z}_{ij}\|_2^2 \right) \right), \tag{7-4}$$

where $\boldsymbol{\lambda}$ is the Lagrange multiplier in vector form, ρ can be interpreted as the weight of the feasibility of the original problem, and $\boldsymbol{z}$ contains the copies of the trajectories $\boldsymbol{\beta}$. There are separate Lagrange multiplier vectors ($\boldsymbol{\lambda}$) for the different vehicles, with $\boldsymbol{\lambda}_i$ containing the Lagrange multipliers for the trajectory of vehicle i itself, and $\boldsymbol{\lambda}_{ij}$ containing the Lagrange multipliers for the vehicle i's trajectory copy of vehicle j.

The Lagrangian of the whole problem given in Equation (7-4) is then used for the ADMM steps from [23]. Given that the method proposed is online and distributed the Lagrangian from Equation (7-4) will not be used in its entirety for each agent, instead each agent individually performs a trajectory optimization step as follows:

$$\begin{aligned} \boldsymbol{\beta}_i^{k+1} := \arg\min_{\boldsymbol{\beta}_i} \quad & \mathcal{L}_{\rho,\boldsymbol{\beta},i} \\ s.t. \quad & \boldsymbol{\beta}_i \in \mathcal{B}_i, \end{aligned} \tag{7-5}$$

where each vehicles optimizes their own trajectory subject to the feasibility constraints, while ignoring the complicating constraint $g(\cdot)$, however taking into account the copies other vehicles have of i. The trajectory optimization step in Equation (7-5) can be performed in parallel on each agent, allowing it to scale well with many agents. With the Lagrangian $\mathcal{L}_{\rho,\beta,i}$ from Equation (7-5) being

$$\mathcal{L}_{\rho,\beta,i} = J_i(\boldsymbol{\beta}_i) + \boldsymbol{\lambda}_i^{k\top} (\boldsymbol{\beta}_i - \boldsymbol{z}_{ii}^k) + \frac{\rho}{2} \|\boldsymbol{\beta}_i - \boldsymbol{z}_{ii}^k\|_2^2 + \sum_{j \in \mathcal{N}_i} \left(\boldsymbol{\lambda}_{ji}^{k\top} (\boldsymbol{\beta}_i - \boldsymbol{z}_{ji}^k) + \frac{\rho}{2} \|\boldsymbol{\beta}_i - \boldsymbol{z}_{ji}^k\|_2^2 \right). \tag{7-6}$$

The newly found trajectory $\boldsymbol{\beta}_i^{k+1}$ is then communicated to the nearby agents and used for the copy optimization step:

$$\begin{aligned} \boldsymbol{z}_{IJ}^{k+1} := \underset{\boldsymbol{z}_{IJ}}{\arg\min} \quad & \mathcal{L}_{\rho,z,i} \\ s.t. \quad & g_{i,j}(\boldsymbol{\beta}_i^{k+1}, \boldsymbol{\beta}_j^{k+1}) = 0, \ \forall j \in \mathcal{N}_i, \end{aligned} \tag{7-7}$$

where $\boldsymbol{z}_{IJ}^{k+1}$ for the sake of writing convenience includes both $\boldsymbol{z}_{ii}$, which is a copy of the ego vehicles trajectory, and $\boldsymbol{z}_{ij}, \forall j \in \mathcal{N}_i$ which is a copy of all trajectories from neighboring vehicles. This step is also performed in parallel, with each vehicle optimizing $\boldsymbol{z}_{IJ}$ in order to satisfy the $g_{i,j}(\cdot) = 0$ constraint. The Lagrangian $\mathcal{L}_{\rho,z,i}$ in Equation (7-7) when fully written out is

$$\mathcal{L}_{\rho,z,i} = J_i(\boldsymbol{\beta}_i^{k+1}) + \boldsymbol{\lambda}_i^{k\top}(\boldsymbol{\beta}_i^{k+1} - \boldsymbol{z}_{ii}) + \frac{\rho}{2}\|\boldsymbol{\beta}_i^{k+1} - \boldsymbol{z}_{ii}\|_2^2 + \sum_{j \in \mathcal{N}_i} \left(\boldsymbol{\lambda}_{ij}^{k\top}(\boldsymbol{\beta}_j^{k+1} - \boldsymbol{z}_{ij}) + \frac{\rho}{2}\|\boldsymbol{\beta}_j^{k+1} - \boldsymbol{z}_{ij}\|_2^2 \right), \tag{7-8}$$

note that $J_i(\boldsymbol{\beta}_i^{k+1})$ can be left out as it is a constant which does not affect the result of $\boldsymbol{z}_{IJ}^{k+1}$. The $\boldsymbol{\lambda}$ vectors can then be updated as follows:

$$\begin{aligned} \boldsymbol{\lambda}_i^{k+1} &:= \boldsymbol{\lambda}_i^k + \rho(\boldsymbol{\beta}_i^{k+1} - \boldsymbol{z}_{ii}^{k+1}), \\ \boldsymbol{\lambda}_{ij}^{k+1} &:= \boldsymbol{\lambda}_{ij}^k + \rho(\boldsymbol{\beta}_j^{k+1} - \boldsymbol{z}_{ij}^{k+1}), \quad \forall j \in \mathcal{N}_i, \end{aligned} \tag{7-9}$$

where $\boldsymbol{\lambda}_i^{k+1}$ penalizes the deviation between the planned trajectory of the ego vehicle, $\boldsymbol{\beta}_i^{k+1}$, and its copy, $\boldsymbol{z}_i^{k+1}$, where $\boldsymbol{\beta}_i^{k+1}$ satisfies the ego vehicle feasibility constraints, and $\boldsymbol{z}_{ii}^{k+1}$ satisfies the complicating constraint $g(\cdot)$. By updating the Lagrange multiplier $\boldsymbol{\lambda}$, the optimal $\boldsymbol{\beta}_i^{k+1}$ from Equation (7-5) is adjusted in the direction of the copy $\boldsymbol{z}_{ii}^k$. The rate at which this adjustment occurs depends on the value of ρ, with higher values increasing the rate. Higher values of ρ can thereby be seen as attributing more value towards the complicating constraint $g(\cdot)$, whereas lower values of ρ can be interpreted as attributing more value towards minimum cost for individual vehicles when ignoring the complicating constraint. A diagram of the online ADMM method proposed in [23] that summarizes the online ADMM steps is given in Figure 7-1.

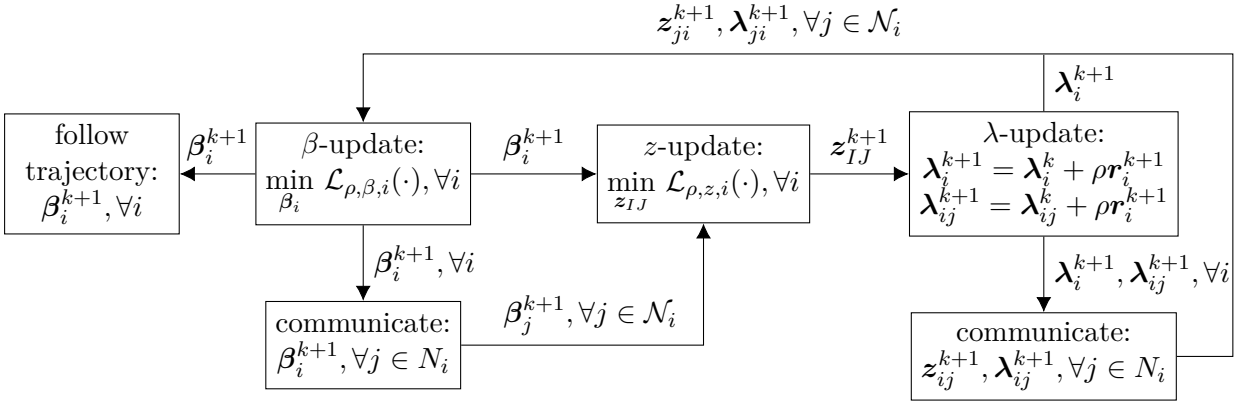


Figure 7-1: Diagram of Online ADMM steps, see Section 7-2-2 for more details.

7-2-3 Convergence

As the method in [23] has a different form than that of conventional ADMM, the convergence proof of ADMM, which is also given in Section 5-2-1, does not guarantee convergence. Van Parys and Pipeleers [23] note that despite the application of ADMM on non-convex problems, convergence is still observed through simulations. The use of ADMM for non-convex problems is not uncommon, for example see [114].

When considering the online distributed motion planning problem convergence is important, especially since the complicating constraint ($g_{i,j}(\cdot)$) is traditionally only guaranteed to be feasible when ADMM has fully converged. In the case of [23], the complicating constraint is responsible for formation control. If the formation is not properly maintained, collision between vehicles can occur which must be avoided.

As the original authors only provide empirical results for the convergence of the algorithm, it is not safe to say that this method will converge in general. The main cause of concern is the fact that the algorithm performs only one ADMM step per iteration, whilst actively following the trajectories from Equation (7-5). If the system is considered to be static, i.e. infinite ADMM iterations can be performed during each iteration, we can ignore the difficulty of proving online ADMM and first prove the convergence of the distributed algorithm. This could use a similar approach to the Lyapunov's direct method approach shown in Section 5-2-1.

7-3 Linearization and Convexification

Linearization and convexification is often necessary for Online-ADMM, or online optimization in general, due to the difficulty of solving non-convex and nonlinear optimization problems; the main computational burden of ADMM lays in the x and z optimizations steps. Solving a quadratic programming problem is much easier and faster than solving a non-convex nonlinear problem, this does however have the downside that often multiple iterations have to be performed, as the linearized and convexification will almost never directly result in the same optimum.

A simple overview on linearization is given in Section 7-3-1, followed by a summary on Convex Feasible Set in Section 7-3-2, which is an approach for convexifying non-convex constraints.

7-3-1 Linearization

The simplest method to ensure convexity of a function is to linearize it, this is often used for nonlinear inequality constraints, which are too difficult to solve in real-time. Generally, linearization is performed by finding a linear function which approximates the nonlinear function the closest. The most common method to achieve this is the first order Taylor expansion.

For a nonlinear function $f(x)$, the first order Taylor expansion is

$$\hat{f}(x) = f(a) + f'(a)(x - a), \tag{7-10}$$

where $\hat{f}(x)$ is the linear approximation, f' is the first derivative of f, and a is the linearization point. This ensures that the $\hat{f}(a) = f(a)$ and $\hat{f}'(a) = f'(a)$, i.e. the value and derivative of

the linear approximation $\hat{f}$ is identical to that of the original function f in the linearization point.

Given that the accuracy of the linearization is highly dependent on the distance from the linearization point, it is often necessary for the linearization to take place every time step. This ensures that the linearization remains as accurate as possible, whilst avoiding the additional computational costs of nonlinear constraints.

To give an example we will show the linearization of a distance function $d(\boldsymbol{p}_i, \boldsymbol{p}_j)$, which is the squared Euclidean distance between two points p_i and p_j. The squared Euclidean distance is used as this simplifies the linearization, this does not affect its use as a minimum/maximum distance constraint, as the minimum/maximum distance is a constant which can squared. The to be linearized distance function is

$$d(\boldsymbol{p}_i, \boldsymbol{p}_j) = \|\boldsymbol{p}_i - \boldsymbol{p}_j\|_2^2 = \begin{bmatrix} \boldsymbol{p}_i^\top & \boldsymbol{p}_j^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_i \\ \boldsymbol{p}_j \end{bmatrix}, \tag{7-11}$$

which we will linearize around $\bar{\boldsymbol{p}}_i, \bar{\boldsymbol{p}}_j$ using the first order Taylor expansion given in Equation (7-10), resulting in

$$\begin{aligned} \bar{d}(\boldsymbol{p}_i, \boldsymbol{p}_j) &= d(\bar{\boldsymbol{p}}_i, \bar{\boldsymbol{p}}_j) + d'(\bar{\boldsymbol{p}}_i, \bar{\boldsymbol{p}}_j)\left(\begin{bmatrix} \boldsymbol{p}_i \\ \boldsymbol{p}_j \end{bmatrix} - \begin{bmatrix} \bar{\boldsymbol{p}}_i \\ \bar{\boldsymbol{p}}_j \end{bmatrix} \right) \\ &= \begin{bmatrix} \bar{\boldsymbol{p}}_i^\top & \bar{\boldsymbol{p}}_j^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{p}}_i \\ \bar{\boldsymbol{p}}_j \end{bmatrix} + 2 \begin{bmatrix} \bar{\boldsymbol{p}}_i^\top & \bar{\boldsymbol{p}}_j^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{I} \end{bmatrix} \left(\begin{bmatrix} \boldsymbol{p}_i \\ \boldsymbol{p}_j \end{bmatrix} - \begin{bmatrix} \bar{\boldsymbol{p}}_i \\ \bar{\boldsymbol{p}}_j \end{bmatrix} \right), \end{aligned} \tag{7-12}$$

where $\bar{d}(\boldsymbol{p}_i, \boldsymbol{p}_j)$ is the linearized function. This distance function is non-convex when used for a minimum distance constraint, i.e.

$$d(\boldsymbol{p}_i, \boldsymbol{p}_j) \geq D_{min}^2,$$

where D_{min} is a specified minimum distance between the points p_i and p_j. The function is non-convex given that $d(\boldsymbol{p}_i, \boldsymbol{p}_j) < 0$, which is the opposite of the minimum distance constraint, is convex. This can also be easily seen in Figure 7-2.

7-3-2 Convex Feasible Set Algorithm

Convex Feasible Set (CFS) is an algorithm proposed by Liu *et al.* in [103] and [102] as an algorithm for real-time optimization in motion planning. The algorithm is intended for non-convex optimization problems, where the cost function is convex, with the constraints non-convex. The algorithm is based on the idea that solving an easier convexified problem iteratively, is faster than solving the difficult non-convex problem once. In terms of solving a difficult problem using convex subproblems, this is similar to Sequential Quadratic Programming (SQP), the main difference being that CFS takes the geometric structure of the original problem into account.

As an example we take a problem of the form

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & J(\boldsymbol{x}), \\ s.t. \quad & \boldsymbol{x} \in \mathcal{X}^f, \end{aligned} \tag{7-13}$$

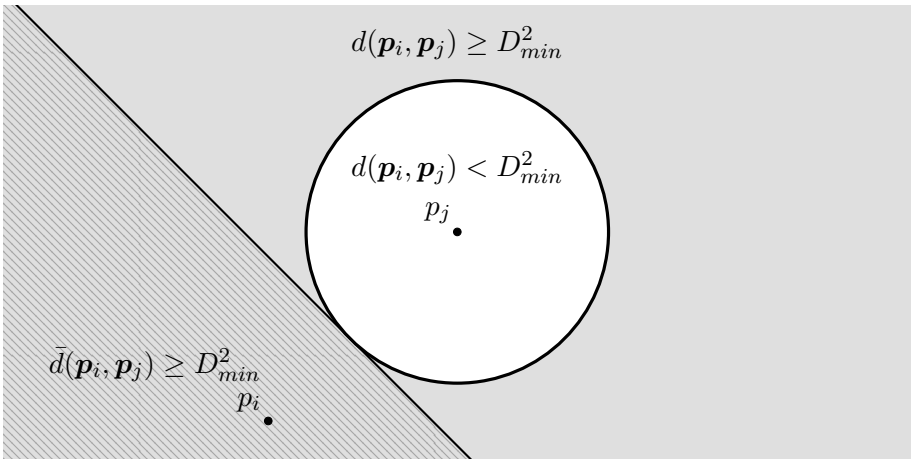


Figure 7-2: Figure showing the feasible regions for the maximum distance function $d(\boldsymbol{p}_i, \boldsymbol{p}_j) < D^2_{min}$ (white region), the minimum distance function $d(\boldsymbol{p}_i, \boldsymbol{p}_j) \geq D^2_{min}$ (gray region), and the linearized minimum distance function $\bar{d}(\boldsymbol{p}_i, \boldsymbol{p}_j) \geq D^2_{min}$ (hatched region).

where $J(\boldsymbol{x})$ is the convex cost function, and $\mathcal{X}^f$ is the non-convex feasible set. Two assumptions have to be made regarding this problem: first that cost function has to be radially unbounded, i.e. $J(\boldsymbol{x}) \to \infty$ when $\|\boldsymbol{x}\|_2$; second, the non-convex feasible set is required to be one connected and closed set, with a boundary that is piecewise smooth and non-self-intersecting. There is an additional requirement that there exists a polytope P for every point $\boldsymbol{x} \in \mathcal{X}^f$. These assumptions are to ensure the feasibility of CFS to iteratively find the optimum, for the exact mathematical description of the assumptions please refer to [103].

The CFS approach first requires a convex feasible subset ($\mathcal{X}^f_C$) of $\mathcal{X}^f$, i.e. $\mathcal{X}^f_C \subset \mathcal{X}^f$. A simple approach to finding a convex subset $\mathcal{X}^f_C$ is to linearize the non-convex constraints as is done in Section 7-3-1. This approach is also used by CFS when the infeasible set is convex. When the infeasible set is concave, the feasible set is per definition convex, which gives $\mathcal{X}^f_C = \mathcal{X}^f$. If the infeasible set is neither concave nor convex, the feasible subset is found using

$$\mathcal{X}^f_C := \{\boldsymbol{x} : \phi(\bar{\boldsymbol{x}}) + \hat{\nabla}\phi(\bar{\boldsymbol{x}})(\boldsymbol{x} - \bar{\boldsymbol{x}}) \geq \frac{1}{2}(\boldsymbol{x} - \bar{\boldsymbol{x}})^\top \boldsymbol{H}^*(\boldsymbol{x} - \bar{\boldsymbol{x}})\}, \tag{7-14}$$

where $\bar{x}$ is the reference point or linearization point and ϕ is the boundary function of the infeasible set, with $\phi = 0$ on the boundary, and $\phi > 0$ in the feasible region. The set definition given in Equation (7-14) can be seen as a more conservative linearization, where the linearized constraint is shifted by $(\boldsymbol{x} - \bar{\boldsymbol{x}})^\top \boldsymbol{H}^*(\boldsymbol{x} - \bar{\boldsymbol{x}})$ to ensure the feasibility of the convexified subset.

Once an appropriate convex feasible subset is found, the convexified optimization problem can be solved, with the solution used as the reference for the next iteration. This is repeated until the goal is reached with small enough residuals. The CFS algorithm is also given in summarized form in Algorithm 7.1. For proofs or more information on the method please see [103], or see [102] for an example of CFS applied to trajectory smoothing.

Algorithm 7.1: Convex Feasible Set (CFS) Algorithm.

initialize $\boldsymbol{x}^0$
while goal() $\neq$ TRUE **do**
  Find $\mathcal{X}_C^f \subset \mathcal{X}^f$ given $\boldsymbol{x}^k$
  $\boldsymbol{x}^{k+1} = \arg\min_{\boldsymbol{x} \in \mathcal{X}_C} J(\boldsymbol{x})$
  $k = k + 1$
end

7-4 Online Feasibility

During online optimization, it is important that feasibility is maintained, as this can otherwise lead to infeasible solutions or unforeseen behavior. How this can be achieved is however very dependent on the application and the constraints of the problem. If there are no time-dependent constraints, this is achieved by ADMM if it is performed until convergence, which is essentially batch ADMM.

Given that the intended application is autonomous vehicles it is assumed that the online optimization in question is Model Predictive Control (MPC); online feasibility is also referred to as recursive feasibility or persistent feasibility in the field of MPC. Commonly, online feasibility of MPC is addressed by properly choosing terminal constraints; closed-loop stability is addressed by properly choosing terminal costs. This method does however not provide a mathematical guarantee of online feasibility, and is instead a more practical approach which is analyzed in [115]. When the uncertainties of the system is bounded, online feasibility can be proven using invariance set theory [116].

An elaborate discussion on achieving online feasibility of MPC or online optimization in general is beyond the scope of this thesis, we do however stress that there is a large amount of literature available on this topic, from which we will briefly summarize a few relevant works. When a theoretical proof for online feasibility is not available, analysis can be done to find the problematic states where recursive feasibility is lost [117]. [118] provides a stability proof for nonlinear MPC by applying a monotonically increasing stage cost penalty, instead of a singular terminal cost as is commonly used. MPC with time-varying and uncertain state constraints is analyzed in [119], where online feasibility and asymptotic stability can be found when the change of the constraints is bounded or a model is available for the change of the constraints.

Chapter 8

Online Adaptive-ADMM (OA-ADMM)

In this chapter we propose a novel ADMM-based method/strategy, which gives the user more control over the online convergence of certain states and constraints compared with other existing methods. This property is mainly achieved through the introduction of the adaptation function ϕ and the vectorization of ρ. Note that this chapter focuses on the general form of Online Adaptive Alternating Direction Method of Multipliers (OA-ADMM), and does not contain examples or design tips for the adaptation function ϕ, for those please refer to Chapter 9. The chapter starts with Section 8-1, describing the problems OA-ADMM is designed to handle. It is then followed by Section 8-2, which contains the methodology of OA-ADMM. Finally, the convergence results and proofs are given in Section 8-3.

8-1 Problem Formulation

Online Adaptive ADMM is designed for problems with the following properties:

- The problem cannot be solved fast enough using conventional optimization methods in order to achieve a desired control frequency.
- It is necessary that the relative importance of the constraints for run-time solutions can be designed to meet the demands of the user.

Given the first assumption, a method suitable for online optimization needs to be used: this is achieved by OA-ADMM by applying an ADMM-based strategy which can achieve permissible online results. The second assumption is satisfied by designing an adaptive penalty parameter ρ which allows a form of prioritization of the constraint violations in online results. One of these type of problems are multi-agent vehicle trajectory optimization problems, where the collision avoidance between vehicles prevents the use of conventional optimization methods. The strategy used by OA-ADMM can however be applied to any problem where online (multi-agent) optimization is required.

The main optimization problems tackled by the proposed method can be formulated as

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{z}} \quad & f(\boldsymbol{x}) + g(\boldsymbol{z}) \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} = \boldsymbol{c}, \end{aligned} \tag{8-1}$$

where $f(\boldsymbol{x})$ and $g(\boldsymbol{z})$ are convex functions, and the variables have the following dimensions: $\boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{z} \in \mathbb{R}^m, \boldsymbol{A} \in \mathbb{R}^{p\times n}, \boldsymbol{B} \in \mathbb{R}^{p\times m}$, and $\boldsymbol{c} \in \mathbb{R}^p$. This is the most basic and general form of problems that OA-ADMM can handle. For an example that better showcases the strengths of OA-ADMM, see Chapter 9 where the problem of decentralized conflict resolution problem for autonomous vehicles is solved using OA-ADMM.

8-2 Methodology

Given that OA-ADMM is a variant of Alternating Direction Method of Multipliers (ADMM) it also has a similar structure and requirements as conventional ADMM. The main difference between OA-ADMM and conventional ADMM in terms of execution is the use of a dynamic penalty vector and function, i.e. $\boldsymbol{\rho} = \phi(\cdot)$, instead of the conventional static penalty parameter ρ, combined with the application to real-time systems.

Like in all ADMM methods the coupled constraints are integrated into an augmented Lagrangian in order to separate the problem, the augmented Lagrangian for OA-ADMM is

$$\mathcal{L}_{\boldsymbol{\rho}}(\boldsymbol{x},\boldsymbol{z},\boldsymbol{\lambda}) = f(\boldsymbol{x}) + g(\boldsymbol{z}) + \boldsymbol{\lambda}^\top(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c}) + \frac{1}{2}\|\boldsymbol{R}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c})\|_2^2. \tag{8-2}$$

where $\boldsymbol{R} \in \mathbb{R}^{p\times p}$ is a diagonal matrix with $\text{diag}(\boldsymbol{R}) = \boldsymbol{\rho}^{\circ\frac{1}{2}}$. The operator denoted by $(\cdot)^\circ$ is the *Hadamard power* (or element-wise power). The use of the matrix $\boldsymbol{R}$ is to simplify notation of a vectorized ρ, which can no longer be a scalar outside of the L2-norm. When the vector $\boldsymbol{\rho}$ is set to be $\boldsymbol{\rho} = \rho_s\mathbf{1}$, where ρ_s is the conventional penalty parameter with the vector $\mathbf{1} \in \mathbb{R}^p = 1, ..., 1$, the augmented Lagrangian given in Equation (8-2) becomes the exact same as the augmented Lagrangian for conventional ADMM given in Equation (5-7). The use of a penalty vector $\boldsymbol{\rho}$ therefore does not fundamentally differ in function from the regular penalty parameter ρ. Rather, the penalty vector has the added benefit of applying the penalty to each element of the primal residual $\boldsymbol{r}$ separately.

The structure used by OA-ADMM is similar to other adaptive ADMM methods, with two optimization steps, a Lagrangian multiplier update, and a ρ update. The overall structure therefore is:

$$\boldsymbol{x}^{k+1} := \arg\min_{\boldsymbol{x}} \mathcal{L}_{\boldsymbol{\rho}}(\boldsymbol{x}, \boldsymbol{z}^k, \boldsymbol{\lambda}^k, \boldsymbol{\rho}^k), \tag{8-3a}$$

$$\boldsymbol{z}^{k+1} := \arg\min_{\boldsymbol{z}} \mathcal{L}_{\boldsymbol{\rho}}(\boldsymbol{x}^{k+1}, \boldsymbol{z}, \boldsymbol{\lambda}^k, \boldsymbol{\rho}^k), \tag{8-3b}$$

$$\boldsymbol{\lambda}^{k+1} := \boldsymbol{\lambda}^k + \boldsymbol{\rho}^k \circ \boldsymbol{r}^{k+1}, \tag{8-3c}$$

$$\boldsymbol{\rho}^{k+1} := \phi(\cdot), \tag{8-3d}$$

where ϕ is a user defined adaptation function, and $\boldsymbol{r}^{k+1}$ is the primal residual. For the online case we assume that the problem at the next time step has not changed significantly from

the previous time step. This allows the next time step to reuse the Lagrangian multiplier $\boldsymbol{\lambda}$ (and perhaps the penalty parameter $\boldsymbol{\rho}$) to some extent. This is performed using a function $\mu(\cdot)$, which represents the similarity between the previous time step and the current time step, e.g. if the system in the previous time step is identical to the system in the current time step $\mu(\cdot) = 1$, if the system is very dissimilar, $\mu(\cdot) \approx 0$. The function should always return a value between zero and one and can be designed to fit the needs of the user. It is possible for OA-ADMM to perform one to multiple steps of Equation (8-3) each controller time step depending on the desired control frequency and the difficulty of the problem; the amount of iterations per control step is defined by N_k. With the first OA-ADMM iteration sometimes being referred to as k_0, conversely the last iteration is then k_N. Regardless, it is necessary that the following step is performed once per every control time step:

$$\boldsymbol{\lambda}^{k+1} := \mu(\cdot)\boldsymbol{\lambda}^k + \boldsymbol{\rho}^k \circ \boldsymbol{r}^{k+1}, \tag{8-4}$$

where $\circ$ is the *Hadamard product* (or element-wise product) operator, note that $\mu(\cdot)$ returns a scalar here, however it is also possible to use a vectorized μ when certain constraints are desired to be scaled differently.

The overall pseudocode for OA-ADMM is given in Algorithm 8.1. If the algorithm is applied to a system with additional time dependent constraints, it is necessary to update these after each control loop. The main loop of the algorithm stops when either the predefined final time T_{final} has been reached, or the desired goal has been reached. The control applied in Line 10 is where the online control is performed, for example if the problem is a Model Predictive Control problem, the control will apply the input given in x^k. After the control is applied, the problem is updated for the current time-step and the solutions of the previous time-step are used to initialize OA-ADMM.

Algorithm 8.1: OA-ADMM Pseudocode.

1 initialize $\boldsymbol{x}^0, \boldsymbol{z}^0, \boldsymbol{\lambda}^0, \boldsymbol{\rho}^0$
2 **while** $t \leq T_{final}$ and goal() $\neq$ TRUE **do**
3 **while** $k \leq N_k$ and $\|\boldsymbol{r}^k\|_2 > \epsilon_p$ and $\|\boldsymbol{s}^k\|_2 > \epsilon_s$ **do**
4 $\boldsymbol{x}^{k+1} = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) + {\boldsymbol{\lambda}^k}^\top(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z}^k - \boldsymbol{c}) + \frac{1}{2}\|\boldsymbol{R}^k(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z}^k - \boldsymbol{c})\|_2^2$
5 $\boldsymbol{z}^{k+1} = \arg\min_{\boldsymbol{z}} g(\boldsymbol{z}) + {\boldsymbol{\lambda}^k}^\top(\boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c}) + \frac{1}{2}\|\boldsymbol{R}^k(\boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c})\|_2^2$
6 $\boldsymbol{\lambda}^{k+1} := \boldsymbol{\lambda}^k + \boldsymbol{\rho}^k \circ \boldsymbol{r}^{k+1}$,
7 $\boldsymbol{\rho}^{k+1} = \phi(\cdot)$
8 $k = k + 1$
9 **end**
10 Apply control(x^k)
11 $t = t + \delta t$
12 Update problem
13 set $\boldsymbol{x}^0 = \boldsymbol{x}^k, \boldsymbol{z}^0 = \boldsymbol{z}^k, \boldsymbol{\lambda}^0 = \mu(\cdot)\boldsymbol{\lambda}^k, \boldsymbol{\rho}^0 = \boldsymbol{\rho}^k$
14 $k = 0$
15 **end**

8-2-1 Stopping Criteria

The stopping criteria for OA-ADMM are mostly similar to that of regular ADMM and will therefore be briefly summarized. The primal residual and dual residuals are

$$\boldsymbol{r}^{k+1} = \boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z}^{k+1} - \boldsymbol{c}, \tag{8-5}$$

and the dual residual being

$$\boldsymbol{s}^{k+1} = \boldsymbol{A}^\top \boldsymbol{\rho}^k \circ \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k). \tag{8-6}$$

Given that the dual residual involves the vectorized ρ, we will repeat the derivation of the dual residual for OA-ADMM. The dual residual Equation (8-6) results from the fact that $\boldsymbol{x}^{k+1}$ is a minimizer for $\mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{z}^k, \boldsymbol{\lambda}^k, \boldsymbol{\rho}^k)$, i.e.

$$\begin{aligned}
0 &\in \partial \mathcal{L}_{\boldsymbol{\rho}}(\boldsymbol{x}, \boldsymbol{z}^k, \boldsymbol{\lambda}^k) \\
&= \partial f(\boldsymbol{x}^{k+1}) + \boldsymbol{A}^\top \boldsymbol{\lambda}^k + \boldsymbol{A}^\top \boldsymbol{\rho}^k \circ (\boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z}^k - \boldsymbol{c}) \\
&= \partial f(\boldsymbol{x}^{k+1}) + \boldsymbol{A}^\top (\boldsymbol{\lambda}^k + \boldsymbol{\rho}^k \circ \boldsymbol{r}^{k+1}) + \boldsymbol{A}^\top \boldsymbol{\rho}^k \circ \boldsymbol{B}(\boldsymbol{z}^k - \boldsymbol{z}^{k+1}) \\
&= \partial f(\boldsymbol{x}^{k+1}) + \boldsymbol{A}^\top \boldsymbol{\lambda}^{k+1} + \boldsymbol{A}^\top \boldsymbol{\rho}^k \circ \boldsymbol{B}(\boldsymbol{z}^k - \boldsymbol{z}^{k+1}),
\end{aligned}$$

which when reformulated as

$$\boldsymbol{A}^\top \boldsymbol{\rho}^k \circ \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k) \in \partial f(\boldsymbol{x}^{k+1}) + \boldsymbol{A}^\top \boldsymbol{\lambda}^{k+1}, \tag{8-7}$$

results in the optimality condition of $\boldsymbol{A}^\top \boldsymbol{\rho}^k \circ \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k) = 0$, which gives Equation (8-6).

Similarly to ADMM, the actual termination conditions are based on acceptable tolerances for both residuals, i.e.

$$\|\boldsymbol{r}^k\|_2 \leq \epsilon_p \quad \text{and} \quad \|\boldsymbol{s}^k\|_2 \leq \epsilon_d, \tag{8-8}$$

where ϵ_p and ϵ_s are the primal and dual residual tolerances respectively.

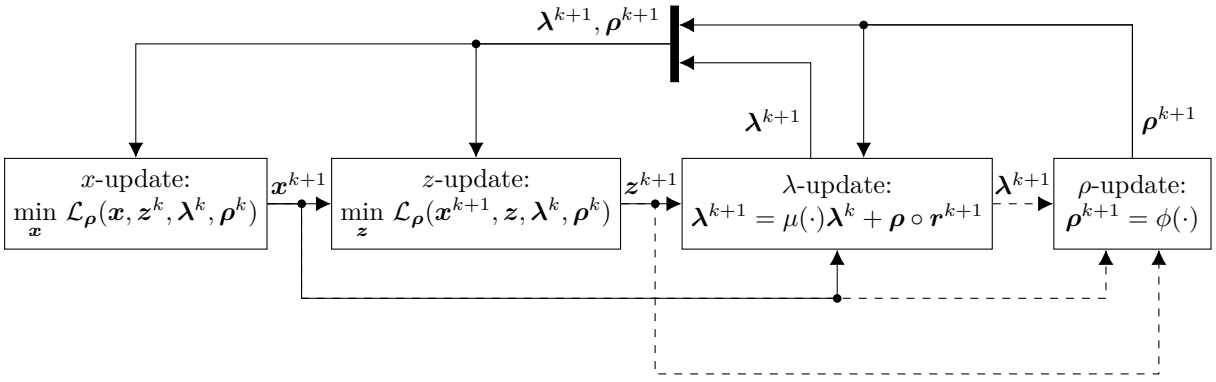


Figure 8-1: Diagram of OA-ADMM steps, dashed arrows resemble potential, but not necessary, inputs.

8-3 Convergence Results

The convergence results for OA-ADMM are split into two subsections, the first (Section 8-3-1) contains the converge results and proofs for a static case, the second (Section 8-3-2) describes the requirements for proving online convergence and suggests approaches for achieving online convergence.

8-3-1 Static Case

In a static case we assume that there are no real-time requirements for the problem, allowing (almost) infinite time/iterations for OA-ADMM. We acknowledge that it is indeed counter intuitive for OA-ADMM to be applied to static problems, however this step is necessary for further online convergence arguments. First we intend to prove that OA-ADMM converges as $k \to \infty$, this corresponds to $N_k = \infty$ for Algorithm 8.1. $N_k = \infty$ corresponds to a static case.

The convergence proof for ADMM, which is given in Section 5-2-1, is used as the foundation for proving the convergence of OA-ADMM. The major changes between OA-ADMM and ADMM for a static problem are the use of an adaptive penalty parameter ρ, which is defined by the function ϕ, and the vectorization of ρ.

In order to prove convergence, a candidate Lyapunov function, i.e. a function V where $V \geq 0$ with $V = 0$ only at saddle point, is proposed:

$$V^k = \|\boldsymbol{R}^{\circ -1}(\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^\star)\|_2^2 + \|\boldsymbol{R}\boldsymbol{B}(\boldsymbol{z}^k - \boldsymbol{z}^\star)\|_2^2, \tag{8-9}$$

alongside with the assumption that the original Lagrangian has a saddle point at $(\boldsymbol{x}^\star, \boldsymbol{z}^\star, \boldsymbol{\lambda}^\star)$.

Lemma 8-3.1 (Converging penalty parameter)**.** *If $\boldsymbol{\rho}^k \to \boldsymbol{\rho}^\star$ as $k \to \infty$, then the convergence results for a static $\boldsymbol{\rho}$ hold for the dynamic $\boldsymbol{\rho}$ as well.*

Lemma 8-3.2 (Objective suboptimality lower bound)**.** *The suboptimality of the objective p is lower bounded by:*

$$p^\star - p^{k+1} \leq \boldsymbol{\lambda}^{\star\top} \boldsymbol{r}^{k+1}.$$

Lemma 8-3.3 (Objective suboptimality upper bound)**.** *The suboptimality of the objective p is upper bounded by:*

$$p^{k+1} - p^\star \leq -\boldsymbol{\lambda}^{k+1\top} \boldsymbol{r}^{k+1} - \left(\boldsymbol{\rho}^k \circ \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right)^\top \left(-\boldsymbol{r}^{k+1} + \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^\star)\right)$$

Lemma 8-3.4 (Lyapunov decrease)**.** *The Lyapunov-function V from Equation* (8-9) *decreases with each iteration, proportionally to the norm of the residual $\boldsymbol{r}^{k+1}$ and the change between $\boldsymbol{z}^k$ and $\boldsymbol{z}^{k+1}$, i.e.:*

$$V^{k+1} \leq V^k - \|\boldsymbol{R}^k \boldsymbol{r}^{k+1}\|_2^2 - \|\boldsymbol{R}^k \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\|_2^2, \tag{8-10}$$

Where in Equation (8-10), $\boldsymbol{R}$ is a diagonal matrix with $\boldsymbol{\rho}^{\circ \frac{1}{2}}$ on its diagonal, with $^\circ$ being the *Hadamard power* (or element-wise power) operator.

The convergence results for the static case can be summarized in the following theorem:

Theorem 8-3.1. *When applying the OA-ADMM algorithm of Equation* (8-3)*, given closed, proper, and convex functions f and g, a saddle point $p^\star$ in the Lagrangian $\mathcal{L}$, and a converging $\boldsymbol{\rho}^k$, it holds that the primal and dual residuals converge to zero, i.e. $\boldsymbol{r}^k \to 0$ and $\boldsymbol{s}^k \to 0$, and that the objective converges to the saddle point, i.e. $p^k \to p^\star$.*

Proof. Iterating Lemma 8-3.4 from $k = 0$ to $k = \infty$ gives

$$\sum_{k=0}^{\infty} \left(\|\boldsymbol{R}^k \boldsymbol{r}^{k+1}\|_2^2 + \|\boldsymbol{R}^k \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\|_2^2 \right) \leq V^0,$$

which simply states that for $k = \{0, ..., \infty\}$ the sum of the Lyapunov function is bounded, implying that $V^k \to 0$ as $k \to \infty$. Given that the Lyapunov function V is a sum of two squared L2-norms, it has to hold that both $\boldsymbol{R}^k \boldsymbol{r}^{k+1} \to 0$ and $\boldsymbol{R}^k \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k) \to 0$. Because $\boldsymbol{R}$ is a symmetric positive definite matrix, it also holds that $\boldsymbol{r}^{k+1} \to 0$ and $\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k) \to 0$, i.e. the primal and dual residuals converge to zero. Lemma 8-3.3 and Lemma 8-3.2 further provide an upper and lower bound for the objective suboptimality, i.e. $(p^{k+1} - p^\star)$. Both right-hand sides of Lemma 8-3.3 and Lemma 8-3.2 converge to zero as the residuals converge to zero, for that reason the objective suboptimality is shown to converge to zero as $k \to \infty$. Lemma 8-3.1 allows the use of static $\boldsymbol{\rho}$ convergence results as long as the dynamic ρ converges. □

Proof for Lemma 8-3.1

The convergence results of Lemma 8-3.1 are derived from the conditions given in Section 6-2-1. The two conditions put on the adaptive penalty parameter are Condition 6-2.1 and Condition 6-2.2, repeated here for convenience:

Condition 8-3.1 (Bounded increasing)**.** $\inf\{\rho^k\}_1^\infty > 0$ and $\sum_{k=1}^{\infty}(\eta^k)^2 < +\infty$, where

$$\eta^k = \sqrt{\max(\frac{\rho^{k+1}}{\rho^k}, 1) - 1}.$$

Condition 8-3.2 (Bounded decreasing)**.** $\sum_{k=1}^{\infty}(\theta^k)^2 < +\infty$, where

$$\theta^k = \sqrt{\max(\frac{\rho^k}{\rho^{k+1}}, 1) - 1}.$$

When these two conditions hold it is proven in [28] that ADMM achieves convergence for a dynamic ρ. If ρ converges to a certain $\rho^\star$, it will eventually hold that $\rho^{k+1} = \rho^k = \rho^\star$. When ρ^k has converged, $\eta^k = 0$ and $\theta^k = 0$ are true, which also results in $\sum_{k=1}^{\infty}(\eta^k)^2 < +\infty$ and $\sum_{k=1}^{\infty}(\theta^k)^2 < +\infty$ being true. Given that $\inf\{\rho^k\}_1^\infty > 0$ already is required for a static ρ, we have shown that Lemma 8-3.1 holds.

Proof for Lemma 8-3.2

The proof given for Lemma 8-3.2 is essentially identical to the proof given for Lemma 5-2.3, given that the vectorization of ρ does not affect the proof. For convenience we will briefly repeat

the proof: The saddle point of the original Lagrangian $\mathcal{L}$ is defined to be at $(\boldsymbol{x}^\star, \boldsymbol{z}^\star, \boldsymbol{\lambda}^\star)$, given the definition of the saddle point it holds that $\mathcal{L}(\boldsymbol{x}^\star, \boldsymbol{z}^\star, \boldsymbol{\lambda}^\star) \leq \mathcal{L}(\boldsymbol{x}^{k+1}, \boldsymbol{z}^{k+1}, \boldsymbol{\lambda}^\star)$. Furthermore, since $\boldsymbol{r}^\star = 0$ in the saddle point and $\mathcal{L}(\boldsymbol{x}^k, \boldsymbol{z}^k, \boldsymbol{\lambda}^k, \boldsymbol{\rho}^k) = f(\boldsymbol{x}^k) + g(\boldsymbol{z}^k) + {\boldsymbol{\lambda}^k}^\top \boldsymbol{r}^k = p^k + {\boldsymbol{\lambda}^k}^\top \boldsymbol{r}^k$, we get

$$p^\star \leq p^{k+1} + {\boldsymbol{\lambda}^\star}^\top \boldsymbol{r}^{k+1},$$

which proves Lemma 8-3.2

Proof for Lemma 8-3.3

The proof given in Section 5-2-1 utilizes the optimality condition for the $\boldsymbol{x}^{k+1}$ optimization step, which minimizes the augmented Lagrangian. With the augmented Lagrangian for OA-ADMM in general being

$$\mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}, \boldsymbol{\rho}) = f(\boldsymbol{x}) + g(\boldsymbol{z}) + \boldsymbol{\lambda}^\top(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c}) + \frac{1}{2}\|\boldsymbol{R}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c})\|_2^2. \tag{8-11}$$

The subgradient optimality condition results in the fact that the minimizer $\boldsymbol{x}^{k+1}$ results in zero being a subgradient of the augmented Lagrangian, i.e.

$$0 \in \partial\mathcal{L}_{\boldsymbol{\rho}}(\boldsymbol{x}^{k+1}, \boldsymbol{z}^k, \boldsymbol{\lambda}^k, \boldsymbol{\rho}^k) = \partial f(\boldsymbol{x}^{k+1}) + \boldsymbol{A}^\top\boldsymbol{\lambda}^k + \boldsymbol{A}^\top {\boldsymbol{R}^k}^\top \boldsymbol{R}^k(\boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z}^k - \boldsymbol{c}), \tag{8-12}$$

where the subdifferential is found through the sum of the subdifferential for the subdifferentiable function and the gradient for the differentiable function. The optimality condition of Equation (8-12) can be rearranged using the λ-update, i.e. $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \boldsymbol{\rho}^k \circ (\boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z}^{k+1} - \boldsymbol{c})$, into

$$0 \in \partial f(\boldsymbol{x}^{k+1}) + \boldsymbol{A}^\top\left(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\rho}^k \circ \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right), \tag{8-13}$$

where we make use of $\text{diag}(\boldsymbol{R}) = \boldsymbol{\rho}^{\circ\frac{1}{2}}$, which allows $\boldsymbol{R}^\top\boldsymbol{R} \cdot (...)$ to be written as $\boldsymbol{\rho} \circ (...)$. The result of this is that $\boldsymbol{x}^{k+1}$ is a minimizer for

$$f(\boldsymbol{x}) + \left(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\rho}^k \circ \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right)^\top \boldsymbol{A}\boldsymbol{x}. \tag{8-14}$$

The fact that $\boldsymbol{x}^{k+1}$ minimizes Equation (8-14) also implies that

$$f(\boldsymbol{x}^{k+1}) + \left(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\rho}^k \circ \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right)^\top \boldsymbol{A}\boldsymbol{x}^{k+1} \leq f(\boldsymbol{x}^\star) + \left(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\rho}^k \circ \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right)^\top \boldsymbol{A}\boldsymbol{x}^\star, \tag{8-15}$$

where $\boldsymbol{x}^\star$ is $\boldsymbol{x}$ in the saddle point. Equation (8-15) simply states that Equation (8-14) with $\boldsymbol{x} = \boldsymbol{x}^{k+1}$ is always lesser or equal to Equation (8-14) with $\boldsymbol{x} = \boldsymbol{x}^\star$, which has to be true given that $\boldsymbol{x}^{k+1}$ minimizes Equation (8-15).

The subgradient optimality condition can be applied in a similar way for $\boldsymbol{z}^{k+1}$, resulting in

$$0 \in \partial\mathcal{L}_{\boldsymbol{\rho}}(\boldsymbol{x}^{k+1}, \boldsymbol{z}^{k+1}, \boldsymbol{\lambda}^k, \boldsymbol{\rho}^k) = \partial g(\boldsymbol{z}^{k+1}) + \boldsymbol{B}^\top\boldsymbol{\lambda}^k + \boldsymbol{B}^\top {\boldsymbol{R}^k}^\top \boldsymbol{R}^k(\boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z}^{k+1} - \boldsymbol{c}), \tag{8-16}$$

which can be simplified in a similar method utilizing the λ-update, giving

$$0 \in \partial g(\boldsymbol{z}^{k+1}) + \boldsymbol{B}^\top\boldsymbol{\lambda}^{k+1}, \tag{8-17}$$

which results in the conclusion that $\boldsymbol{z}^{k+1}$ minimizes

$$g(\boldsymbol{z}) + \boldsymbol{\lambda}^{k+1^\top} \boldsymbol{B}\boldsymbol{z}, \tag{8-18}$$

with the resulting inequality

$$g(\boldsymbol{z}^{k+1}) + \boldsymbol{\lambda}^{k+1^\top} \boldsymbol{B}\boldsymbol{z}^{k+1} \leq g(\boldsymbol{z}^\star) + \boldsymbol{\lambda}^{k+1^\top} \boldsymbol{B}\boldsymbol{z}^\star. \tag{8-19}$$

Since the inequalities given in Equation (8-15) and Equation (8-19) simply state that $\boldsymbol{x}^{k+1}$ and $\boldsymbol{z}^{k+1}$ are minimizing their respective functions, it is possible to combine the two into

$$\begin{aligned} f(\boldsymbol{x}^{k+1}) + g(\boldsymbol{z}^{k+1}) + \left(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\rho}^k \circ \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right)^\top \boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{\lambda}^{k+1^\top} \boldsymbol{B}\boldsymbol{z}^{k+1} \\ \leq f(\boldsymbol{x}^\star) + g(\boldsymbol{z}^\star) + \left(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\rho}^k \circ \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right)^\top \boldsymbol{A}\boldsymbol{x}^\star + \boldsymbol{\lambda}^{k+1^\top} \boldsymbol{B}\boldsymbol{z}^\star, \end{aligned}$$

which can be rewritten and abbreviated, using $p^k = f(\boldsymbol{x}^k) + g(\boldsymbol{z}^k)$, into

$$p^{k+1} - p^\star \leq \boldsymbol{\lambda}^{k+1^\top} \left(\boldsymbol{A}\boldsymbol{x}^\star + \boldsymbol{B}\boldsymbol{z}^\star - (\boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z}^{k+1})\right) - \left(\boldsymbol{\rho}^k \circ \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right)^\top (\boldsymbol{A}\boldsymbol{x}^\star - \boldsymbol{A}\boldsymbol{x}^{k+1}), \tag{8-20}$$

which can be further rewritten when using $\boldsymbol{r}^{k+1} = \boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z}^{k+1} - \boldsymbol{c}$ and $\boldsymbol{A}\boldsymbol{x}^\star + \boldsymbol{B}\boldsymbol{z}^\star = \boldsymbol{c}$ into

$$p^{k+1} - p^\star \leq -\boldsymbol{\lambda}^{k+1^\top} \boldsymbol{r}^{k+1} - \left(\boldsymbol{\rho}^k \circ \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right)^\top \left(-\boldsymbol{r}^{k+1} + \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^\star)\right), \tag{8-21}$$

which proves the inequality given in Lemma 8-3.3.

Proof for Lemma 8-3.4

In order to prove Lemma 8-3.4 we combine the inequalities from Lemma 8-3.3 and Lemma 8-3.2, i.e.

$$0 \leq -(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^\star)^\top \boldsymbol{r}^{k+1} - \left(\boldsymbol{\rho}^k \circ \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right)^\top \left(-\boldsymbol{r}^{k+1} + \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^\star)\right),$$

rearranging and multiplying by two then gives

$$\begin{aligned} 2(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^\star)^\top \boldsymbol{r}^{k+1} - 2\left(\boldsymbol{\rho}^k \circ \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right)^\top \boldsymbol{r}^{k+1} \\ +2\left(\boldsymbol{\rho}^k \circ \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right)^\top \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^\star) \leq 0. \end{aligned} \tag{8-22}$$

Given the size of Equation (8-22) we will separate the equation into three terms which will be first be rewritten separately and later combined back together. The first term of Equation (8-22) is $2(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^\star)^\top \boldsymbol{r}^{k+1}$, which using the λ-update, i.e. $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \boldsymbol{\rho}^k \circ \boldsymbol{r}^{k+1}$ can be written as

$$2(\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^\star)^\top \boldsymbol{r}^{k+1} + \|\boldsymbol{R}^k \boldsymbol{r}^{k+1}\|_2^2 + \|\boldsymbol{R}^k \boldsymbol{r}^{k+1}\|_2^2. \tag{8-23}$$

The reason for previously multiplying the equation with two is to separate the residual norm into two terms; the latter norm is used later in the proof. Substituting $\boldsymbol{r}^{k+1} = (\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k) \oslash \boldsymbol{\rho}$, where $\oslash$ is the *Hadamard division* (or element-wise division) operator, results in

$$2(\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^\star)^\top (\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k) \oslash \boldsymbol{\rho}^k + \|\boldsymbol{R}^{k^{\circ -1}} (\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k)\|_2^2 + \|\boldsymbol{R}^k \boldsymbol{r}^{k+1}\|_2^2,$$

which can be rewritten using the simple algebra trick, i.e. $\boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^{k}=(\boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^{\star})-(\boldsymbol{\lambda}^{k}-\boldsymbol{\lambda}^{\star})$:

$$\begin{aligned}&2(\boldsymbol{\lambda}^{k}-\boldsymbol{\lambda}^{\star})^{\top}(\boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^{\star})\oslash\boldsymbol{\rho}^{k}-2\|\boldsymbol{R}^{k^{\circ-1}}(\boldsymbol{\lambda}^{k}-\boldsymbol{\lambda}^{\star})\|_2^2\\&+\left\|\boldsymbol{R}^{k^{\circ-1}}\left((\boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^{\star})-(\boldsymbol{\lambda}^{k}-\boldsymbol{\lambda}^{\star})\right)\right\|_2^2+\|\boldsymbol{R}^{k}\boldsymbol{r}^{k+1}\|_2^2.\end{aligned}$$

Using $\|u-v\|_2^2=\|u\|_2^2+\|v\|_2^2-2v^{\top}u$, the above equation can be simplified into

$$\|\boldsymbol{R}^{k^{\circ-1}}(\boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^{\star})\|_2^2-\|\boldsymbol{R}^{k^{\circ-1}}(\boldsymbol{\lambda}^{k}-\boldsymbol{\lambda}^{\star})\|_2^2+\|\boldsymbol{R}^{k}\boldsymbol{r}^{k+1}\|_2^2. \tag{8-24}$$

With the first term of Equation (8-22) simplified, the full inequality can be written as

$$\begin{aligned}\|\boldsymbol{R}^{k^{\circ-1}}(\boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^{\star})\|_2^2-\|\boldsymbol{R}^{k^{\circ-1}}(\boldsymbol{\lambda}^{k}-\boldsymbol{\lambda}^{\star})\|_2^2+\|\boldsymbol{R}\boldsymbol{r}^{k+1}\|_2^2-2\left(\boldsymbol{\rho}^{k}\circ\boldsymbol{B}(\boldsymbol{z}^{k+1}-\boldsymbol{z}^{k})\right)^{\top}\boldsymbol{r}^{k+1}\\+2\left(\boldsymbol{\rho}^{k}\circ\boldsymbol{B}(\boldsymbol{z}^{k+1}-\boldsymbol{z}^{k})\right)^{\top}\boldsymbol{B}(\boldsymbol{z}^{k+1}-\boldsymbol{z}^{\star})\leq 0.\end{aligned} \tag{8-25}$$

Proceeding the separated rewriting, the remaining terms of Equation (8-25), including the leftover $\|\boldsymbol{R}^{k}\boldsymbol{r}^{k+1}\|_2^2$ from Equation (8-24), are rewritten. First we reuse the simple algebra trick, i.e. $\boldsymbol{z}^{k+1}-\boldsymbol{z}^{\star}=(\boldsymbol{z}^{k+1}-\boldsymbol{z}^{k})+(\boldsymbol{z}^{k}-\boldsymbol{z}^{\star})$, resulting in the remaining terms being

$$\begin{aligned}\|\boldsymbol{R}^{k}\boldsymbol{r}^{k+1}\|_2^2-2\left(\boldsymbol{\rho}^{k}\circ\boldsymbol{B}(\boldsymbol{z}^{k+1}-\boldsymbol{z}^{k})\right)^{\top}\boldsymbol{r}^{k+1}+2\|\boldsymbol{R}^{k}\boldsymbol{B}(\boldsymbol{z}^{k+1}-\boldsymbol{z}^{k})\|_2^2\\+2\left(\boldsymbol{\rho}^{k}\circ\boldsymbol{B}(\boldsymbol{z}^{k+1}-\boldsymbol{z}^{k})\right)^{\top}\boldsymbol{B}(\boldsymbol{z}^{k}-\boldsymbol{z}^{\star})\leq 0,\end{aligned}$$

where the first three terms can be simplified by combining the norms, again using $\|u\|_2^2+\|v\|_2^2-2v^{\top}u=\|u-v\|_2^2$, into

$$\begin{aligned}\|\boldsymbol{R}^{k}\left(\boldsymbol{r}^{k+1}-\boldsymbol{B}(\boldsymbol{z}^{k+1}-\boldsymbol{z}^{k})\right)\|_2^2+\|\boldsymbol{R}^{k}\boldsymbol{B}(\boldsymbol{z}^{k+1}-\boldsymbol{z}^{k})\|_2^2\\+2\left(\boldsymbol{\rho}^{k}\circ\boldsymbol{B}(\boldsymbol{z}^{k+1}-\boldsymbol{z}^{k})\right)^{\top}\boldsymbol{B}(\boldsymbol{z}^{k}-\boldsymbol{z}^{\star})\leq 0.\end{aligned}$$

Satisfied with the terms within the first norm, we now focus on simplifying the remaining terms, i.e. $\|\boldsymbol{R}^{k}\boldsymbol{B}(\boldsymbol{z}^{k+1}-\boldsymbol{z}^{k})\|_2^2+2\left(\boldsymbol{\rho}^{k}\circ\boldsymbol{B}(\boldsymbol{z}^{k+1}-\boldsymbol{z}^{k})\right)^{\top}\boldsymbol{B}(\boldsymbol{z}^{k}-\boldsymbol{z}^{\star})$. Once again we substitute using $\boldsymbol{z}^{k+1}-\boldsymbol{z}^{k}=(\boldsymbol{z}^{k+1}-\boldsymbol{z}^{\star})-(\boldsymbol{z}^{k}-\boldsymbol{z}^{\star})$ resulting in

$$\begin{aligned}\|...\|_2^2+\|\boldsymbol{R}^{k}\boldsymbol{B}\left((\boldsymbol{z}^{k+1}-\boldsymbol{z}^{\star})-(\boldsymbol{z}^{k}-\boldsymbol{z}^{\star})\right)\|_2^2+2\left(\boldsymbol{\rho}^{k}\circ\boldsymbol{B}(\boldsymbol{z}^{k+1}-\boldsymbol{z}^{\star})\right)^{\top}\boldsymbol{B}(\boldsymbol{z}^{k}-\boldsymbol{z}^{\star})\\-2\|\boldsymbol{R}^{k}\boldsymbol{B}(\boldsymbol{z}^{k}-\boldsymbol{z}^{\star})\|_2^2\leq 0,\end{aligned}$$

which can be reduced into

$$\|...\|_2^2+\|\boldsymbol{R}^{k}\boldsymbol{B}(\boldsymbol{z}^{k+1}-\boldsymbol{z}^{\star})\|_2^2-\|\boldsymbol{R}^{k}\boldsymbol{B}(\boldsymbol{z}^{k}-\boldsymbol{z}^{\star})\|_2^2\leq 0.$$

Recombining all the terms back together results in

$$\begin{aligned}\|\boldsymbol{R}^{k^{\circ-1}}(\boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^{\star})\|_2^2-\|\boldsymbol{R}^{k^{\circ-1}}(\boldsymbol{\lambda}^{k}-\boldsymbol{\lambda}^{\star})\|_2^2+\|\boldsymbol{R}^{k}\left(\boldsymbol{r}^{k+1}-\boldsymbol{B}(\boldsymbol{z}^{k+1}-\boldsymbol{z}^{k})\right)\|_2^2\\+\|\boldsymbol{R}^{k}\boldsymbol{B}(\boldsymbol{z}^{k+1}-\boldsymbol{z}^{\star})\|_2^2-\|\boldsymbol{R}^{k}\boldsymbol{B}(\boldsymbol{z}^{k}-\boldsymbol{z}^{\star})\|_2^2\leq 0.\end{aligned} \tag{8-26}$$

Recall the Lyapunov candidate function, i.e. $V^k = \|\boldsymbol{R}^{k\circ -1}(\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^\star)\|_2^2 + \|\boldsymbol{R}^k\boldsymbol{B}(\boldsymbol{z}^k - \boldsymbol{z}^\star)\|_2^2$, which allows Equation (8-26) to be written as

$$V^k - V^{k+1} \geq \|\boldsymbol{R}^k \left(\boldsymbol{r}^{k+1} - \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right)\|_2^2, \tag{8-27}$$

which can also be expanded into

$$V^{k+1} \leq V^k - \|\boldsymbol{R}^k\boldsymbol{r}^{k+1}\|_2^2 - \|\boldsymbol{R}^k\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\|_2^2 + 2(\boldsymbol{\rho}^k \circ \boldsymbol{r}^{k+1})^\top \left(\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right). \tag{8-28}$$

Equation (8-28) proves Lemma 8-3.4 if $2(\boldsymbol{\rho}^k \circ \boldsymbol{r}^{k+1})^\top \left(\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right) \leq 0$. This can be proven by investigating the definition of $\boldsymbol{z}^{k+1}$, i.e. a minimizer for $g(\boldsymbol{z}) + {\boldsymbol{\lambda}^{k+1}}^\top \boldsymbol{B}\boldsymbol{z}$. Using this definition, it can be stated that

$$g(\boldsymbol{z}^{k+1}) + {\boldsymbol{\lambda}^{k+1}}^\top \boldsymbol{B}\boldsymbol{z}^{\boldsymbol{k+1}} \leq g(\boldsymbol{z}^k) + {\boldsymbol{\lambda}^{k+1}}^\top \boldsymbol{B}\boldsymbol{z}^{\boldsymbol{k}},$$

and vice versa

$$g(\boldsymbol{z}^k) + {\boldsymbol{\lambda}^k}^\top \boldsymbol{B}\boldsymbol{z}^{\boldsymbol{k}} \leq g(\boldsymbol{z}^{k+1}) + {\boldsymbol{\lambda}^k}^\top \boldsymbol{B}\boldsymbol{z}^{\boldsymbol{k+1}}.$$

Combining the two results in

$$(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k)^\top (\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)) \leq 0,$$

which can also be rewritten, using $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \boldsymbol{\rho}^k \circ \boldsymbol{r}^{k+1}$, into

$$(\boldsymbol{\rho}^k \circ \boldsymbol{r}^{k+1})^\top (\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)) \leq 0. \tag{8-29}$$

Equation (8-29) proves that Equation (8-28) can be used to conclude Lemma 8-3.4.

8-3-2 Online Convergence

A general discussion on the online application of ADMM is given in Chapter 7, this subsection builds upon that. In order to prove online convergence we need analyze the change of the system compared with the convergence of OA-ADMM. We distinguish between the OA-ADMM iteration parameter k and the real time time step t, e.g. $\boldsymbol{x}^\star(t)$ is the $\boldsymbol{x}$ at the saddle point for time t, whereas $\boldsymbol{x}^k(t)$ is $\boldsymbol{x}$ at iteration k at time step t.

Whilst the static convergence given in Theorem 8-3.1 proves residual convergence and objective convergence as $k \to \infty$, this result does not directly extend to the online case. The reason being that for online systems, $\boldsymbol{x}^\star(t) = \boldsymbol{x}^\star(t + \delta t)$ cannot be assumed unless $\delta \to 0$, i.e. the control time step is very small/ the control frequency is very high. It is therefore possible that the algorithm is proven to converge towards $\boldsymbol{x}^\star(t)$ from k to $k+1$, whilst not being able to comment on the convergence towards $\boldsymbol{x}^\star(t + \delta t)$. However, each iteration of OA-ADMM is proven to converge towards the optimum for that time step, for that reason it could be argued that OA-ADMM has online convergence. This however does not account for the possibility that the change in optimum is larger than the rate of convergence of OA-ADMM, which can result in each iteration converging towards an optimum, whilst never reaching the optimum for its own time step. We therefore define online convergence to be that OA-ADMM can always converge towards the online optimum as $k, t \to \infty$.

This convergence requirement can be written as

$$\begin{aligned} \boldsymbol{x}^\star(t) &= \boldsymbol{x}^{k+1}(t), \\ \boldsymbol{z}^\star(t) &= \boldsymbol{z}^{k+1}(t), \text{ for } k, t \to \infty. \end{aligned} \tag{8-30}$$

There are several ways to guarantee the requirements given in Equation (8-30); a trivial method which can guarantee this is applying OA-ADMM to problems which are time independent, i.e. $\boldsymbol{x}^\star(t_0) = \boldsymbol{x}^\star(t), \forall t$. Given that proving Equation (8-30) depends on the rate of change of the optimum over time, it is required that the convergence rate of OA-ADMM always dominates the change of optimum as $k, t \to \infty$, i.e.

$$\begin{aligned} \|\boldsymbol{x}^{k_N}(t) - \boldsymbol{x}^\star(t)\|_2 - \|\boldsymbol{x}^{k_0}(t) - \boldsymbol{x}^\star(t)\|_2 > \|\boldsymbol{x}^\star(t+\delta t) - \boldsymbol{x}^\star(t)\|_2, \\ \|\boldsymbol{z}^{k_N}(t) - \boldsymbol{z}^\star(t)\|_2 - \|\boldsymbol{z}^{k_0}(t) - \boldsymbol{z}^\star(t)\|_2 > \|\boldsymbol{z}^\star(t+\delta t) - \boldsymbol{z}^\star(t)\|_2, \forall t, \end{aligned} \tag{8-31}$$

where k_0 (resp. k_N) is the first (resp. last) iteration per time step. There are however no known methods which can guarantee this generally for ADMM based methods; online convergence analysis should be performed on a system specific basis.

In general, if online convergence is not observed, it is advised to either increase the amount of OA-ADMM iterations per control step, or the decrease the time step size δ. This approach is the only approach which is proven to work in general; as mentioned earlier, increasing the amount of iterations, i.e. $k \to \infty$, or decreasing the time step size, i.e. $\delta \to 0$, allows the system to be viewed as a static system.

If more knowledge about the online convergence for a specific system is desired, it is possible to analyze the system and the OA-ADMM algorithm together as a dynamical system. Online convergence can then be compared with stability for the dynamical system, for example using the Lyapunov Direct method. Proving online convergence for OA-ADMM is not too dissimilar from MPC stability proofs, e.g. [120].

Part III

Decentralized Conflict Resolution

This part covers the application of OA-ADMM to decentralized conflict resolution in combination with model predictive control (MPC) in Chapter 9, the method is then compared with other methods in Chapter 10, after which the results are discussed in Chapter 11.

Chapter 9 begins with deriving a decentralized formulation of the main problem given in Chapter 4, followed by the design of a similarity function and an adaptation function, along with some initial numerical results. Chapter 10 then compares the method against other decentralized conflict resolution methods such as AMP-IP and TDCR. Chapter 11 concludes this thesis with a discussion of the proposed method, summarizing its advantages and potential pitfalls, along with an in-depth analysis of simulation results.

Chapter 9

Decentralized Conflict Resolution using OA-ADMM and MPC

Decentralized-optimization-based conflict resolution can be achieved by decomposing the central optimization problem using Online Adaptive Alternating Direction Method of Multipliers (OA-ADMM), with the adaptation function ϕ allowing safe conflict resolution. An example of the type of problem is given in Figure 9-1, where 4 vehicles are depicted at an intersection, along with their respective MPC trajectories.

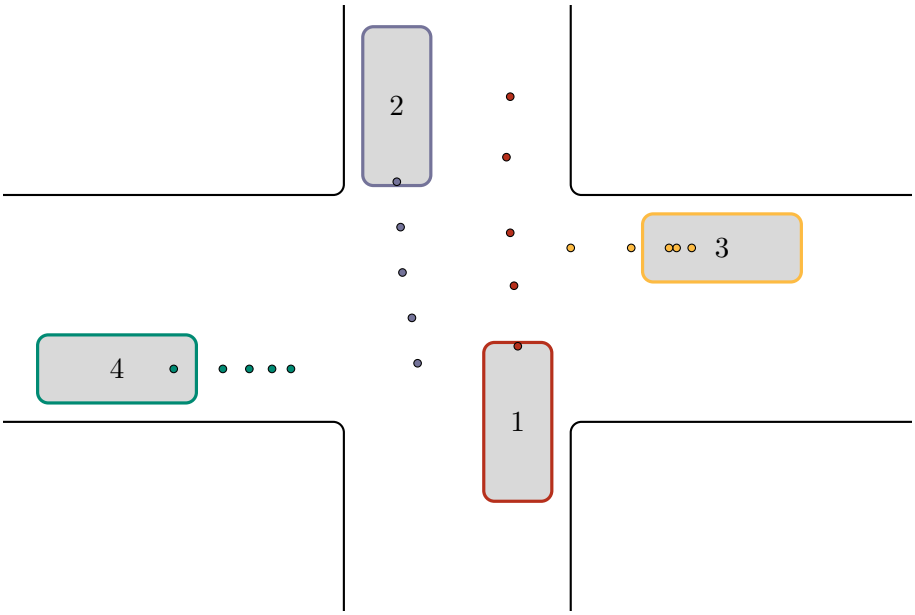


Figure 9-1: Example of conflict resolution to be solved using OA-ADMM and MPC. The colored dots represent the MPC trajectories of each vehicle.

9-1 Methodology

First, the main problem given in Equation (3-1) is rewritten into an MPC based formulation; using an MPC based formulation allows the conflict resolution to act in real-time. The MPC version results in a main equation of the form:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \sum_{i=1}^{N} J_i(\boldsymbol{x}) \\ s.t. \quad & \boldsymbol{x}_i \in \mathcal{X}_i^f, \\ & d_{i,j}(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0, \forall j \in \mathcal{N}_i, \\ & \forall i \in \{1, ..., N\}, \end{aligned} \tag{9-1}$$

where $\boldsymbol{x}_i$ is the concatenated state vector for vehicle i, e.g. in the case of a holonomic vehicle $\boldsymbol{x}_i = [\mathrm{x}, \dot{\mathrm{x}}, \mathrm{y}, \dot{\mathrm{y}}, u^{\mathrm{x}}, u^{\mathrm{y}}]^\top$, where x,y are coordinates and $u^{\mathrm{x}}, u^{\mathrm{y}}$ are the acceleration inputs; $\mathcal{X}_i^f$ is the feasible set for $\boldsymbol{x}_i$ that includes all trajectories that adhere to the system dynamics, input constraint, and environmental collision avoidance constraints; $d_{i,j}(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$ is the distance (or collision avoidance) constraint between the agents, defined as

$$d_{i,j}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \begin{bmatrix} \boldsymbol{x}_i^\top & \boldsymbol{x}_j^\top \end{bmatrix} \begin{bmatrix} I & -I \\ -I & I \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_i \\ \boldsymbol{x}_j \end{bmatrix} - D_{min}^2, \tag{9-2}$$

where D_{min} is the minimum distance between vehicles. Note: D_{min} is used as a global constant for all vehicles for convenience, it is also possible to have a separate D_{min} between any two agents using this method. Furthermore, it is important to note that the complicating constraint does not have to be a nonconvex function like $d_{i,j}(x_i, x_j)$, e.g. an equality constraint like $g(\cdot)$ in Section 7-2-2 would work equally well, in which case it would be formation control instead of collision avoidance.

In order to decompose the problem given in Equation (9-1), the problem has to be rewritten into a form that is compatible with the general OA-ADMM form given in Equation (8-1). Similarly to [23], this is done by introducing a slack variable $\boldsymbol{z}$ and an equality constraint $\boldsymbol{x} = \boldsymbol{z}$, i.e.

$$\begin{aligned} \min_{\boldsymbol{x}, \boldsymbol{z_{ij}}} \quad & \sum_{i=1}^{N} J_i(x) \\ s.t. \quad & \boldsymbol{x}_i \in \mathcal{X}_i^f, \\ & d_{i,j}(\boldsymbol{z}_i, \boldsymbol{z}_j) \geq 0, \forall j \in \mathcal{N}_i, \\ & \boldsymbol{x}_i = \boldsymbol{z}_{ii}, \boldsymbol{x}_j = \boldsymbol{z}_{ij}, \forall j \in \mathcal{N}_i \\ & \forall i \in \{1, ..., N\}. \end{aligned} \tag{9-3}$$

The OA-ADMM augmented Lagrangian $\mathcal{L}_{\boldsymbol{\rho}}$ for this form is then

$$\mathcal{L}_{\boldsymbol{\rho}} = \sum_{i=1}^{N} \Big(J_i(\boldsymbol{x}_i) + \boldsymbol{\lambda}_{ii}^\top (\boldsymbol{x}_i - \boldsymbol{z}_i) + \|\boldsymbol{R}_{ii}(\boldsymbol{x}_i - \boldsymbol{z}_{ii})\|_2^2 + \sum_{j \in \mathcal{N}_i} \Big(\boldsymbol{\lambda}_{ij}^\top (\boldsymbol{x}_j - \boldsymbol{z}_{ij}) + \|\boldsymbol{R}_{ij}(\boldsymbol{x}_j - \boldsymbol{z}_{ij})\|_2^2 \Big) \Big), \tag{9-4}$$

where $\boldsymbol{\lambda}_{ii}$ (resp. $\boldsymbol{\lambda}_{ij}$) is the Lagrange multiplier in vector form for agent i (resp. agent i to agent j), and $\boldsymbol{R}_{ii}$ (resp. $\boldsymbol{R}_{ij}$) is the diagonal matrix form of the penalty parameter $\boldsymbol{\rho}_{ii}$

(resp. $\boldsymbol{\rho}_{ij}$) with the diagonal of $\boldsymbol{R}$ being $\rho^{\circ\frac{1}{2}}$, where $\circ$ is the *Hadamard power* (or element-wise power).

Similarly to the online ADMM strategy from [23], described in Section 7-2 we separate the overall optimization problem into smaller steps to enable the distribution of computational load. The first step then being the trajectory optimizations step (or x-update):

$$\begin{aligned} \boldsymbol{x}_i^{k+1} := \arg\min_{\boldsymbol{x}_i} \quad & \mathcal{L}_{\boldsymbol{\rho},x,i}(\boldsymbol{x}_i, \boldsymbol{z}_{JI}^k, \boldsymbol{\lambda}_{JI}^k, \boldsymbol{\rho}_{JI}^k) \\ s.t. \quad & \boldsymbol{x}_i \in \mathcal{X}_i^f, \end{aligned} \tag{9-5}$$

where for writing convenience we combine the variables into $\boldsymbol{z}_{JI}, \boldsymbol{\lambda}_{JI}, \boldsymbol{\rho}_{JI}$; $\boldsymbol{z}_{JI}$ includes $\boldsymbol{z}_{ii}$ and $\boldsymbol{z}_{ji}, \forall j \in \mathcal{N}_i$, $\boldsymbol{\lambda}_{JI}$ is the combination of $\boldsymbol{\lambda}_{ii}$ and $\boldsymbol{\lambda}_{ji}, \forall j \in \mathcal{N}_i$, and $\boldsymbol{\rho}_{JI}$ consists out of $\boldsymbol{\rho}_{ii}$ and $\boldsymbol{\rho}_{ji}, \forall j \in \mathcal{N}_i$. Given that the x-update only adjusts $\boldsymbol{x}_i$, a lighter version of the full augmented Lagrangian of Equation (9-4) is used, i.e.

$$\mathcal{L}_{\boldsymbol{\rho},x,i} = J_i(\boldsymbol{x}_i) + {\boldsymbol{\lambda}_{ii}^k}^\top(\boldsymbol{x}_i - \boldsymbol{z}_{ii}^k) + \|\boldsymbol{R}_{ii}^k(\boldsymbol{x}_i - \boldsymbol{z}_{ii}^k)\|_2^2 + \sum_{j\in\mathcal{N}_i}\left({\boldsymbol{\lambda}_{ji}^k}^\top(\boldsymbol{x}_i - \boldsymbol{z}_{ji}^k) + \|\boldsymbol{R}_{ji}^k(\boldsymbol{x}_i - \boldsymbol{z}_{ji}^k)\|_2^2\right), \tag{9-6}$$

where $\boldsymbol{\lambda}_{ji}^k$ (resp. $\boldsymbol{R}_{ji}^k$) is the Lagrange multiplier (resp. penalty matrix) for agent j w.r.t agent i. The trajectory optimization step is fully parallelizable given that all the values in the Augmented Lagrangian of Equation (9-6) are either known or independent of other agents. In order to proceed, the resulting $\boldsymbol{x}_i^{k+1}$ has to be communicated with all nearby agents. After sending $\boldsymbol{x}_i^{k+1}$ to, and receiving $\boldsymbol{x}_j^{k+1}$ from, all $j \in \mathcal{N}_i$, the copy optimization step (or z-update) can be performed. The z-update can be seen as the collision avoidance update given that this update includes the $d_{i,j}(\cdot) \geq 0$ constraint, with its definition being

$$\begin{aligned} \boldsymbol{z}_{IJ}^{k+1} := \arg\min_{\boldsymbol{z}_{IJ}} \quad & \mathcal{L}_{\boldsymbol{\rho},z,i}(\boldsymbol{x}_I^{k+1}, \boldsymbol{z}_{IJ}^k, \boldsymbol{\lambda}_{IJ}^k, \boldsymbol{\rho}_{IJ}^k) \\ s.t. \quad & d_{i,j}(\boldsymbol{z}_i^{k+1}, \boldsymbol{z}_j^{k+1}) \geq 0, \ \forall j \in \mathcal{N}_i, \end{aligned} \tag{9-7}$$

where $\boldsymbol{z}_{IJ}$ (resp. $\boldsymbol{x}_I$) contains both $\boldsymbol{z}_{ii}$ and $\boldsymbol{z}_{ij}, \forall j \in \mathcal{N}_i$ (resp. $\boldsymbol{x}_i$ and $\boldsymbol{x}_j, \forall j \in \mathcal{N}_i$). The reduced augmented Lagrangian for the z-update ($\mathcal{L}_{\boldsymbol{\rho},z,i}$) is defined as

$$\mathcal{L}_{\boldsymbol{\rho},z,i} = {\boldsymbol{\lambda}_{ii}^k}^\top(\boldsymbol{x}_i^{k+1} - \boldsymbol{z}_i) + \|\boldsymbol{R}_{ii}^k(\boldsymbol{x}_i^{k+1} - \boldsymbol{z}_{ii})\|_2^2 + \sum_{j\in\mathcal{N}_i}\left({\boldsymbol{\lambda}_{ij}^k}^\top(\boldsymbol{x}_j^{k+1} - \boldsymbol{z}_{ij}) + \|\boldsymbol{R}_{ij}^k(\boldsymbol{x}_j^{k+1} - \boldsymbol{z}_{ij})\|_2^2\right). \tag{9-8}$$

Following the x and z updates, the λ-update is performed.

$$\begin{aligned} \boldsymbol{\lambda}_{ii}^{k+1} &:= \mu_i \cdot \boldsymbol{\lambda}_{ii}^k + \boldsymbol{\rho}_{ii}^k \circ (\boldsymbol{x}_i^{k+1} - \boldsymbol{z}_{ii}^{k+1}), \\ \boldsymbol{\lambda}_{ij}^{k+1} &:= \mu_{ij} \cdot \boldsymbol{\lambda}_{ij}^k + \boldsymbol{\rho}_{ij}^k \circ (\boldsymbol{x}_j^{k+1} - \boldsymbol{z}_{ij}^{k+1}), \forall j \in \mathcal{N}_i \end{aligned} \tag{9-9}$$

where $\boldsymbol{\rho}_{ii}^{k+1}, \boldsymbol{\rho}_{ij}^{k+1}, \boldsymbol{z}_{ii}^{k+1}$, and $\boldsymbol{z}_{ij}^{k+1}$ are all available locally, and μ is a forgetting factor which represents the similarity between the system of the current time step and the previous time step. When multiple iterations of OA-ADMM are performed per real-time time step, the value of μ can be assumed to be 1 for all iterations in the same real-time time step.

The final step of a single OA-ADMM iteration involves updating the penalty vector ρ, i.e.

$$\begin{aligned} \boldsymbol{\rho}_{ij}^{k+1} &:= \phi_{ij}(\boldsymbol{x}_i^{k+1}, \boldsymbol{x}_j^{k+1}), \forall j \in \mathcal{N}_i \\ \boldsymbol{\rho}_{ii}^{k+1} &:= \phi_{ii}(\boldsymbol{x}_i^{k+1}), \end{aligned} \tag{9-10}$$

where $\phi(\cdot)$ is a function dependent on the states, with the requirement that the resulting $\boldsymbol{R}$ is always a positive definite diagonal matrix. The exact formulation of $\phi(\cdot)$ can be varied dependent on the desired behavior. The updated values of $\boldsymbol{z}^{k+1}$, $\boldsymbol{\rho}^{k+1}$, and $\boldsymbol{\lambda}^{k+1}$ are then communicated to complete one OA-ADMM iteration.

The primal residual for this method for an agent i is simply the difference between its planned trajectory and the desired copies, i.e.

$$\boldsymbol{r}_i^{k+1} = \boldsymbol{x}_i^{k+1} - \boldsymbol{z}_i^{k+1} i + \sum_{j \in \mathcal{N}_i} (\boldsymbol{x}_i^{k+1} - \boldsymbol{z}_{ji}^{k+1}), \tag{9-11}$$

with the dual residual being

$$\boldsymbol{s}_i^{k+1} = \boldsymbol{\rho}_{JI}^{k} \circ (\boldsymbol{z}_{JI}^{k} - \boldsymbol{z}_{JI}^{k+1}), \tag{9-12}$$

where $\boldsymbol{\rho}_{JI}$ contains $\boldsymbol{\rho}_{ji}, \forall j \in \mathcal{N}_i$.

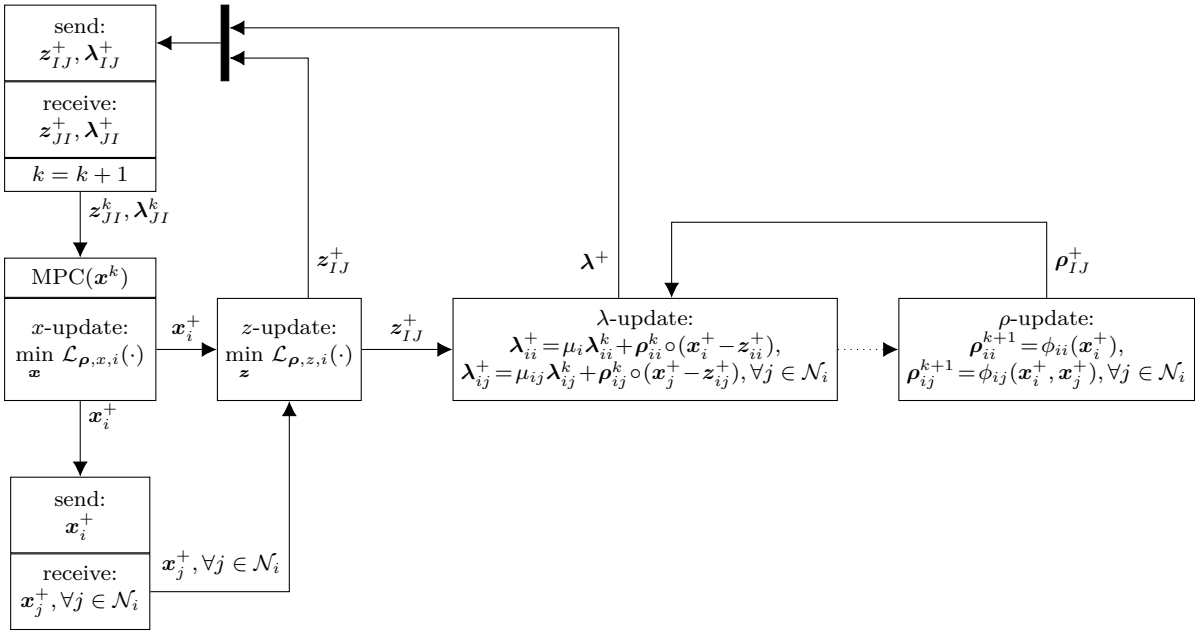


Figure 9-2: Diagram of the OA-ADMM and MPC based conflict resolution algorithm for an agent i. The superscript $+$ is used as a short version of $k+1$, i.e. $\boldsymbol{x}^+ := \boldsymbol{x}^{k+1}$; dotted arrow indicates code execution order; necessary variables are automatically perpetuated along the arrows, e.g. the λ-update receives $\boldsymbol{x}_j^+$ via the z-update block.

9-1-1 Designing the similarity function $\mu(\cdot)$

Despite the importance of the functions $\mu(\cdot)$ and ϕ, they have not yet been thoroughly investigated yet. First we begin with μ, which is supposed to represent the similarity between two real-time steps. When $\mu(\cdot) = 1$, it is implied that the change from t to $t + \delta t$ has no effect on the previous OA-ADMM iterations, i.e. $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \boldsymbol{\rho}^k \circ \boldsymbol{r}^{k+1}$. Conversely a $\mu(\cdot) = 0$ implies that there is no useful relation between the previous time step t and the current time step $t + \delta t$, i.e. $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\rho}^k \circ \boldsymbol{r}^{k+1}$. The difficulty is however designing a function μ that, using the information available, results in effective online performance. If the system is fully

known, it might be possible to analytically find the optimal $\mu^\star$, this is however a time intensive procedure and very system dependent. Instead we attempt an intuitive approach to find a μ which approximates the behavior of $\mu^\star$.

We know that if $\mu(\cdot)^\star = 1$ the following should hold:

$$\begin{aligned} \boldsymbol{x}^\star(t) &= \boldsymbol{x}^\star(t+\delta t), \\ \boldsymbol{z}^\star(t) &= \boldsymbol{z}^\star(t+\delta t), \\ \boldsymbol{\lambda}^\star(t) &= \boldsymbol{\lambda}^\star(t+\delta t), \\ \boldsymbol{\rho}^\star(t) &= \boldsymbol{\rho}^\star(t+\delta t), \end{aligned} \tag{9-13}$$

simply said, the optimum should not change from t to $t+\delta t$ if $\mu = 1$. This cannot be used directly however, as the optimum is what OA-ADMM is attempting to find, and is not available. We do however know that, if OA-ADMM has reached the optimum, then the following holds

$$\begin{aligned} \boldsymbol{x}^k(t) &= \boldsymbol{x}^{k+1}(t), \\ \boldsymbol{z}^k(t) &= \boldsymbol{z}^{k+1}(t), \\ \boldsymbol{\lambda}^k(t) &= \boldsymbol{\lambda}^{k+1}(t), \\ \boldsymbol{\rho}^k(t) &= \boldsymbol{\rho}^{k+1}(t). \end{aligned} \tag{9-14}$$

If, on top of that, we add the requirements from Equation (9-13) we get that, if OA-ADMM has converged and $\mu^\star = 1$, then

$$\begin{aligned} \boldsymbol{x}^k(t) &= \boldsymbol{x}^{k+1}(t+\delta t), \\ \boldsymbol{z}^k(t) &= \boldsymbol{z}^{k+1}(t+\delta t), \\ \boldsymbol{\lambda}^k(t) &= \boldsymbol{\lambda}^{k+1}(t+\delta t), \\ \boldsymbol{\rho}^k(t) &= \boldsymbol{\rho}^{k+1}(t+\delta t). \end{aligned} \tag{9-15}$$

Given that the values in Equation (9-15) are actually known in run-time, we can utilize this to construct a μ which approximates $\mu^\star$ in the optimum. For example, the following formula satisfies Equation (9-15):

$$\begin{aligned} \mu(\cdot) = w_x \left(1 - \frac{\|\boldsymbol{x}^{k+1}(t+\delta t) - \boldsymbol{x}^k(t)\|_2}{\|\boldsymbol{x}^k(t)\|_2}\right) + w_z \left(1 - \frac{\|\boldsymbol{z}^{k+1}(t+\delta t) - \boldsymbol{z}^k(t)\|_2}{\|\boldsymbol{z}^k(t)\|_2}\right) \\ + \; w_\lambda \left(1 - \frac{\|\boldsymbol{\lambda}^{k+1}(t+\delta t) - \boldsymbol{\lambda}^k(t)\|_2}{\|\boldsymbol{\lambda}^k(t)\|_2}\right) + w_\rho \left(1 - \frac{\|\boldsymbol{\rho}^{k+1}(t+\delta t) - \boldsymbol{\rho}^k(t)\|_2}{\|\boldsymbol{\rho}^k(t)\|_2}\right), \end{aligned} \tag{9-16}$$

where $w_x + w_z + w_\lambda + w_\rho = 1$. When OA-ADMM is not performed until convergence, the requirement from Equation (9-14) is no longer guaranteed. The similarity function from Equation (9-16) underestimates the ideal $\mu^\star$. To address this, a comparison should be made between a previous change and the current change. First we define the following short-hand notations:

$$\begin{aligned} \Delta^k x &= \|\boldsymbol{x}^{k+1}(t) - \boldsymbol{x}^k(t)\|_2, \text{where } k > 0 \\ \Delta^t x &= \|\boldsymbol{x}^{k_0+1}(t+\delta t) - \boldsymbol{x}^{k_N+1}(t)\|_2, \text{where } \boldsymbol{x}^{k_N}(t) = \boldsymbol{x}^{k_0}(t+\delta t), \end{aligned}$$

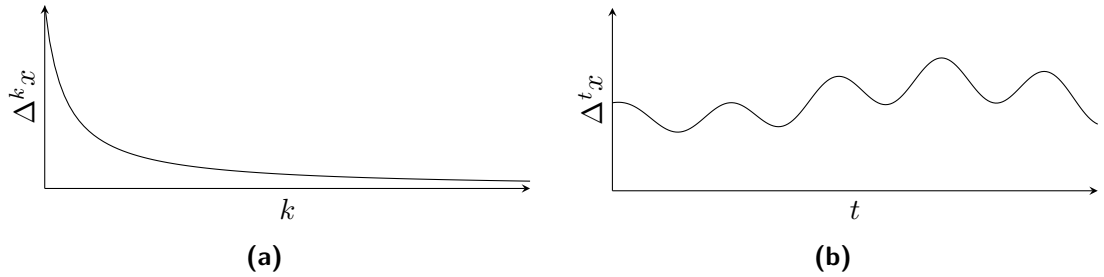


Figure 9-3: Potential behavior of Δ^t and Δ^k

where k_0 is the initial iteration used when initiating the OA-ADMM loop, and k_N is the last iteration of the OA-ADMM loop with $k = N^k$. Similarly, this notation applies to $\boldsymbol{z}, \boldsymbol{\lambda}$, and $\boldsymbol{\rho}$. Given that $\Delta^k x = 0$ when OA-ADMM has converged it can be used as a measure of convergence, similarly $\Delta^t x = 0$ when $\mu^\star = 1$. We also know that $\Delta^k x$ is decreasing as $k \to \infty$ given the static convergence properties given in Section 8-3-1. This behavior is also depicted in Figure 9-3. By comparing the change between two OA-ADMM iterations at one time step, and two OA-ADMM iterations at different time steps we can approximate μ, i.e.

$$\mu(\cdot) = w_x \frac{\left|\Delta^k x - \Delta^t x\right|}{\left|\Delta^k x\right|} + w_z \frac{\left|\Delta^k z - \Delta^t z\right|}{\left|\Delta^k z\right|} + w_\lambda \frac{\left|\Delta^k \lambda - \Delta^t \lambda\right|}{\left|\Delta^k \lambda\right|} + w_\rho \frac{\left|\Delta^k \rho - \Delta^t \rho\right|}{\left|\Delta^k \rho\right|}. \tag{9-17}$$

This however has the requirement that at least two OA-ADMM iterations are performed per time step, and additionally requires the recalculation of $\boldsymbol{x}^k$ each time step.

In the case that only one OA-ADMM iteration can be performed per control step, i.e. $N_k = 1$, it is not possible to get $\Delta^k x$ as we do not get both $\boldsymbol{x}^k(t)$ and $\boldsymbol{x}^{k+1}(t)$ with $k > 0$. Additionally, if we can only perform one OA-ADMM iteration per time step, we also cannot obtain $\Delta^t x$ as it requires an additional iteration. Instead we can only directly obtain $\boldsymbol{x}^{k+1}(t)$ for a certain iteration/control step t. Comparing the change between two control steps now results in a combination of $\Delta^t x$ and $\Delta^k x$, as the change in variables will be dependent on both the convergence and time. We will denote this new change as

$$\Delta^{t,k} x = \|\boldsymbol{x}^{k+1}(t + \delta t) - \boldsymbol{x}^{k+1}(t)\|, \text{where } \boldsymbol{x}^k(t + \delta t) = \boldsymbol{x}^{k+1}(t).$$

In the case that $\mu^\star(\cdot) = 1$, and the system has fully converged, $\Delta^{t,k} x = 0$ should hold. When we combine that with the knowledge that $\Delta^k x$ should be decreasing when the system is converging, along with an assumption that $\Delta^t x$ is constant, we can approximate $\mu(\cdot)$ by analyzing the value of $\Delta^{t,k} x$ over the time steps. If $\Delta^t x$ is constant, then the change in $\Delta^{t,k} x$ should be $\Delta^k x$, i.e.

$$\hat{\Delta}^k x = \Delta^{t+\delta t,k} x - \Delta^{t,k} x.$$

If we also assume that $\Delta^{t,k}$ is a linear combination of Δ^t and Δ^k, we can also approximate Δ^t as

$$\hat{\Delta}^t x = \Delta^{t,k} x - \hat{\Delta}^k x,$$

resulting in the approximated version of Equation (9-17) being

$$\mu(\cdot) = w_x \frac{\Delta^{t,k} x}{\left|\Delta^{t+\delta t,k} x - \Delta^{t,k} x\right|} + w_z \frac{\Delta^{t,k} z}{\left|\Delta^{t+\delta t,k} z - \Delta^{t,k} z\right|} + w_\lambda \frac{\Delta^{t,k} \lambda}{\left|\Delta^{t+\delta t,k} \lambda - \Delta^{t,k} \lambda\right|} + w_\rho \frac{\Delta^{t,k} \rho}{\left|\Delta^{t+\delta t,k} \rho - \Delta^{t,k} \rho\right|}. \tag{9-18}$$

The formula for μ given in Equation (9-18) is only a simple and somewhat crude approximation of $\mu^\star$. It is likely that in an actual case the $\Delta^{t,k}$ is not simply a linear combination of Δ^t

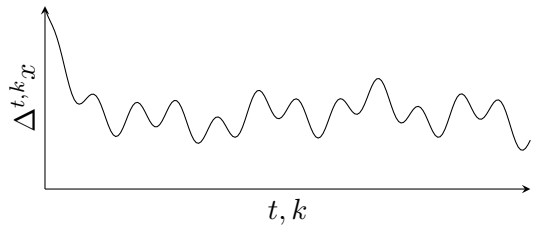


Figure 9-4: Potential behavior of $\Delta^{t,k}$ for the same system as Figure 9-3

and Δ^k, and has a behavior similar to that of Figure 9-4 instead, assuming that it is the same system as depicted in Figure 9-3. In this case this approach to designing μ given in Equation (9-18) cannot be used, and instead it is recommended that the user attempts to identify Δ^t and Δ^k separately from $\Delta^{t,k}$. For example, one can perform the OA-ADMM algorithm until convergence for the system with many initial values and compare it with the online case to obtain more information on Δ^t and Δ^k. If the behavior of Δ^k and/or Δ^t appears to be consistent among all results, it is for example possible to use a Kalman filter to observe the values of Δ^t and/or Δ^k and apply them in Equation (9-17).

Another possible approach to constructing a $\mu(\cdot)$ is to consider the cases where $\mu < 1$ is beneficial for the online performance. Given that $\mu < 1$ affects how much of $\boldsymbol{\lambda}^k$ is utilized in the λ-update, lowering μ is only beneficial when $\boldsymbol{\lambda}^\star < \boldsymbol{\lambda}^k$, i.e. the optimal $\boldsymbol{\lambda}^\star$ for this time step is lower than the previous updated $\boldsymbol{\lambda}$. This is however somewhat dealt with in the original update, given that

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \boldsymbol{\rho}^k \circ \boldsymbol{r}^{k+1},$$

resulting in the $\boldsymbol{\lambda}^{k+1}$ converging to $\boldsymbol{\lambda}^\star$ due to the residual $\boldsymbol{r}^{k+1}$. This is however insufficient in adaptive-ADMM, given that the value of $\boldsymbol{\rho}^k$ is no longer static. If the penalty parameter decreases significantly from k to $k+1$, it is likely that an overshoot in $\boldsymbol{\lambda}^k$ will not be corrected sufficiently with the conventional λ-update. The ideal μ would account for this, i.e.

$$\mu(\cdot) = ({\boldsymbol{\lambda}^k}^\top)^{-1}(\boldsymbol{\lambda}^\star - \boldsymbol{\rho}^k \boldsymbol{r}^{k+1}), \tag{9-19}$$

resulting in $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^\star$. Given that μ has relatively little hard requirements many approaches can be used. It is however important that μ is not consistently underestimating $\mu^\star$ as this can make online convergence hard or impossible.

9-1-2 Designing the adaptation function $\phi(\cdot)$

Two examples of adaptive penalty parameters have already been given in Sections 6-2 and 6-3 with their corresponding design process. Whilst both the residual balancing, and the spectral penalty parameter method are designed to accelerate convergence in general for all types of ADMM problems, it can be desired to design an adaption method which can be designed to fit certain needs. Furthermore, instead of designing rule based adaptation schemes as in [28, 31], we aim to design an adaptation function ϕ instead.

The requirements for $\phi(\cdot)$ for convergence is given in Section 8-3-1, where it is stated that if $\boldsymbol{\rho}$ converges to a certain $\rho^\star$, then OA-ADMM converges for the static case. This can also be written as

$$\dot{\phi}(\cdot, t) = 0, \text{for } t \to \infty. \tag{9-20}$$

In addition of the basic requirement on ϕ, we also have to take into account the purpose of ϕ in the online case. Given that it is difficult for online convergence to be proven for the decentralized conflict resolution/motion planning problem, we will utilize the ability of ϕ to adjust the weights of each constraint individually.

Since we are applying OA-ADMM and MPC to a motion planning problem involving autonomous vehicles, it is desired that ϕ is used to improve the online collision avoidance behavior. As stated in Section 6-1, the value of the penalty parameter $\boldsymbol{\rho}$ has a large influence on the rate of convergence of ADMM and thereby also OA-ADMM; additionally, $\boldsymbol{\rho}$ affects the importance of the primal and dual residuals during optimization, with large a $\boldsymbol{\rho}$ prioritizing the primal residual $\boldsymbol{r}^k$, and a smaller $\boldsymbol{\rho}$ prioritizing the dual residual $\boldsymbol{s}^k$.

Given that the primal residual for Equation (9-3) is $\boldsymbol{x} - \boldsymbol{z}$, increasing $\boldsymbol{\rho}$ effectively increases the penalty for the actual trajectory deviating from the copies. However since Equation (9-3) is a multi-agent problem, there are also agent specific values of $\boldsymbol{\rho}$. In essence, each agent i has three relevant types of $\boldsymbol{\rho}$, namely $\boldsymbol{\rho}_{ii}$, $\boldsymbol{\rho}_{ij}$ and $\boldsymbol{\rho}_{ji}$. The first type, $\boldsymbol{\rho}_{ii}$, directly affects $\boldsymbol{\lambda}_{ii}^{k+1}$ by scaling part of the primal residual $(\boldsymbol{x}_{ii}^{k+1} - \boldsymbol{z}_{ii}^{k+1})$, it is also present in the augmented Lagrangian, acting as a weight on the residual inside the squared L2 norm. The value of $\boldsymbol{\rho}_{ii}$ can therefore be summarized as the weight for the cost of the deviation between $\boldsymbol{x}_{ii}^{k+1}$ (resp. $\boldsymbol{z}_{ii}^{k+1}$) and $\boldsymbol{z}_{ii}^{k}$ (resp. $\boldsymbol{x}_{ii}^{k+1}$). The second type of $\boldsymbol{\rho}$ is $\boldsymbol{\rho}_{ij}$, which is present in the calculation of $\boldsymbol{\lambda}_{ij}^{k+1}$ as a weight for the residual $\boldsymbol{x}_{k}^{k+1} - \boldsymbol{z}_{ij}^{k+1}$ and in the augmented Lagrangian $\mathcal{L}_{\boldsymbol{\rho},z,i}$ in which it scales the same residual inside the norm. As a result, the value of $\boldsymbol{\rho}_{ij}$ directly affects the z-update for agent i, with larger values allowing less deviation of $\boldsymbol{z}_{ij}^{k+1}$ from $\boldsymbol{x}_{j}^{k+1}$. The final $\boldsymbol{\rho}$ is $\boldsymbol{\rho}_{ji}$, which is in essence the reverse of $\boldsymbol{\rho}_{ij}$, i.e. $\boldsymbol{\rho}_{1,2} = \boldsymbol{\rho}_{2,1}$. From the perspective of agent i, $\boldsymbol{\rho}_{ji}$ acts purely in the x-update, penalizing deviation of $\boldsymbol{x}_{i}^{k+1}$ from $\boldsymbol{z}_{ji}^{k+1}$.

To summarize, for agent i, $\boldsymbol{\rho}_i$ acts on the residual $\boldsymbol{x}_i - \boldsymbol{z}_i$, $\boldsymbol{\rho}_{ij}$ acts on the residual $\boldsymbol{x}_{j}^{k+1} - \boldsymbol{z}_{ij}^{k+1}$; and $\boldsymbol{\rho}_{ji}$ acts on the residual $\boldsymbol{x}_{i}^{k+1} - \boldsymbol{z}_{ji}^{k+1}$. The difference between the types of penalty vectors is what allows OA-ADMM to achieve conflict resolution through designing ϕ. For example, a priority based solution can be implemented through ϕ as it is possible to adjust the amount each agent values a certain residual.

A simple example of an adaptation function that can enhance online robustness adapts the penalty parameter based on the physical states, i.e.,

$$\phi(\cdot, k)_{ij} = w_i \left(\frac{D_{min}}{dist(\boldsymbol{x}_i^k, \boldsymbol{x}_j^k)} \right)^a, \tag{9-21}$$

where w_i is a weight that can modify the importance of agent i, $dist(\boldsymbol{x}_i, \boldsymbol{x}_j)$ returns the distance between agents i and j, and a is a variable that determines the shape of ϕ. The ϕ given in Equation (9-21) therefore adjusts the value of ρ_{ij} when the online MPC is planning unsafe trajectories, increasing the primal feasibility whilst sacrificing individual optimality. The advantage of this method is that this distance based approach is very simple to design, yet achieves results similar to methods using more advanced techniques. Due to the role of ρ,

the function given in Equation (9-21) can in a way interpreted as a control barrier function. A recent example for the conventional control barrier function applied to MPC can be found in [121].

Given that Equation (9-21) only changes the penalty vector for an agent w.r.t. another agent, we also need to define the penalty vector for its own copies. This can be simply done by taking the average of its penalty vectors w.r.t. its neighboring agents.

$$\phi(\cdot,k)_{ii} = w_i \frac{1}{N_i} \sum_{j \in \mathcal{N}_i} \left(\frac{D_{min}}{dist(\boldsymbol{x}_i^k, \boldsymbol{x}_j^k)} \right)^a, \tag{9-22}$$

where N_i is the amount of vehicles in $\mathcal{N}_i$.

9-2 Numerical Results

In this section we compare our proposed OA-ADMM based method and a conventional ADMM based method for a simple conflict resolution environment. Since the intention of OA-ADMM is to provide a framework with increased robustness, ease of use, and safety compared with conventional ADMM, we will attempt to provide an accurate representation of these metrics through simulating a simple conflict resolution problem in MATLAB.

9-2-1 Simulation Setup

The simulations are carried out in MATLAB 2019b, with OA-ADMM implemented as proposed in this section, compared with an Online ADMM approach which does not adapt itself during the simulation. The O-ADMM version simply utilizes a static penalty parameter ρ and a static forgetting factor μ. The pseudocode for OA-ADMM MPC is given in Algorithm 9.1, the algorithm for the O-ADMM implementation can be achieved by removing Line 9 and setting a static μ. The simulation is performed at a frequency of 10 Hz, with the MPC having a finite horizon of 10 steps, each of which 0.175 s apart[23]; only one (OA-)ADMM step is performed per iteration, i.e. $N_k = 1$, and communication is assumed to be perfect and instant. The model is simulated in MATLAB, using CVX in combination with MOSEK to solve the optimization steps.

9-2-1-1 Environment

In order to compare the effectiveness of OA-ADMM we simulate a simple conflict resolution problem with 4 agents, as depicted in Figure 9-5. Each agent is identical in terms of model, constraints, and algorithm, with the only difference that agents driving on the horizontal road have a lower priority than agents in the vertical road. A priority order is implemented by adjusting the weights to allow a simple and consistent conflict resolution method applicable to both conventional ADMM and OA-ADMM. The weights are implemented in the adaptation functions given in Equations (9-21) and (9-22), whilst conventional ADMM uses this as the regular ρ. The difference in priority is achieved by multiplying the weights for the vehicles traversing the horizontal road.

[23] This time step was found to perform well in simulations, however more conventional time steps of 0.1 s or 0.2 s also work well.

Algorithm 9.1: OA-ADMM MPC Pseudocode for a single agent i

initialize $\boldsymbol{x}^{k_0}, \boldsymbol{z}^{k_0}, \boldsymbol{\lambda}^{k_0}, \boldsymbol{\rho}^{k_0}$
while $t \leq T_{final}$ and goal() $\neq$ TRUE **do**
  while $\text{mod}(k, N_k) \neq 0$ and $\|\boldsymbol{r}^k\|_2 > \epsilon_p$ and $\|\boldsymbol{s}^k\|_2 > \epsilon_s$ **do**
    $\boldsymbol{x}_i^{k+1} = \arg\min_{\boldsymbol{x}} \mathcal{L}_{\boldsymbol{\rho},x,i}(\boldsymbol{x}, \boldsymbol{z}_{JI})$, i.e. Equation (9-5)
    Broadcast: $\boldsymbol{x}_i^{k+1}$, Receive: $\boldsymbol{x}_j^{k+1} \forall j \in \mathcal{N}_i$
    $\boldsymbol{z}_{IJ}^{k+1} = \arg\min_{\boldsymbol{z}} \mathcal{L}_{\boldsymbol{\rho},z,i}(\boldsymbol{z}, \boldsymbol{x}_I)$, i.e. Equation (9-7)
    $\boldsymbol{\lambda}_{IJ}^{k+1} = ...$, see Equation (9-9)
    Send: $\boldsymbol{z}_{ij}^{k+1}, \boldsymbol{\lambda}_{ij}^{k+1}$ To: $\forall j \in \mathcal{N}_i$, Receive: $\boldsymbol{z}_{ji}^{k+1}, \boldsymbol{\lambda}_{ji}^{k+1}$, From: $\forall j \in \mathcal{N}_i$
    $\boldsymbol{\rho}_{IJ}^{k+1} = \phi(\cdot)$, i.e. Equation (9-10)
    $k = k + 1$
  end
  Apply control(x^k)
  $t = t + \delta t$
  Get current system state Update OA-ADMM MPC problem set $\boldsymbol{x}^0 = \boldsymbol{x}^k, \boldsymbol{z}^0 = \boldsymbol{z}^k, \boldsymbol{\lambda}^0 = \mu(\cdot)\boldsymbol{\lambda}^k, \boldsymbol{\rho}^0 = \boldsymbol{\rho}^k$
  $k = 0$
end

9-2-1-2 Parameters

Given that the effectiveness of the algorithm is dependent on the values of several parameters, a fair comparison between OA-ADMM and O-ADMM requires us to test for multiple values. The most important parameter is the base value of ρ, which is used as a weight, w, for OA-ADMM, and as the actual static penalty parameter ρ in conventional ADMM. The range of ρ that we test for is as follows: $\rho_{base} = \{0.25, 0.5, ..., 5\}$ Additionally, in order to improve conflict resolution, and to avoid simple deadlocks, the D_{min} from Equation (9-2) can be inflated compared with the actual minimum distance, this is implemented through a multiplier D_{mult}. We have simulated both OA-ADMM and O-ADMM using the following multipliers: $\{1, 1.1, 1.2, ..., 2\}$. The actual minimum distance used to measure constraint violation is D_{min} with a value of 2.75 m center to center.

The MPC controller is implemented with a finite horizon of 8 steps, with a step size of 0.2 seconds. Additionally the MPC controller attempts to maintain a reference velocity of 6 m/s.

The priority multiplier mentioned in Section 9-2-1-1 is implemented by multiplying the ρ_{base} by 6[24] for vehicles in the horizontal lane; the value of this multiplier is found through trial and error. However, since the multiplier is mainly relevant to the priority order we will not go deeper into this value in this section.

For the adaptation function of OA-ADMM, the method given in Equations (9-21) and (9-22) is used, with $a = 6$ and $D_{min}^{\phi} = 1.05 D_{min}^{\star}$, where D_{min}^{ϕ} refers to the D_{min} used in Equations (9-21) and (9-22).

[24] This value was found by roughly increasing ρ_{base} until the vehicles properly yield.

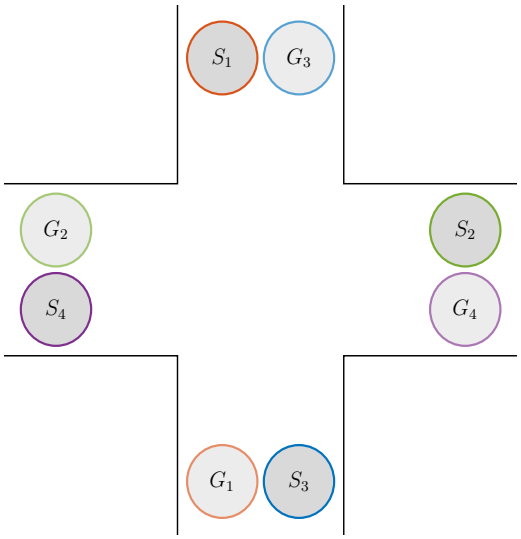


Figure 9-5: Simple conflict resolution simulation environment for comparison between OA-ADMM and ADMM. Agents are circle shapes and considered to be holonomic, S_i represent the initial state for agent i, and G_i represents the goal for agent i.

9-2-1-3 System Constraints

As the intention of the simulation is to investigate the ability of OA-ADMM to be applied for the conflict resolution of autonomous vehicles, it is necessary that it can handle constraints. The constraints can be separated into two parts, the first being constraints on the vehicle dynamics, and the second being the collision avoidance constraints.

The vehicle dynamics constraints include: the maximum input $u_{max} = 20$ m/s^2, the maximum velocity $v_{max} = 6.25$ m/s, the minimum velocity $v_{min} = -1$ m/s, and the system dynamics. The collision avoidance constraints include the minimum distance between vehicles, along with the minimum distance to the edges of the road.

9-2-2 Results

The results of OA-ADMM for a single case with a $\rho_{base} = 1$ and $D_{mult} = 1.75$ is given in Figure 9-6. This shows four vehicles approaching the intersection with identical starting distances and initial velocities, the only difference between the agents is their w_i used in the adaptation function ϕ. Vehicles are shown to maintain a safe distance and avoid a deadlock as Agent1 and Agent3 yield for Agent2 and Agent4. This shows that OA-ADMM MPC is able to achieve collision avoidance, motion planning, control, and conflict resolution with one integrated system.

In Figure 9-6(a), it can be seen that all agents are following their desired reference velocity. Once the vehicle trajectories start to lead towards unsafe states, the vehicles with a lower weight w_i will adjust their trajectories to be closer to that of the collision free z-update copies from the higher weighted vehicles, as depicted in Figures 9-6(b) and 9-6(c). Since the vehicle control is performed using the x-update, the vehicles will not follow trajectories which are infeasible, preventing infeasible solutions from the z-update leading towards deadlocks or

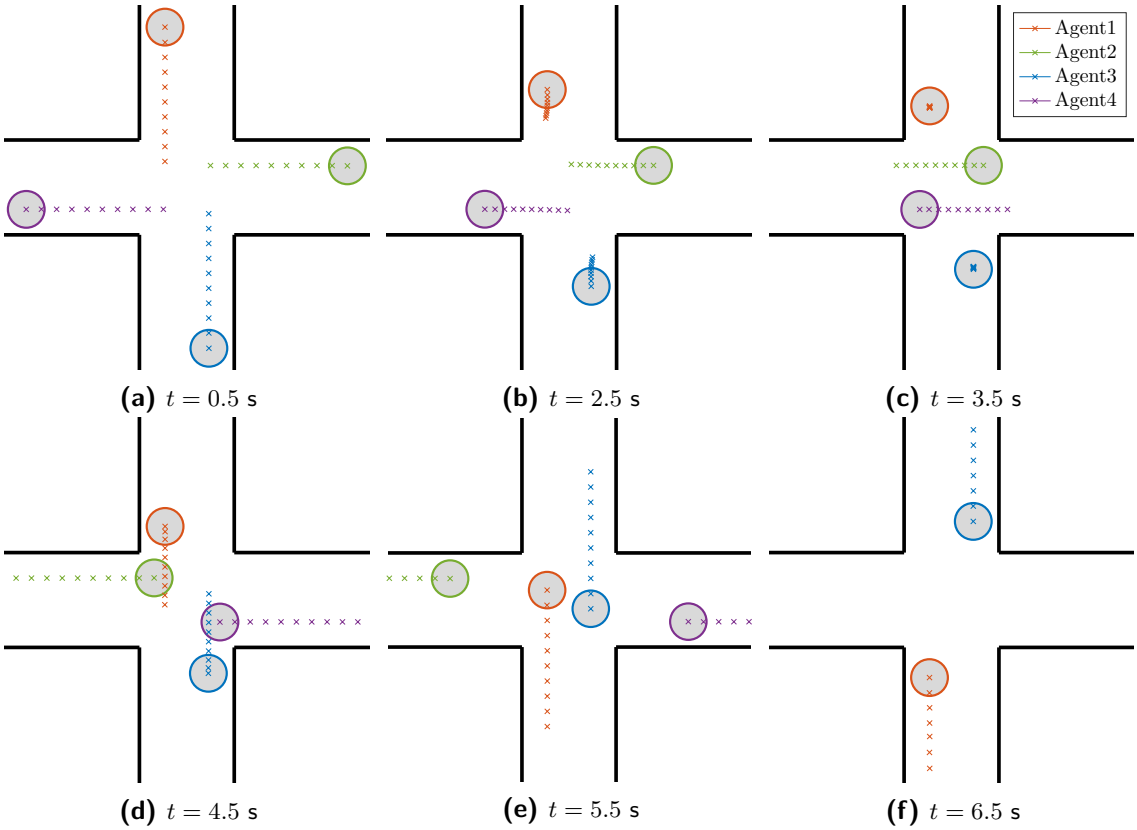


Figure 9-6: Snapshots of decentralized conflict resolution for four vehicles using OA-ADMM and MPC. The finite horizon trajectories of the vehicles is marked by the crosses in their respective colors.

collisions. As the value of D_{mult} is 1.75, the opposing vehicles 2 and 4, or 1 and 3, will slow down when their trajectories are close, as the inflated minimum distance causes the vehicles to repel each other's trajectories. This effect can be reduces by decreasing D_{mult}, or by changing the shape of ϕ.

9-2-2-1 OA-ADMM MPC with eight agents

In order to show the robustness of OA-ADMM an additional four vehicles are added as can be seen in Figure 9-7, the exact same configuration as Figure 9-6 is used, with $\rho_{base} = 1$ and $D_{mult} = 1.75$, with the agents in the horizontal lane still having a higher priority than those in the vertical lane.

Similarly to the four agents case, agents 1 and 3 adjust their trajectories to avoid colliding with agents 2 and 4, as depicted in Figure 9-7(a). At this point, the follower agents, i.e. agents 5-8, have not adjusted their trajectories yet. As agents 2 and 4 slow down when passing by each other, their follower agents adjust their trajectories as well to avoid colliding with the agent in front of them. Figures 9-7(b) and 9-7(c) shows agents 1 and 3 slowing down to a halt to allow vehicles 2 and 4 to pass, whilst agents 5 and 7 slow down to avoid colliding with agents 1 and 3. Given the close proximity of agents 6 and 8, and their equal weight to agents

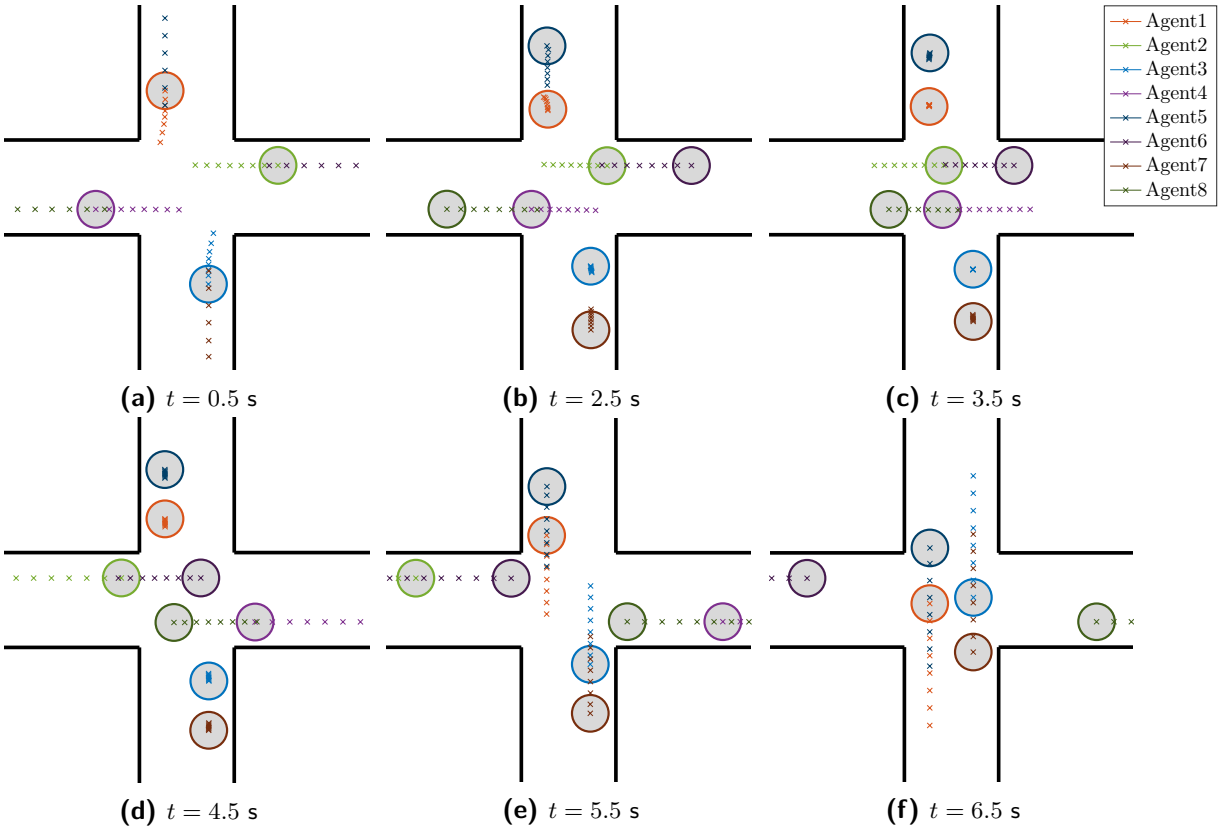


Figure 9-7: Snapshots of decentralized conflict resolution for eight vehicles using OA-ADMM and MPC. The finite horizon trajectories of the vehicles is marked by the crosses in their respective colors.

2 and 4, they simply cross the intersection together with agents 2 and 4. In the meantime, agents 1,3,5, and 7, cannot cross until the intersection is clear. Figures 9-7(e) and 9-7(f) shows the remaining agents crossing the intersection, avoiding any collisions.

9-2-2-2 Comparison OA-ADMM and O-ADMM

To compare the robustness, safety, and ease of use for OA-ADMM and O-ADMM, both methods are simulated for a wide range of parameters of ρ_{base} and D_{mult}. The metrics measures are the conflict resolution time and the mean squared violation (MSV). The conflict resolution time is measured by taking the average of the exit times for all vehicles; the exit time is measured from the beginning of the simulation until the agent has passed the center of the intersection by 7.5 meters. The MSV can be summarized as the mean squared error (MSE) for the constraint violation, where the error is then defined as $\epsilon^{viol} = dist(\boldsymbol{x}_i, \boldsymbol{x}_j) - D_{min}$.

Given the possibilities for deadlocks, a maximum time of 30 seconds is set; this is to prevent spending too much time simulation a case which will never resolve, if the 30 seconds have passed the simulation is cut off, resulting in a timeout. The results for the conflict resolution time using O-ADMM is shown in Figure 9-8, with the time for OA-ADMM shown in Figure 9-10. The mean squared violation is given in Figure 9-9, with the MSV for OA-ADMM shown

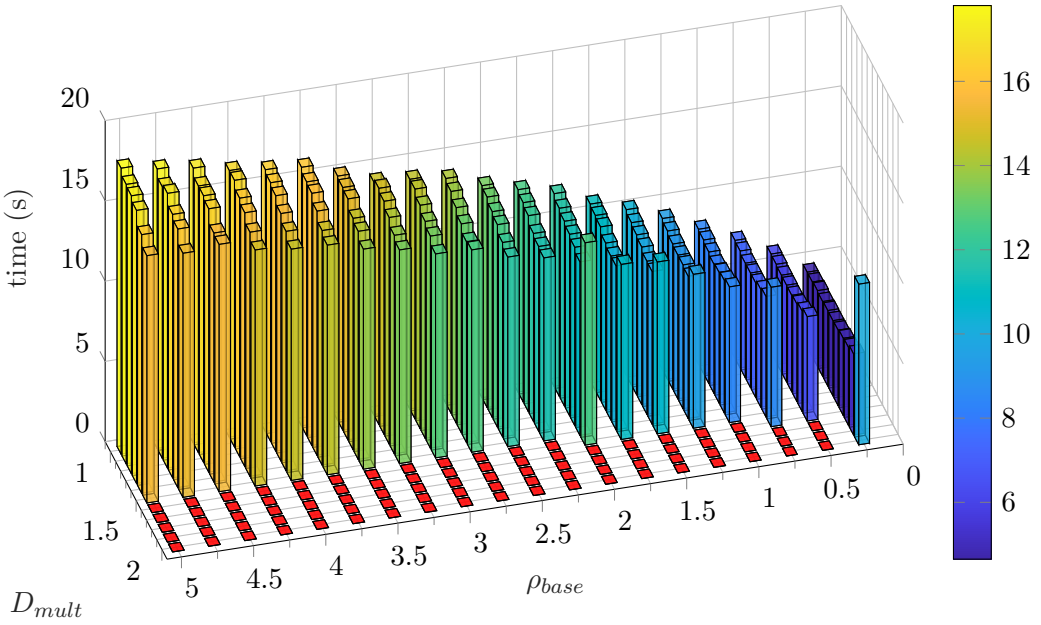


Figure 9-8: Conflict Resolution Time for O-ADMM for various ρ_{base} and D_{mult}. Red bars indicate a time out, e.g. the conflict did not resolve within the maximum time of 30 seconds.

in Figure 9-11.

Algorithm	TO	V	R	Mean time (s)	Min time (s)	Mean resolved time (s)	Min resolved time (s)	Mean MSV (m^2)
O-ADMM	86	133	1	11.720	4.65	10.000	10.00	$2.330 \cdot 10^{-2}$
OA-ADMM	49	83	88	7.473	4.10	7.027	4.70	$8.828 \cdot 10^{-3}$

Table 9-1: Summary of O-ADMM vs OA-ADMM results for 220 cases. TO stands for timeouts, implying that the vehicles did not cross the intersection within 30 seconds; V stands for cases with constraint violations; R stands for resolved cases, implying that no constraint violation has occurred and the vehicles have crossed the intersection within30 seconds.

A summary of the statistics for the simulations is given in Section 9-2-2-2. Resolved cases include all cases where no violation of the constraint occurs and the mean MSV statistic is the average of all cases, except for cases that timed out.

9-3 Discussion

The results obtained in Section 9-2-2 show that OA-ADMM is able to resolve conflicts between multiple vehicles, integrating optimal trajectory planning, control, conflict detection, and conflict resolution, into one multi-agent algorithm. For example, Figure 9-6 shows the vehicles with initially identical trajectories, adjusting their trajectories to avoid collisions, whilst preventing any deadlocks. This is achieved by simply adjusting the weight w in the adaptation functions of Equations (9-21) and (9-22). Compared with traditional conflict resolution

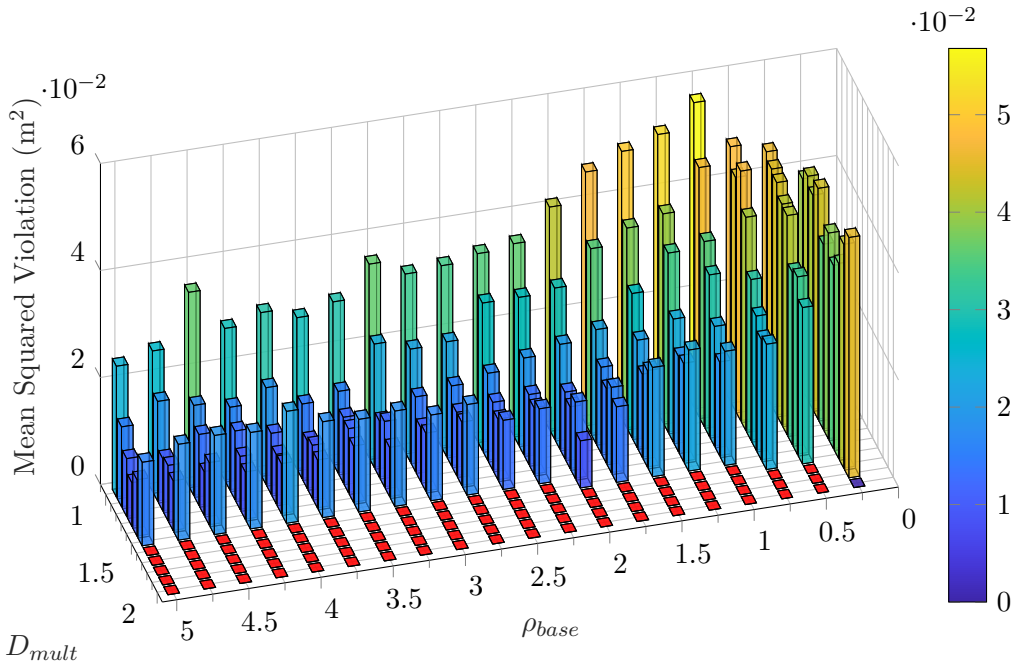


Figure 9-9: Mean squared violation for O-ADMM for various ρ_{base} and D_{mult}. Red bars indicate a time out, e.g. the conflict did not resolve within the maximum time of 30 seconds.

methods, such as that of [22], we have no requirements on the intersection environment, as it is implemented purely as a constraint in the x-update. Whereas the method used in [22] requires the segmentation of the intersection into separate conflict zones, and then separates the conflict resolution and motion planning in a hierarchical scheme.

The results of Figure 9-7 show the ability of OA-ADMM MPC to effectively balance conflict resolution and optimal motion planning using a simple ϕ by simulating the same problem with four additional vehicles. Without any changes, the additional four vehicles automatically adapt safe following trajectories whilst also taking the future trajectories of neighboring vehicles into account.

9-3-1 O-ADMM MPC Results

To compare the effectiveness of the adapting ϕ and μ we investigate the results from Section 9-2-2-2, we first discuss the performance of O-ADMM. The 3D bar plot given in Figure 9-8 represent the time it takes to resolve a conflict for various D_{mult} and ρ_{base}, with the red squares with a time of 0 s representing the simulation timing out.

Increasing the D_{mult} appears to decrease the conflict resolution time, with a few exceptions where an increase in D_{mult} results in an increase in time. The increase in time as D_{mult} increases appears to only occur for $\rho_{base} < 2.5$. It is likely that this behavior is linked with a special case described in Section 9-3-4, where a small change in the value of ρ_{base} and D_{mult} can lead to the system behaving different than expected. Increasing D_{mult} leading to a lower time is likely related with the finite horizon approach, when the value of D_{mult} is larger, possible collisions will be detected earlier and adjusted for. Detecting a collision too late could

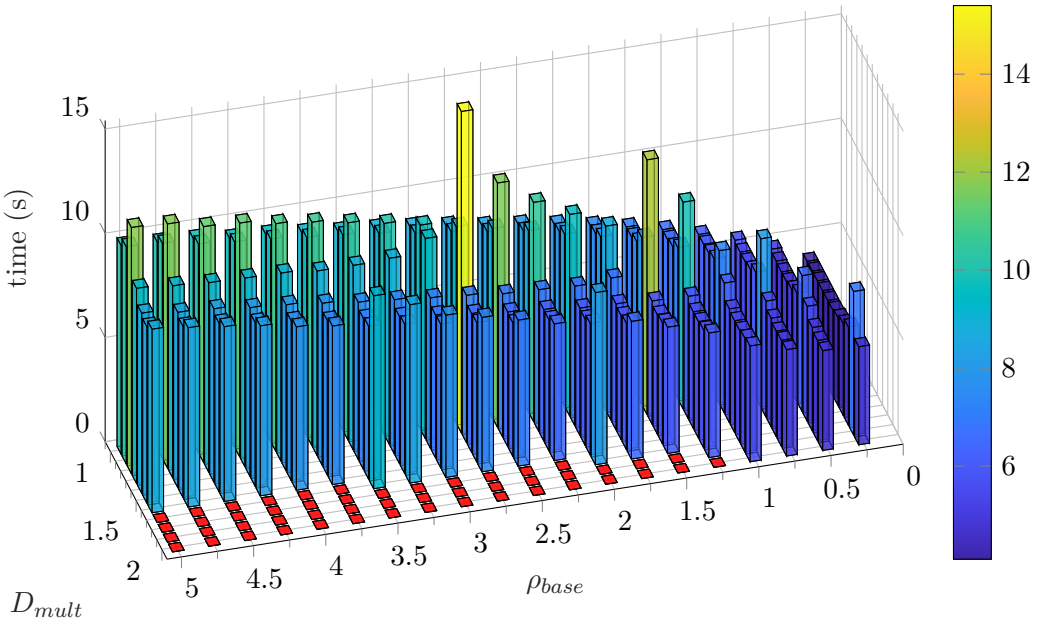


Figure 9-10: Conflict Resolution Time for OA-ADMM for various ρ_{base} and D_{mult}. Red bars indicate a time out, e.g. the conflict did not resolve within the maximum time of 30 seconds.

result in the collision avoidance being unable to adjust the trajectories appropriately, either resulting in collisions or in inefficient behavior such as having to reverse.

Simply increasing D_{mult} can however easily lead the multi-agent system into a deadlock, as the minimum distance can prevent a vehicle from actually crossing the intersection. An example of such a deadlock is given in Figure 9-12, where none of the agents can cross the intersection if the minimum distance is followed. In this example a combination of a high ρ_{base}, which reduces the violation of the collision avoidance constraint, and a restrictive D_{mult} causes the system to be deadlocked. This is also what causes a large amount of timeouts for O-ADMM in Figure 9-8.

When investigating the MSV of O-ADMM, it is apparent that it is difficult to tune the algorithm such that no violation occurs at all, as there is only one case where there is no violation and no timeout for the 220 tested cases. Considering this, it is likely that the algorithm will be prone to environmental changes, the amount of vehicles, or other external factors. It appears that as ρ_{base} increases, the MSV decreases, which is expected as the penalty parameter penalizes the collision avoidance constraint violation. What is however unexpected is that this decrease in violation slows down significantly for $\rho_{base} > 2$, at this point it is possible that the remaining violation cannot be remedied by increasing ρ_{base} for O-ADMM, with changes requires in D_{mult}, the finite horizon, or the sampling time.

Increasing D_{mult} appears to give varying results depending on the value of ρ and the value of D_{mult}. When D_{mult} is small, increasing it generally leads to lower values of the MSV. Increasing the D_{mult} beyond around 1.35, however, seems to result in an increased MSV. Since the MSV is measured using the regular D_{min}, whilst the algorithm attempts to maintain an inflated distance of $D_{mult} \cdot D_{min}$, an intuitive result will have the MSV decreasing as the value of D_{mult} increases.

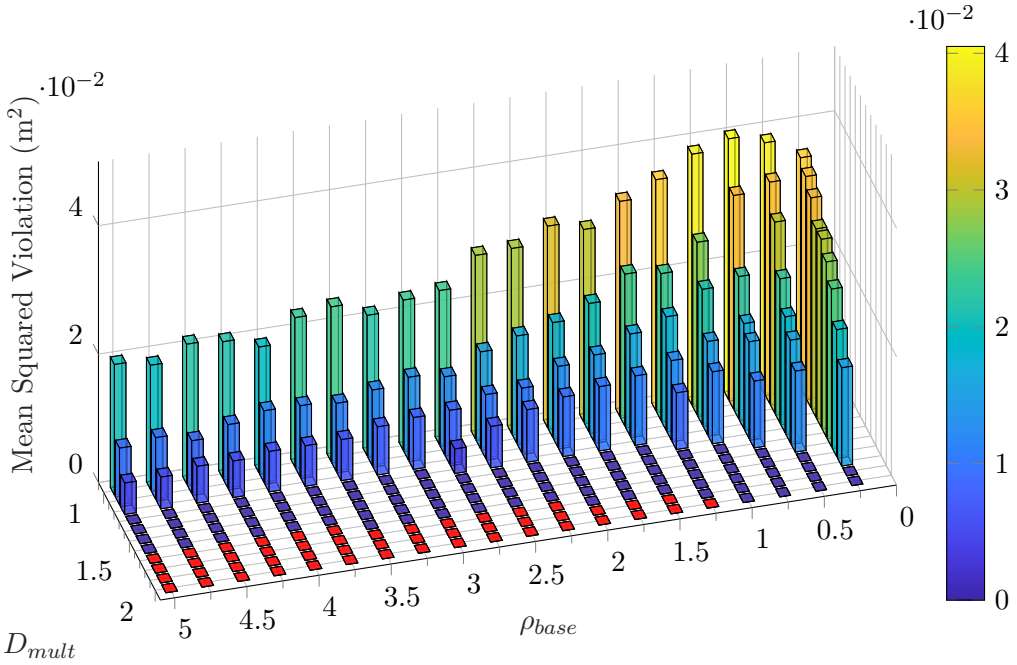


Figure 9-11: Mean squared violation for OA-ADMM for various ρ_{base} and D_{mult}. Red bars indicate a time out, e.g. the conflict did not resolve within the maximum time of 30 seconds.

Investigating the simulations further reveals that the O-ADMM algorithm solves the conflict by crossing the intersection all at once, after which vehicles slowly nudge forward until they can pass by each other for $D_{mult} < 1.6$. Higher values of D_{mult} leads to a different approach where the higher priority vehicles cross first, however this transition leads to an increase in MSV as neither approaches are performed effectively.

9-3-2 OA-ADMM MPC Results

The conflict resolution time for OA-ADMM MPC depicted in Figure 9-10 appears to increase with ρ_{base}, whilst it appears to generally decrease as D_{mult} is increased. The reason that the conflict resolution time increases with ρ_{base} is due to the increased penalty for violating the desired minimum distance. This effect is however reduced, as the ϕ used by OA-ADMM shapes $\boldsymbol{\rho}$ online based on the actual distance and the desired minimum distance used for the adaptation function, i.e. D_{min}^{ϕ}.

Increasing D_{mult} seems to initially increase the conflict resolution time up until $D_{mult} \approx 1.25$, whilst decreasing the time afterwards. There are several possible reasons for this, one of which is explored in Section 9-3-4. Given that the cases where the time increases as D_{mult} increases all have a MSV> 0, it is likely that the increase in D_{mult} has the system transitioning from mostly ignoring the collision avoidance constraints towards actually avoiding collisions. Naturally, it takes vehicles longer to cross the intersection if they are not allowed to drive through each other.

In terms of the mean squared violation (MSV) for OA-ADMM MPC, as depicted in Figure 9-11, it is noticeable that there is a wide range of parameters where there is no violation at

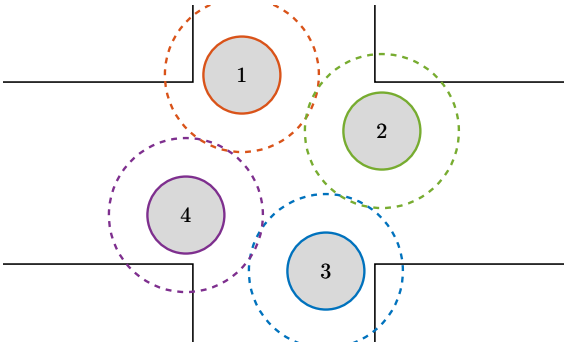


Figure 9-12: Example of a deadlock caused by a combination of D_{mult} and ρ_{base} being too high.

all, outnumbering the amount of combinations where the algorithm times out. The MSV also decreases as D_{mult} is increased regardless of the value of ρ_{base}, this is expected as the algorithm aims to keep a larger distance between vehicles. The reason this does not easily lead towards timeouts is that the adaptation function reduces the effect of D_{mult} as $\boldsymbol{\rho}$ is scaled based on another parameter D^{ϕ}_{min}, which is set to $1.05D_{min}$ in this case. When vehicle trajectories are further away than D^{ϕ}_{min} the value of $\boldsymbol{\rho}$ decreases exponentially, reducing the possibility for D_{mult} to cause timeouts.

9-3-3 O-ADMM vs OA-ADMM

When comparing the conflict resolution times for O-ADMM and OA-ADMM, depicted in Figure 9-8 and Figure 9-10 respectively, it is easy to see that O-ADMM has a lot more timeouts than OA-ADMM. For the cases which do not time out, OA-ADMM seems to be generally affected by changes in parameters compared with O-ADMM, there are however a few outliers for OA-ADMM which do not appear for O-ADMM. These outliers are likely due to some special cases, which are analyzed in Section 9-3-4.

Comparing the mean squared violation (MSV) for both algorithms, depicted in Figure 9-9 and Figure 9-11, the OA-ADMM algorithm appears to decrease the MSV when increasing ρ_{base} and D_{mult} almost everywhere. However, the O-ADMM version appears to have inconsistent results when increasing D_{mult} and ρ_{base}, where increasing D_{mult} results in the MSV first decreasing, followed by the MSV increasing.

Both OA-ADMM and O-ADMM appear to have MSV decrease when ρ_{base} is increased, however this effect seems to plateau significantly earlier for O-ADMM compared with OA-ADMM. This is likely due to adaptation function ϕ decreasing the effective value of ρ_{base} as the penalty vector is decreased when the trajectories are not close to colliding.

When comparing O-ADMM and OA-ADMM in terms of conflict resolution time it is clear that OA-ADMM outperforms O-ADMM when considering cases without any constraint violations, to make this clearer the conflict resolution time is shown only for the safe cases in Figure 9-13 and Figure 9-14. As stated in Section 9-2-2-2, the average conflict resolution time for the safe cases of O-ADMM MPC depicted in Figure 9-13 is 10.00 s; for the safe cases of OA-ADMM MPC there average conflict resolution time is 7.027s. Not only does OA-ADMM perform significantly better than O-ADMM in terms of conflict resolution time, it also has a much larger range of safe parameter combinations. For the tested range of ρ_{base} and D_{mult},

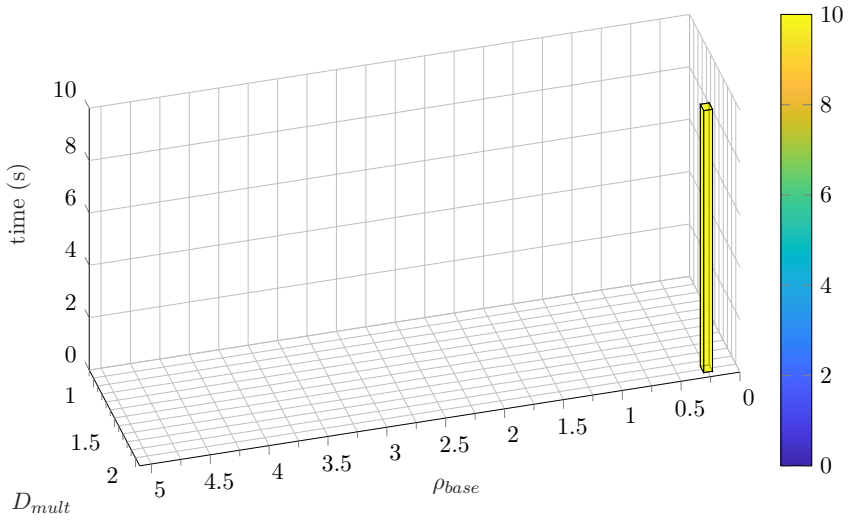


Figure 9-13: Conflict Resolution Time for O-ADMM for all combinations of ρ_{base} and D_{mult} without any constraint violations.

with a total count of 220 cases, OA-ADMM is able to resolve 88 cases within 30 seconds without any constraint violation; whereas O-ADMM is only able to resolve 1 cases within 30 seconds without any constraint violation. The O-ADMM algorithm therefore takes on average around 43% longer to resolve the conflict whilst OA-ADMM has 87 more feasible parameter combinations for the tested cases.

When only considering the shortest times, OA-ADMM appears even further ahead, with a minimum safe time of 4.70 seconds, more than twice as fast as the fastest time for O-ADMM. The mean MSV values also has OA-ADMM a factor lower than O-ADMM, with a MSV of 8.828×10^{-3} m^2 as opposed to the O-ADMM MSV of 2.330×10^{-2} m^2.

From these results it appears that OA-ADMM is a lot more predictable and robust when it comes to adjusting the parameters ρ_{base} and D_{mult}. Whereas O-ADMM requires careful tuning to achieve a safe result, OA-ADMM has a wide range of parameters available where safe and fast conflict resolution can be performed. The bar plots showing the safe conflict resolution times for OA-ADMM in Figure 9-14 seems to indicate that small disturbances in either D_{mult} and/or ρ_{base} will not lead to unsafe behavior or significant inefficiencies.

9-3-4 Special Cases

There are a few special cases in which small changes in the variables can result in undesired changes in the system. One of such cases is when the value D_{mult} is too low, resulting in an ineffective z-update, i.e. the collision avoidance step. An example is shown in Figure 9-15, where the value of D_{mult} affects the direction at which the collision avoidance step adjusts the trajectory of the other vehicle. In the case of the small D_{mult}, the z-update of vehicle B plans does not adjust the trajectory of vehicle A up until $t = 0.2$ s; the next time step, the z-update adjusts the trajectory of vehicle A such that it is no longer in collision with the trajectory of vehicle B, this however forces vehicle A to accelerate, despite B having a higher weight. When examining the case with a larger D_{mult}, this issue is avoided: the larger D_{mult} allows

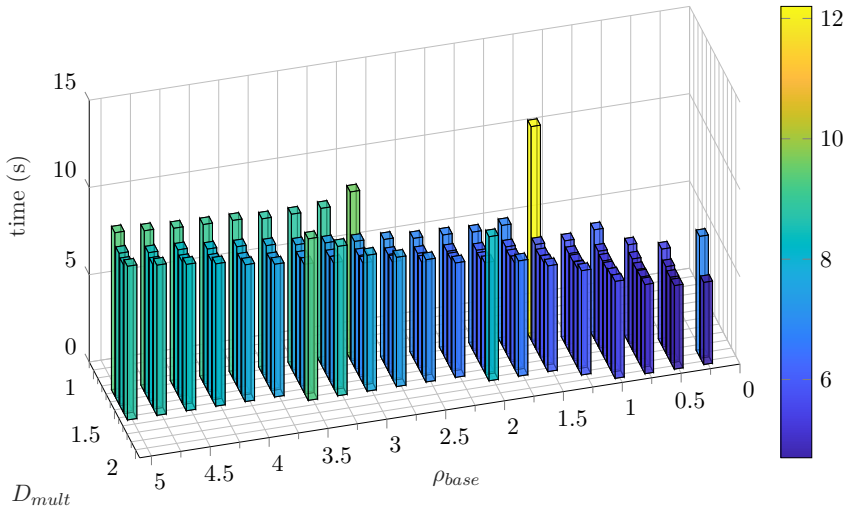
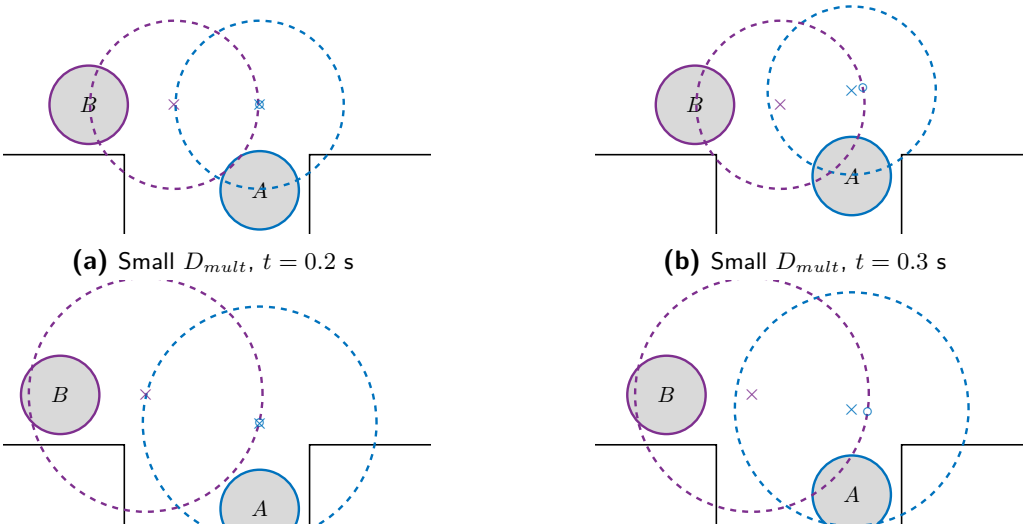


Figure 9-14: Conflict Resolution Time for OA-ADMM for all combinations of ρ_{base} and D_{mult} without any constraint violations.

the z-update to kick in sooner, adjusting the trajectory of vehicle A to require deceleration, allowing B to pass without issues.



(a) Small D_{mult}, $t = 0.2$ s

(b) Small D_{mult}, $t = 0.3$ s

(c) Large D_{mult}, $t = 0$ s

(d) Large D_{mult}, $t = 0.1$ s

Figure 9-15: Example of how D_{mult} affects the system behavior with in this example vehicle B having a higher weight than vehicle A. The $\times$ marks indicate the planned trajectories of the agent with the corresponding color, the $\circ$ marks indicate the trajectory planned by Agent B for Agent A in the OA-ADMM MPC z-update.

Chapter 10

Simulations

This chapter includes a section on the available simulators and the specifics of the chosen simulator (CARLA) in Section 10-1. Afterwards, the specific implementations of the protocols in CARLA is discussed in Section 10-4, including necessary changes and potential shortcomings of the protocols described in Chapter 3.

10-1 Simulator Choice

The choice of simulator mainly affects the accuracy of vehicle dynamics, where ease of use, availability, and customizability should also be taken into consideration. In addition the simulator should be capable of facilitating vehicle to vehicle communication. Aside from established simulators, there also is the option of writing a simulator for the purpose of this analysis.

10-1-1 Groovenet

Groovenet is a hybrid vehicular network simulator that allows communication between real and simulated vehicles using real street map-based topography [122, 123]. It has been used by various studies for simulations such as [19] and [124]. An extension made by [19] to GrooveNet is the inclusion of lane information for roads, this was necessary as Groove Net uses the Tiger / Line data which does not include these [125].

10-1-2 AutoSim

AutoSim is a hybrid emulator and simulator similar to Groovenet and has been referred to as "the next generation of Groovenet" in [126]. The simulator makes use of a modular approach where the vehicle functions have their separate models, such as for localization, motion planning, control, or communication. The simulator also includes built in traffic generation and run time diagnostics tools, which makes it suitable for testing the effectiveness of traffic protocols.

10-1-3 CARLA

CARLA is a recent open-source simulator designed for autonomous driving research, suitable for a wide range of simulations, including low level vehicle control research and higher level traffic and cooperation research [127]. CARLA utilizes Unreal Engine 4 for its foundation, which allows for high fidelity graphical simulations for sensor simulations like LIDAR or cameras. As this is computationally intensive, CARLA also offers a no-rendering mode that is more suitable for traffic simulation. Furthermore, CARLA also includes additional features such as pedestrian modeling, built-in sensor suite, OpenDrive standard based maps, ROS integration, etc.

10-1-4 Discussion

The readily available simulators Groovenet, Autosim, and CARLA all boast similar capabilities in terms of multi vehicle simulations, with CARLA being significantly more common within the field of autonomous driving. CARLA is also widely in use for machine learning, computer vision, and general autonomous driving research. The wide use is beneficial when considering potential future research involving integration of more advanced local behavior, for example the use of observed states when communication is not available. The use of an already existing simulator does however come with the sacrifice of customizability. However, the advantages gained due to its available resources outweighs the need for total control for the simulations in our case. For the aforementioned reasons, the simulator of choice for this work is CARLA.

10-2 Decentralized Protocols

This section analyzes in detail a few decentralized protocols that will be compared against OA-ADMM MPC for benchmarking purposes. The protocols are selected to provide a good representation of currently proposed decentralized protocols. The selected protocols are Advanced Maximum Progression Intersection Protocol (AMP-IP) from [21], and an unnamed protocol from [22], hereafter referred to as Timeslot-based Distributed Conflict Resolution (TDCR). As the original works proposing these protocols do not adhere to the same notation and definitions as in this literature review, some amendments will have to be made. The amendments, however, do not affect the core functionality of the algorithms.

10-2-1 Advanced Maximum Progression - Intersection Protocol (AMP-IP)

AMP-IP is proposed in [21], as a heuristic decentralized conflict resolution protocol where collisions are detected based on the entry and exit times for a given cell in the intersection. The cells are created by overlaying a grid on top of the intersection, dividing the intersection into equally sized cells of a user defined size. If the entry and exit times overlap for a given cell a priority policy is used to determine to passing order. For increased robustness, a safety time margin can be used, whose value is based on the vehicle specifications. The time margin proposed in the paper is 2 seconds.

The protocol relies on communicating the expected entry and exit times for specific cells using messages, where three types of messages are defined. The “ENTER” message indicates that

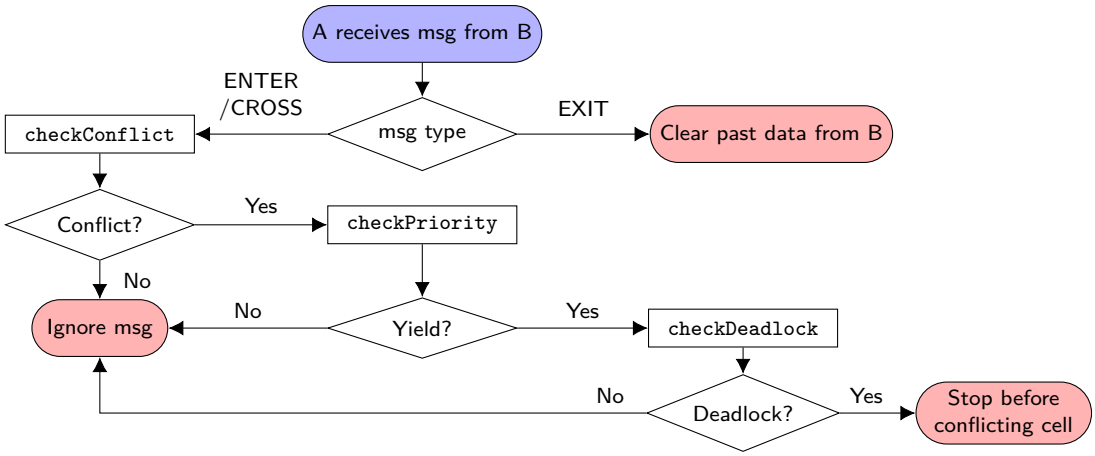


Figure 10-1: A flowchart summarizing how the fixed AMP-IP messages are handled.

the sender vehicle is approaching the intersection, the "CROSS" message indicates that the sender is currently crossing the intersection, the "EXIT" message indicates that the sender has left the intersection. Depending on which message is received, vehicles will check if there is a conflict between itself and the sender. If a conflict is detecting a passing order is determined using a priority protocol, the passing order then determines whether a vehicle is allowed to enter the conflicting cell or not.

The deadlock free proof for AMP-IP relies on the AMP-IP rule: a vehicle A cannot enter the trajectory of vehicle B if the exit time of A at the conflicting cell is later than the arrival time of vehicle B. In the perspective of a vehicle wanting to pass, it can be worded as follows.

Definition 10-2.1 (AMP-IP Rule)**.** Vehicle A is allowed to cross the intersection when it has to yield to vehicle B in cell k if vehicle A can leave cell k before B enters it.

To assess the existence of deadlock there should be the following chain in the yield graph $\{A \to B, B \to C, C \to A\}$, where B can be any chain of vehicles. Given the priority order $\mathcal{P} = \{C, B, A\}$, the AMP-IP rule ensures that A can only enter the path of C if it can exit it before C enters. Consequently, $A \to C$ and $C \to A$ cannot both be true at the same time. The reason being that $C \to A$ is only true for the given priority order if A is occupying the conflicting cell when C is meant to enter it, which goes against the rule that A can only enter the cell if it can also leave it before C enters. By that reasoning, it is claimed that AMP-IP is also deadlock free.

Whilst this is indeed correct, it should be noted that the AMP-IP rule used in the code is not equivalent to the pseudocode and explanation they provide in [21]. The error in judgment lies in the assumption that the exit time of the conflicting cell solely depends on the arrival time at the cell and the ego vehicle dynamics. The equation given for the exit time is

$$T^A(A) + \epsilon^{max} < T^A(B), \tag{10-1}$$

where T^A is the arrival time at the cell and ϵ^{max} is the maximum time it can take for the vehicle to clear the entire cell. This maximum exit time estimate however does not take into account the fact that the ego vehicle might be prevented from exiting the cell at all due to its next cell being another conflict cell.

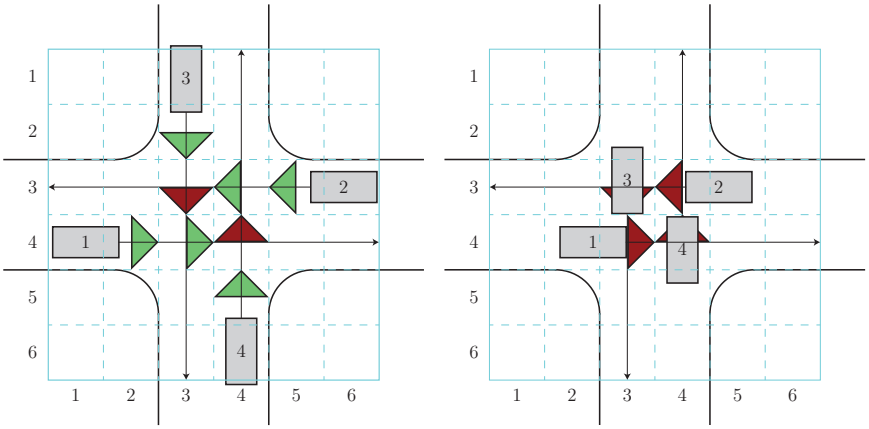


Figure 10-2: Potential deadlock scenario caused by incorrect exit time estimation. The arrows indicate intended paths, and the green and red triangles indicate whether AMP-IP allows the vehicle to enter the next cell or not. On the left is the initial situation that leads to a deadlock, on the right is the situation when a deadlock, or collision, occurs.

Suppose we have four vehicles with priority order $\mathcal{P} = \{1, 2, 3, 4\}$ in the scenario given in Figure 10-2. Vehicle 1 and 2 have priority over 3 and 4 which makes them ignore any messages from 3 and 4. Vehicle 3 receives the CROSS message from 2 and decides to ignore it as $T^A(3) + \epsilon^{max} < T^A(2)$ at cell (3,3) (row,column). Vehicle 3 also receives the CROSS message from vehicle 1 and decides to yield before entering cell (4,3). The same happens for vehicle 4 with respect 1 and 2. On the right of the figure it can be seen what would happen when the version of AMP-IP proposed in [21] is used: vehicles 3 and 4 are waiting for 1 and 2 respectively to cross the conflicting cells. Meanwhile, vehicles 1 and 2 cannot enter their next cell, as doing so would cause a collision with 4 and 3 respectively.

In order for AMP-IP to be deadlock free, a better exit time estimate has to be found, which can take into account the entire system, or the condition for crossing should be stricter. For example, adding the additional requirement that the lower priority vehicle cannot have another conflict zone along its path would prevent the deadlock in Figure 10-2 from occurring. This could potentially be relaxed by taking into account the decisions made for the other conflicts, however which might not be trivial. As finding the best performing AMP-IP rule is beyond the scope of this literature review, we will not develop this concept further in this report.

Further evaluation of AMP-IP shall assume the implementation of an added requirement of having no other conflicts in its path, in order to cross the intersection out of priority order. The flowchart in Figure 10-1 summarizes the AMP-IP protocol with the added deadlock detection in order to prevent the situation in Figure 10-2 from occurring.

10-2-2 Timeslot-based Distributed Conflict Resolution (TDCR)

This distributed conflict resolution protocol has been proposed by Liu *et al.* in [22] utilizing a heuristic priority function, deadlock detection, and optimization based speed profile planning. This is also comparable to [58] that utilizes a distributed model predictive control approach for its control. These methods can also be interpreted as a decentralized variant of the centralized optimization approaches where the passing order is predetermined, such as in described in

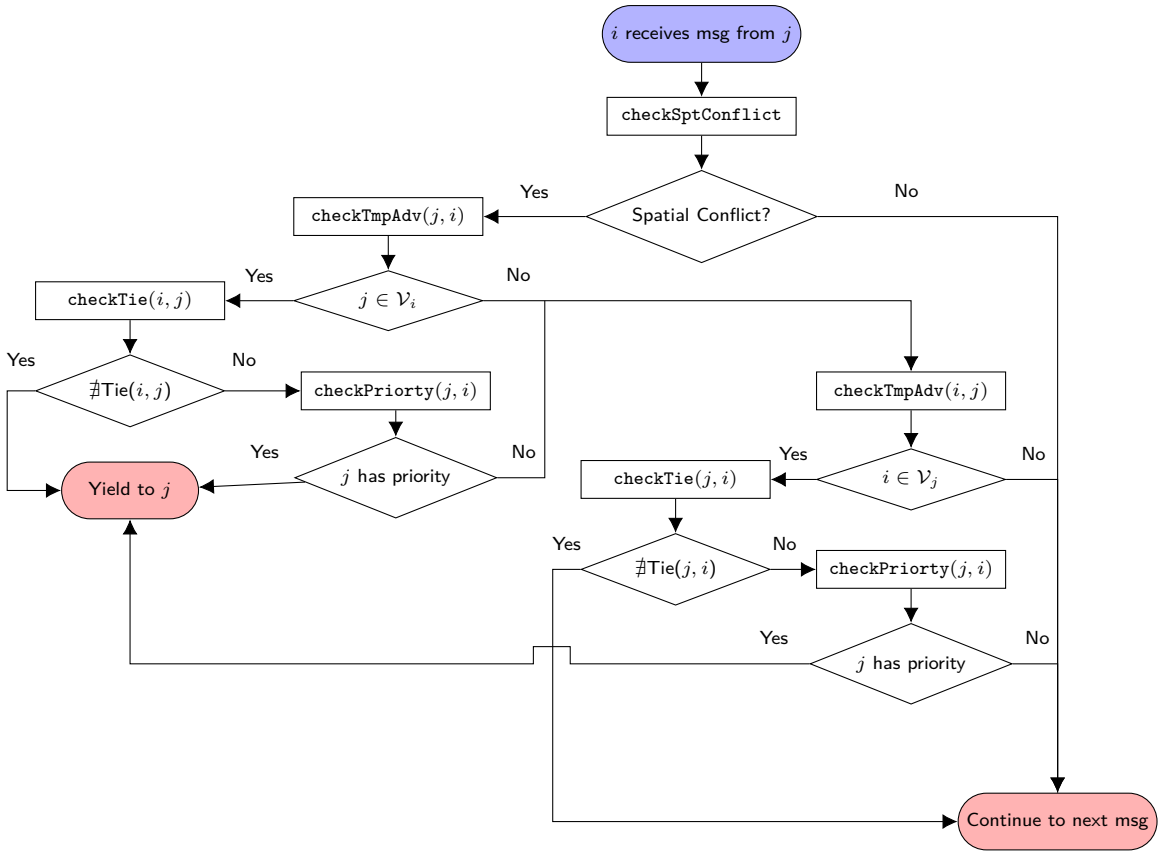


Figure 10-3: A flowchart summarizing the decision making for TDCR, where `checkSptConflict` is a spatial conflict detection algorithm, `checkTmpAdv` checks the *temporal advantages* of the vehicles, `checkTie` is a checks for a *tie*, and `checkPriority` checks for which vehicle has the highest priority.

Section 3-1-3. The protocol differentiates itself from the previous ones by utilizing deadlock detection in order to allow a more flexible priority order. Furthermore, it utilizes timeslots, similar to reservation based systems, which are utilized for its motion planning and control.

The algorithm utilizes two distinct timeslots, a predicted timeslot $\left(\hat{t}_i^A(\cdot), \hat{t}_i^E(\cdot)\right)$ and a reserved timeslot $\left(t_i^A(\cdot), t_i^E(\cdot)\right)$. These timeslots are the entry and exit time for a given conflict region, denoted by $R^c \in \mathcal{R}^c$. The conflict regions are used for spatial conflict detection and can be chosen as needed. The predicted (or estimated) timeslot is obtained from the motion planning and control part of the algorithm, and is used for the temporal conflict detection in the algorithm. The conflict resolution algorithm provides a deadlock free priority order that is used to determine the reserved timeslot from the predicted timeslots. If a vehicle does not have to yield, it will set its reserved timeslot for all conflict zones to be its predicted timeslots, with the note that the exit times are undetermined in the implementation in [22]. The reserved timeslot is then used for the local optimal control problem, which in this case is finding the optimal speed profile for a given vehicle along its path. The protocol will however also function with other control methods, as long as the timeslot constraint is always satisfied. The resulting optimal speed profile is then used for the predicted timeslot in the next iteration.

The protocol utilizes discrete vehicles states (S), these are:

- FIL, first vehicle in any incoming lane.
- IL, other vehicles in an incoming lane.
- I, any vehicle currently crossing the intersection.
- OL, vehicles in an outgoing lane.

The discrete vehicle states are used in three parts of the protocol, the first part being conflict detection: spatial conflicts are only detected for vehicles with a discrete state of FIL or I. The discrete state also affects whether a vehicle is said to have a *temporal advantage* over another vehicle, and finally it affects whether a vehicle is said to have a *tie* with another vehicle. There are 3 possible scenarios in which vehicle i holds a *temporal advantage* over j:

1. $S_i = I, S_j = FIL$, and $\exists R^c \in \mathcal{R}^c : \hat{t}_i^A(R^c) < \hat{t}_j^E(R^c)$. In other words, vehicle i is crossing the intersection, vehicle j is the first vehicle in the incoming lane, and i enters any conflict zone before j leaves it.
2. $S_i = FIL$ & $S_j = I$ and $\forall R^c \in \mathcal{R}^c : \hat{t}_i^E(R^c) < \hat{t}_j^A(R^c)$. In other words, vehicle i is the first vehicle in the incoming lane, vehicle j is crossing the intersection, and i leaves all conflict zone before j enters them.
3. $S_i = S_j = FIL$ or I, and $\exists R^c \in \mathcal{R}^c : \hat{t}_i^A(R^c) < \hat{t}_j^A(R^c)$. In other words, vehicle i has the same discrete state (S) as j, being either FIL or I, and vehicle i enters any conflict zone before j does.

Note that, due to the third condition, i holding a *temporal advantage* over j does not necessarily exclude j holding a temporal advantage over i. Furthermore the set of vehicles holding a temporal advantage over vehicle i is denoted as $\mathcal{V}_i$, with $j \in \mathcal{V}_i$ implying that vehicle j holds a temporal advantage over i.

A *tie* in [22] for vehicle i and j, denoted as $\text{Tie}(i, j)$ holds when:

1. $j \in \mathcal{V}_i$, $S_i = S_j$, and there exists a sequence of vehicles $\{k_1, ..., k_N\}$ with $k_1 = i, k_N = j$ and $S_{k_n} = S_i$, $\forall n$ s.t. $k_n \in \mathcal{V}_{k_{n+1}}$ $\forall n < N$.
2. $j \in \mathcal{V}_i$, $S_i = I, S_j = FIL$, and there exists a sequence of vehicles $\{k_1, ..., k_N\}$ with $k_1 = i, k_N = j$ s.t. $k_n \in \mathcal{V}_{k_{n+1}}$ $\forall n < N$.

An intuitive description for a *tie* is that $\text{Tie}(i, j)$ holds when j holds temporal advantage over i, but i also holds temporal advantage over j (in)directly. The use of a *tie* is comparable with detecting a cycle in a yield graph, which is explored in Section 3-3-3. As the conditions are not symmetric, $\text{Tie}(i, j)$ has no direct effect on $\text{Tie}(j, i)$.

It is further assumed that each vehicle has a unique priority score. In the case of a $\text{Tie}(i, j)$ i holds priority over j if $P(i) < P(k)$ $\forall k \neq i$ (Lower score is higher priority) in the $\text{Tie}(i, j)$ sequence. Liu *et al.* state that this priority score should be such that:

- vehicles in the intersection always have priority over vehicles in the incoming lanes.
- the order determined by priority score should not change over time for vehicles in the same discrete state.

The conflict resolution utilizing these definitions can be summarized in a flow chart in Figure 10-3. The flowchart assumes that vehicles communicate through messages directly with each other without any packet loss or data corruption. Furthermore, the flowchart does not include the yielding mechanism or the motion planning and control part. Yielding occurs by changing

the reserved timeslot as follows for i yielding to j:

$$t_i^A(R^c) = \max\left(t_i^A(R^c), t_j^E(R^c) + \epsilon\right), \forall R^c \in \mathcal{R}_{i,j}^c \tag{10-2}$$

where ϵ is an error margin included to account for any inaccurate prediction, and $\mathcal{R}^c$ is the set of all overlapping conflict zones between i and j. The motion planning and control utilized is a speed profile optimization with the reserved entry time constraints (t_i^A) and no exit time constraints ($t_i^A = \infty$).

10-3 Benchmarking Tool

In order to compare protocols against each other in CARLA, a common benchmark has to be used. As no such benchmark is available at the moment of writing, a benchmark has to be developed[25].

The ability of OA-ADMM MPC to resolve conflicts using a priority order is shown in Section 9-2, to supplement that result, the purpose of the simulations in CARLA is to show the effectiveness of OA-ADMM MPC compared with other methods in terms of conflict resolution efficiency. Another purpose of the simulations is to show the application of OA-ADMM MPC on a more realistic autonomous vehicle, which is significantly more difficult than the simplified holonomic vehicles given in Section 9-2.

The protocols mentioned in Section 10-2 are mainly aimed at reducing total travel delay, hence the main comparison metric will be the delay. The benchmark has the following main requirements and wishes in no particular order:

- Repeatable and consistent results
- Measure protocol performance as per chosen metric
- Modular sub solutions
- Realistic vehicle behavior

These are needed to ensure fair comparisons can be made between the protocols using the benchmark. The benchmark proposed will be referred to as the Modular Conflict Resolution (MCR) benchmark.

10-3-1 Environment

The environment of choice for the MCR benchmark is a 4 way intersection, with an incoming and outgoing lane on each road. This is chosen as it is the most common environment at which conflicts occur. The intersection used for the benchmark is depicted in Figure 10-4, in which an unmanaged intersection in a rural environment is shown. As per the definition set in Section 4-1-1, this intersection is considered a structured environment, because each vehicle has only one possible path to take.

[25]The benchmark can be found here: https://github.com/jerryangit/AddedDelayCRBenchmark

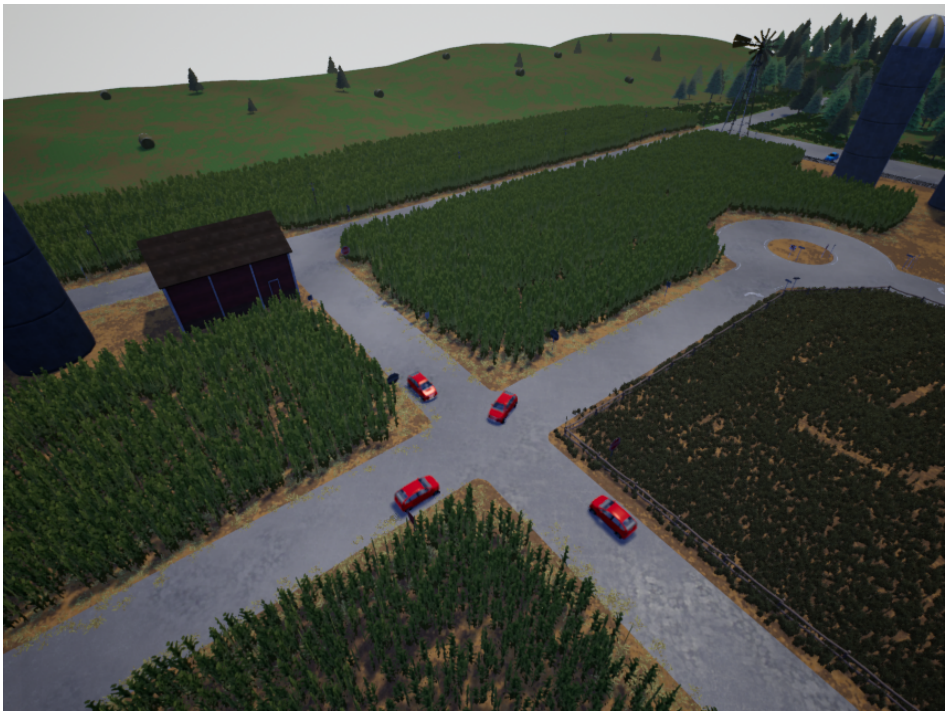


Figure 10-4: The 4 way intersection used for the benchmark in CARLA, with the map being the included Town07. The intersection depicted is a typical rural unmanaged intersection.

10-3-2 Estimated Delay

In order for the MCR benchmark to provide context for the results, a theoretical estimated delay is desired. The theoretical estimated delay for a certain case however is difficult to directly calculate mainly due to the nonlinear nature of the vehicle dynamics, the large combinatorial passing order problem, and calculating how often and to what degree a conflict will occur. Because of that, some assumptions have to be made in order to be able to calculate a usable metric. For that reason, we will investigate the delay when there are two vehicles arriving at the intersection at once. This allows a direct comparison between the travel time for certain conflict resolution protocols in an easy to interpret manner. The main metrics will therefore be travel time and delay, where the delay is measures compared with the no conflict scenario. The no conflict scenario is simply the case where only one vehicle is traversing the intersection at once, serving as the minimum time for a certain trajectory.

In order to calculate the delay for a certain conflict, as depicted in Figure 10-5, some further assumptions are necessary. If we assume that vehicles have a reference velocity, significantly below their maximum velocity we can assume that vehicles have unbounded velocities near the reference velocity. In the case that all vehicles have identical reference velocities, a vehicle following another vehicle taking an identical path will add zero delay to the system. For that reason, we will not investigate the vehicle follower case; the analysis of this case has no added value for the comparison of the chosen conflict resolution protocols. Following the zero delay assumption, the only segment of a trajectory contributing to the delay is the segment where there is an orthogonal component to the trajectory with respect to the trajectory of the yielding vehicle. This component is depicted in Figure 10-6, where the orthogonal component

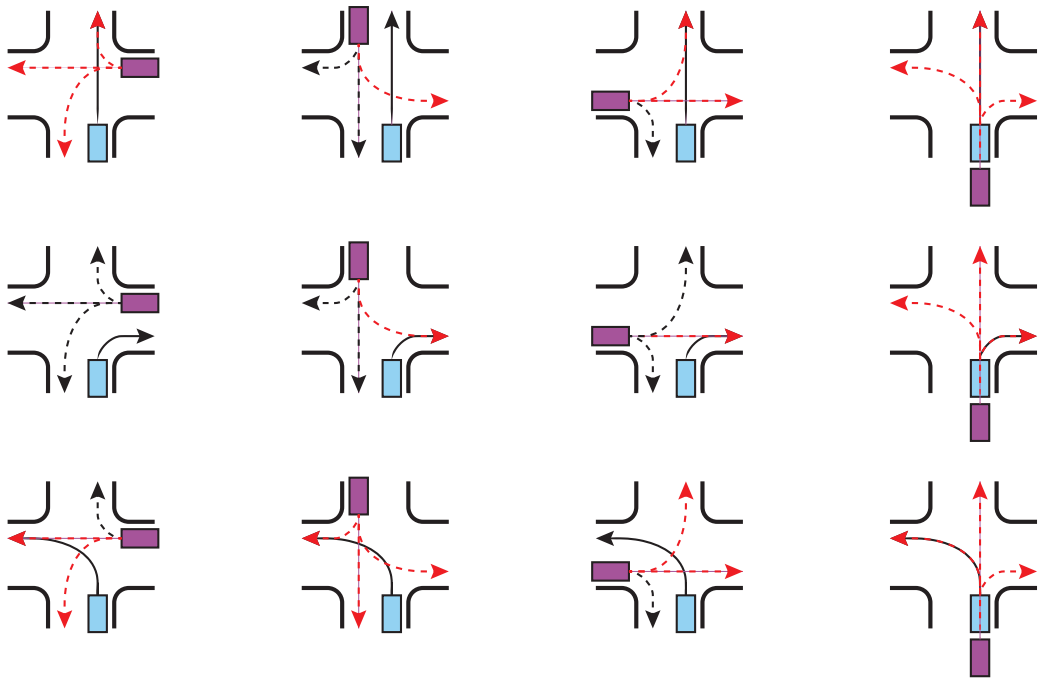


Figure 10-5: All possible conflicts for a small 4 way intersection with two vehicles. Solid lines indicate the desired path for the light blue vehicle, whereas the dashed lines indicate possible paths for the magenta vehicle: conflicting paths are drawn in red.

is called the conflicting component, as this is the portion of the trajectory that contributes to the conflict delay. The delay caused by this conflicting component can be calculated by taking the displacement function along this axis and dividing it by the velocity function along this axis. The estimated added delay $\Delta\hat{t}$ for a single vehicle i which is yielding to j is then

$$\Delta\hat{t}_i^{yield} = \frac{s_{i,j}^{yield}}{v_j^{con}}, \tag{10-3}$$

where $s_{i,j}^{yield}$ is the conflicting distance, and v_j^{con} is the velocity in the direction of the conflicting component. Equation (10-3) is based on the assumption that the yielding vehicle is aware of the trajectory of the other vehicle and can adjust its own trajectory optimally[26].

For simple paths like two vehicles traveling straight perpendicular to each other, finding the exact $s_{i,j}^{yield}$ and v_j^{con} is a relatively easy calculation. However, as these function are difficult to derive for a lot of the possible conflicts a sampling based method utilizing linearization can also be used. If the resolution for the sample is sufficiently high this will have only a marginal difference with the exact result.

The resulting estimated delay only serves as a benchmark; we would like to stress that this is neither an upper bound nor a lower bound for the delays for the given scenarios or for conflict resolution in general. The main benefits of this metric are its relative ease to calculate/estimate

[26]Optimally in this context refers to minimum delay, actual implementations are likely to consider energy consumption an important metric as well.

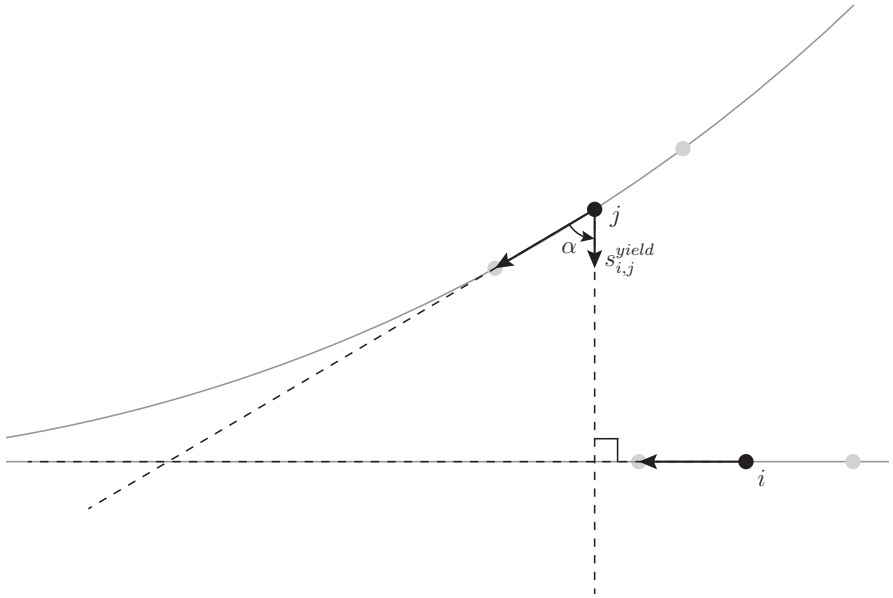


Figure 10-6: Conflicting component of a trajectory, where i and j are samples for two possible maneuvers and $s_{i,j}^{yield}$ is the displacement in the orthogonal direction for j.

and its mathematical definition. The mathematical definition allows the metric to serve as a consistent benchmark regardless of environment and/or simulator, making direct comparisons of conflict resolution protocol efficiencies easier. The main suggested use for this metric would be the percentage or ratio between the actual delay and the estimated delay.

10-4 Implementation

The MCR benchmarking tool is written in Python 3.7.9 for CARLA v0.9.9.4, using its Python API[27]. The benchmarking tool allows the simulation frequency of CARLA to be set independently from the vehicles operating frequency, this feature is to ensure realistic physics, whilst reducing computational load caused by the conflict resolution and vehicle control. In order to achieve consistent and repeatable results the CARLA server will have to be run at a fixed time step. It is also possible to utilize the *no-rendering-mode* in order to disable the graphical 3D rendering.

For the purpose of debugging and visualization the main view used is a simplified view, which is derived from the included "no_rendering_mode.py" from the CARLA PythonAPI examples. The same scenario of Figure 10-4 visualized in this simplified view can be seen in Figure 10-7

10-4-1 Conflict Detection

The main conflict detection algorithm in use will be the grid-based algorithm. The grid-based algorithm functions by overlaying a grid on top of the intersection with configurable dimensions and resolution. The conflict are detecting by comparing which cells will be occupied when

[27]The Python API allows users to access the CARLA simulator through Python programs.

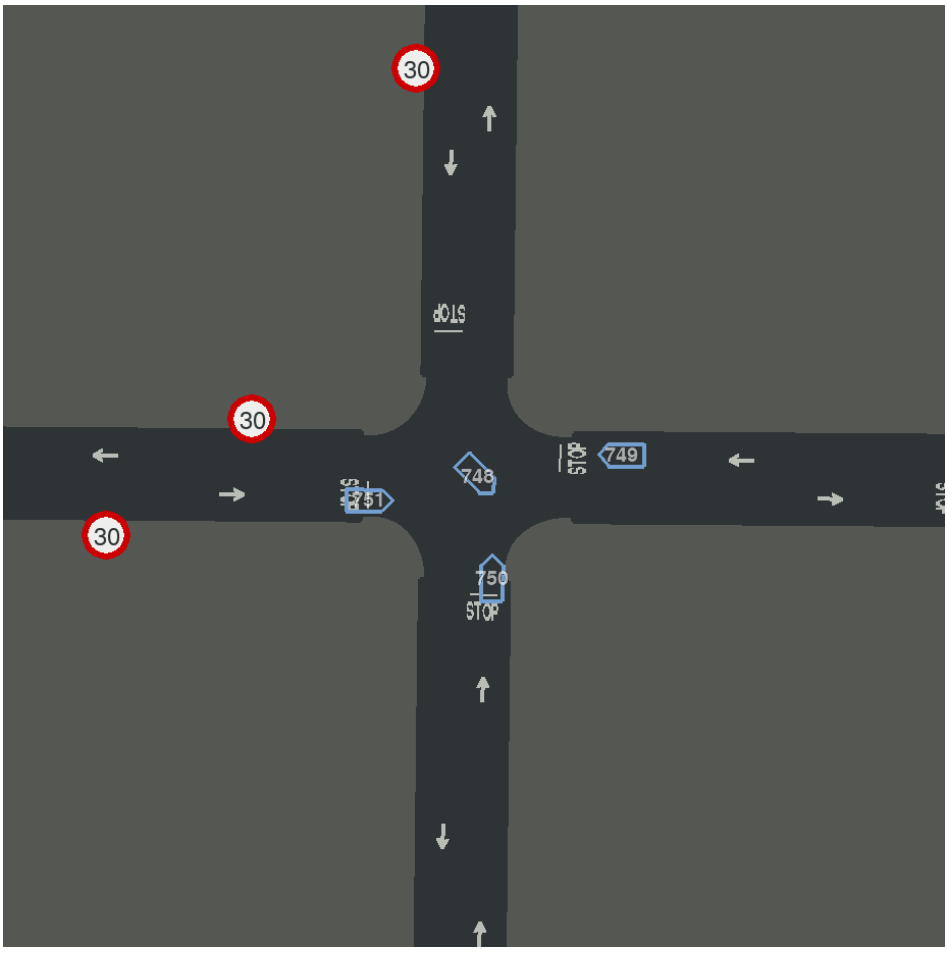


Figure 10-7: The same 4 way intersection as Figure 10-4 when depicted using a slightly modified `no-rendering-mode` view.

traversing the trajectory, if the resolution is too low, the conflict detection can function poorly. Given that both AMP-IP and TDCR require prior knowledge of the intersection map, which is not required for OA-ADMM, the effect of this knowledge is analyzed. This is done by studying the performance of AMP-IP and TDCR using various resolutions. The resolutions used are defined as: high fidelity, i.e. an 8x8 grid, medium fidelity, i.e. a 4x4 grid, and low fidelity, i.e. 1x1 grid. All resolutions are centered at the intersection and have a dimension of 18x18 m. An example of a grid is given in Figure 10-2, which shows a 6x6 grid with a dimension of around 30x30 m.

Detecting whether a vehicle has entered a grid cell is performed by sampling points along the corners and edges of the vehicles, with the default amount being 4 samples on each corner and 10 spread along the edges. When any of the samples first enters a grid cell the vehicle is deemed to have entered that cell until all samples leave that cell.

The temporal conflict detection simply checks for an overlap in the expected occupancy timeslots. For the sake of robustness an extra margin is added with the default being $\epsilon = 0.25$s. The detection is implemented as the following condition:

$$\texttt{conflict if} \quad t_i^A < t_j^E + \epsilon \quad \texttt{and} \quad t_i^E > t_j^A - \epsilon. \tag{10-4}$$

10-4-2 Capsule based collision avoidance

In comparison with the simulations performed in Matlab given in Section 9-2-2, the simulations performed in CARLA are a lot more realistic. When the vehicle is modeled as a double integrator, a sphere-based collision avoidance model is satisfactory; when considering a more advanced model, such as the unicycle or bicycle model, as described in Section 2-3-2, this is no longer ideal. What this means for OA-ADMM MPC and collision avoidance is that it is not longer acceptable to utilize a simple minimum distance constraint like the one given in Equation (9-2), as this only measures the distance between two points. The circular nature of the collision avoidance constraint has a significant downside: in order for a circle to include the entirety of the vehicle, a significant portion of collision avoidance constraint will be overestimating the actual dimensions. For that reason a capsule is used instead, where a capsule can essentially be viewed as a rectangle with a semi-circle placed on both ends.

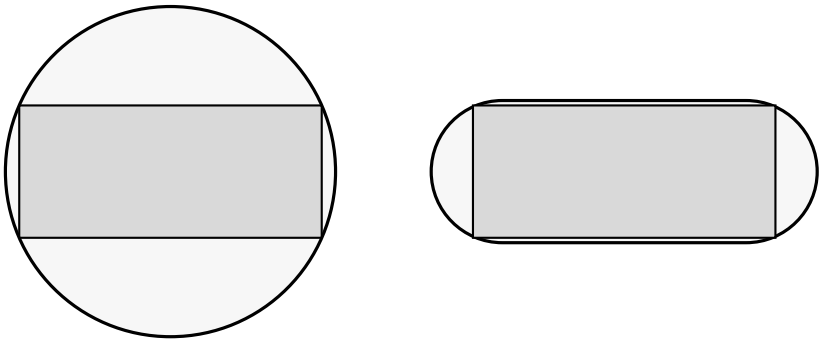


Figure 10-8: The difference between the circular hull and the capsular hull for the car.

The advantage of capsule-based collision avoidance is easily visible in Figure 10-8, where the collision avoidance region is shown for both the circle and the capsule. The capsule shape is a much better approximation for the vehicle shape than the circle. For the example shown in Figure 10-8, which assumes that the car is a rectangle with dimensions of 4 m $\times$1.75 m, the surface area of the rectangle is 7 m^2, whereas the circular hull has a surface area of 14.972 m^2, and the capsular hull has a surface area of 8.971 m^2. The overestimated surface of the circular hull comes down to 54.25 %, which is a majority of the surface area, whereas the overestimated surface of the capsular hull comes down to 21.97 %, which is a significant improvement over the circular hull. The uniform nature of the circular hull has another significant disadvantage; the circular hull extends significantly to its sides, which will lead to constraint violations for vehicles passing from opposing lanes. This can lead to deadlocks, or significant losses in efficiency as the circular is not properly representing the shape of the vehicles. In the example used the dimensions are based on a conventional car, if a bus or a truck is used this difference is even more extreme.

10-4-2-1 Minimum distance between two capsules

The capsule shape requires two parameters to fully define its shape in 2D space: its length l, its radius r. With the addition of its rotation θ, its entire position is 2D space is fully defined. The capsule parameters are depicted in Figure 10-9 for two capsules; alongside the parameters, three points are also defined: the front point F, the center point C and the rear

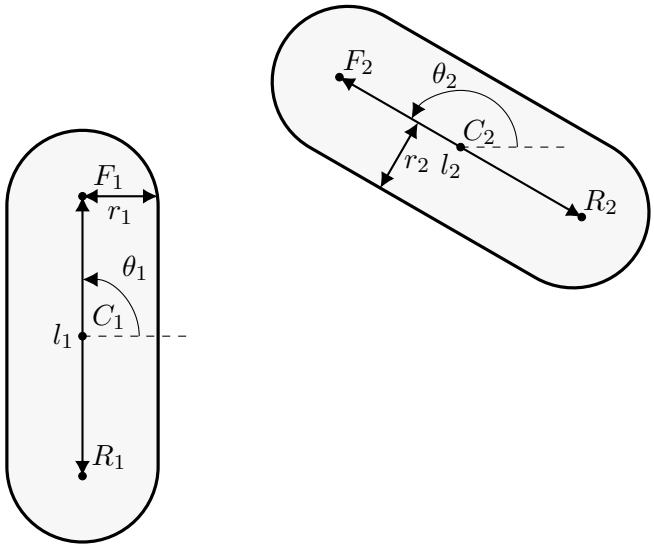


Figure 10-9: Capsule parameters for two capsules and their point definitions.

point R. These points are used to explain how the minimum distance is found between two capsules and how that minimum distance is further linearized.

The minimum distance between two circles in 2D is easily found by taking the Euclidean distance between the two center points and taking away the two radii, e.g.

$$\text{dist}_{cir}(cir_1, cir_2) = \|C_1 - C_2\|_2 - (r_1 + r_2),$$

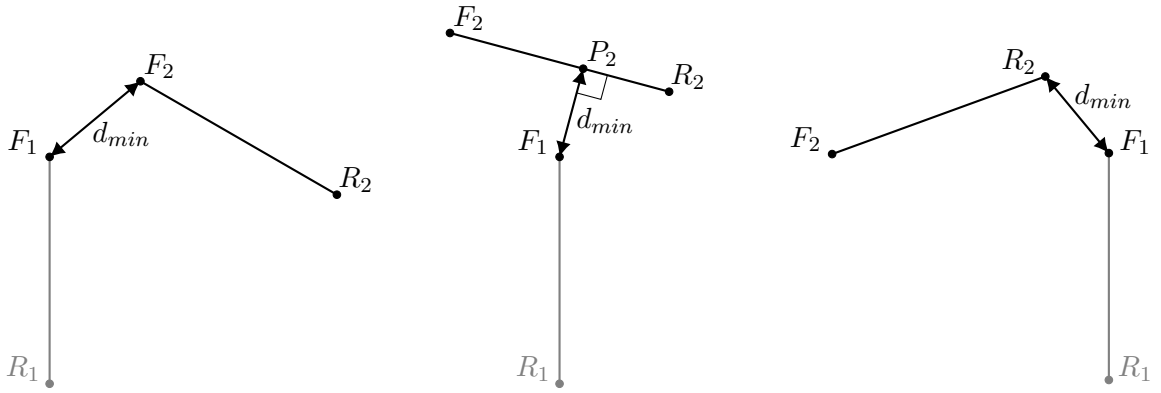
where C_1 and C_2 are the center points of the two circles cir_1 and cir_2, and r_1 and r_2 are the radii of the two circles. Applying capsule-based collision avoidance is however not as easy as circle-based collision avoidance; the minimum distance between two capsules cannot be defined with a single L2-norm. Since a capsule is essentially a line segmented of length l, inflated with a radius r, the minimum distance between two capsules is identical to the minimum distance between two line segments minus the two radii.

The minimum distance d_{min} between two line segments can be found by finding the minimum distance between a point and a line for each end point (F_1, R_1, F_2, R_2) of the two line segments $(\overrightarrow{F_1R_1}, \overrightarrow{F_2R_2})$, i.e.

$$dist_{ls}(ls_1, ls_2) = \min\Big(\text{dist}_{pl}(F_1, \overrightarrow{F_2R_2}), \text{dist}_{pl}(R_1, \overrightarrow{F_2R_2}), \text{dist}_{pl}(F_2, \overrightarrow{F_1R_1}), \text{dist}_{pl}(R_2, \overrightarrow{F_1R_1})\Big), \tag{10-5}$$

where $\text{dist}_{ls}(ls_1, ls_2)$ is the minimum distance between the two line segments ls_1 and ls_2, and $\text{dist}_{pl}(p_1, ls_2)$ is the minimum distance between the point p_1 and the line segment ls_2. In Figure 10-10, three cases are shown for finding the minimum distance between a point and a line segment: Figure 10-10(a) and Figure 10-10(c) show the cases where the minimum distances is between the point F_1 and the ends of the line segment (F_2 and R_2), Figure 10-10(b) shows the case where the minimum distance is between the point F_1 and the point P_2, which is the perpendicular projection of F_1 onto the line segment $\overrightarrow{F_2R_2}$.

In order to find which of the cases shown in Figure 10-10 is applicable a vector projection can



(a) F_1 to F_2 **(b)** F_1 to P_2 **(c)** F_1 to R_2

Figure 10-10: Three essential cases for finding the minimum distance d_{min} between two capsules, with the definitions being identical to that of Figure 10-9. In Figure 10-10(a), d_{min} is the distance between F_1 and F_2; in Figure 10-10(b), d_{min} is the distance between F_1 and P_2, which is the perpendicular distance between F_1 and $\overrightarrow{F_2R_2}$; in Figure 10-10(c) d_{min} is the distance between F_1 and F_2. The other cases can be achieved by swapping F_1 with R_1, or by swapping the the subscripts.

be performed from $\overrightarrow{F_2F_1}$ onto the line segment $\overrightarrow{F_2R_2}$:

$$\overrightarrow{F_2P_2} = \text{proj}_{\overrightarrow{F_2R_2}} \overrightarrow{F_2F_1}.$$

Using this projection we can easily determine which case is relevant by measuring the magnitude m of $\overrightarrow{F_2P_2}$ relative to $\overrightarrow{F_2R_2}$, i.e.

$$m_1 = \frac{\overrightarrow{F_2F_1} \cdot \overrightarrow{F_2R_2}}{\|\overrightarrow{F_2R_2}\|}.$$

If m_1 is negative, then d_{min} is measured between F_1 and F_2; if m_1 is smaller than 1, then d_{min} is measured between F_1 and P_2; if m_1 is larger than 1, then d_{min} is measured between F_1 and R_2. In mathematical form this can be written as

$$\text{dist}_{pl}(F_1, \overrightarrow{F_2R_2}) = \begin{cases} \|\boldsymbol{F}_1 - \boldsymbol{F}_2\| & m_1 \leq 0 \\ \|\boldsymbol{F}_1 - \boldsymbol{P}_2\| & 0 < m_1 < 1 \\ \|\boldsymbol{F}_1 - \boldsymbol{R}_2\| & m_1 \geq 1, \end{cases} \tag{10-6}$$

where $\boldsymbol{F}_1$ is the vector to point F_1. The $\text{dist}_{pl}(\cdot)$ in Equation (10-6) is the equation for the minimum distance between a single point and a single line segment, this has to be performed four times in order to get the d_{min} between two line segments as per Equation (10-5) (dist_{ls}).

To get the distance between two capsules, the radii of the capsules (r_1, r_2) should be subtracted from the dist_{ls} given in Equation (10-6). The minimum distance between two capsules is therefore

$$\text{dist}_{cap}(cap_1, cap_2) = \text{dist}_{ls}(ls_1, ls_2) - (r_1 + r_2), \tag{10-7}$$

where cap_1 and cap_2 are two capsules, and ls_1 and ls_2 are their corresponding line segments.

10-4-2-2 Linearized minimum distance between two capsules

The equation for the minimum distance between two capsules is given in Equation (10-7), which requires Equation (10-6) and Equation (10-5). In order for the capsule-based collision avoidance to be implemented into OA-ADMM MPC, the function will have to be linearized. Regardless of which of the three cases from Equation (10-6) is applicable, the distance function is the Euclidean distance between two points. The function for the Euclidean distance constraint two points i and j, with a minimum distance D_{min}, is $\text{dist}_p(p_i, p_j) \geq 0$, with $\text{dist}_p(p_i, p_j)$ being

$$\text{dist}_p(p_i, p_j) = \begin{bmatrix} \boldsymbol{p}_i^\top & \boldsymbol{p}_j^\top \end{bmatrix} \begin{bmatrix} I & -I \\ -I & I \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_i \\ \boldsymbol{p}_j \end{bmatrix} - D_{min}^2, \tag{10-8}$$

where $\boldsymbol{p}_i \in \mathbb{R}^2$ is the vector to point i, and $\boldsymbol{I} \in \mathbb{R}^{2\times 2}$ is the identity matrix. To linearize this function the first order taylor expansion is taken around $\bar{\boldsymbol{p}}_i$ and $\bar{\boldsymbol{p}}_j$, i.e.

$$\text{ldist}_p(p_i, p_j) = \begin{bmatrix} \bar{\boldsymbol{p}}_i^\top & \bar{\boldsymbol{p}}_j^\top \end{bmatrix} \begin{bmatrix} I & -I \\ -I & I \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{p}}_i \\ \bar{\boldsymbol{p}}_j \end{bmatrix} + 2 \begin{bmatrix} \bar{\boldsymbol{p}}_i^\top & \bar{\boldsymbol{p}}_j^\top \end{bmatrix} \begin{bmatrix} I & -I \\ -I & I \end{bmatrix} \left(\begin{bmatrix} \boldsymbol{p}_i \\ \boldsymbol{p}_j \end{bmatrix} - \begin{bmatrix} \bar{\boldsymbol{p}}_i \\ \bar{\boldsymbol{p}}_j \end{bmatrix} \right) - D_{min}^2, \tag{10-9}$$

where $\text{ldist}_p(p_i, p_j)$ stands for the linearized distance between point p_i and p_j, and $\bar{\boldsymbol{p}}_i$ is the linearization point of $\boldsymbol{p}_i$. In our MPC implementation the state vector only includes the x and y coordinates of vehicle at its geometric center. Because of that, a transformation has to be applied to the states in the state vector in order to refer to the relevant point of the capsule (F, R, or P), i.e.

$$\boldsymbol{p}_i = \boldsymbol{c}_i + \boldsymbol{v}_i, \tag{10-10}$$

where $\boldsymbol{c}_i$ is the center of the capsule, and $\boldsymbol{v}_i$ is a vector pointing from the center to the relevant point.

To summarize, the distance function between two capsules is given in Equation (10-7). To evaluate this function, the distance between the two line segments of the capsules has to be found. Finding this distance requires the minimum distance from each end point to the opposing line segment as stated in Equation (10-5), this can be found using Equation (10-6). The distance between a point and a line segment always results in the distance between two points, which can be linearized as per Equation (10-9). To implement in MPC, $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ should be written in the form of Equation (10-10). Finally, the resulting equation can be implemented as a linear inequality constraint, which has the form of $\boldsymbol{Ax} \leq \boldsymbol{b}$, in the MPC quadratic programming problem.

10-4-3 Deadlock Detection

Deadlock detection is currently only utilized by TDCR for its *tie* detection algorithm. The implementation utilizes a Breadth first search starting from a specified vehicle exploring the edges, terminating when all nodes have been explored or a cycle is found. This implementation is not a general cycle detection algorithm as described in Section 3-3-3 as it only detects whether a cycle exists between two agents. Additionally the cycle detection requires all vehicles to share their *temporal advantage* graph in order for the detection to be completed locally.

10-4-4 Motion Planning and Control

The motion planning and control is separated into path planning, velocity planning and velocity control. This separation allows the utilization of the global route planner included with the CARLA PythonAPI (`agents.navigation`). The algorithm utilizes A* in combination with CARLA waypoints in order to generate a path from any point to any other point, with the waypoints utilizing the OpenDRIVE [128] format for the road information.

The velocity planner decides on a reference velocity which will be fed to the lower level velocity controller. For AMP-IP a simple car-following algorithm is used, where a desired reference velocity is maintained unless it results in a collision with another vehicle or in the vehicle entering the intersection when it should yield. In these cases, the desired velocity is adjusted based on the current velocity and the distance to the vehicle or zone.

Whilst [22] suggested the use of speed profile planning via temporal optimization for TDCR [129], it is possible to use any constrained speed profile planner. A common choice for constrained optimization is an MPC approach, also compatible with the interest in MPC in the automotive industry [130]. As the goal is velocity planning, the states of the system are simplified to be $\boldsymbol{x} = [s, v]^\top, u = a$, where s is the displacement along the path, v is the velocity along the path, and a is the acceleration implemented as $v[k+1] = v[k] + a \cdot dt$ which is fed to the low level velocity controller. The model is a simple discrete integrator system, enhanced with states for the deviation from the reference velocity; additionally upper and lower bounds are set for the velocity and acceleration as inequality constraints. The temporal constraints are implemented as state constraints for a certain time step. For example the entry time constraint $t^A(\cdot) > 1.4s$ with $dt = 0.1$ and $N = 20$ is implemented as the inequality constraint $s[14] < s^A(\cdot)$, where $s[14]$ is the first state at time step 14 and $s^A(\cdot)$ is the displacement at which the vehicle enters a certain region. The cost function J is a quadratic cost function that penalizes control inputs and deviation from the reference velocity.

Due to the nonlinear nature of the vehicle dynamics in CARLA, designing an optimal controller is difficult. The nonlinear behavior is mainly attributed to two main mechanisms: the nonlinear engine torque curve, and air, tire, and internal frictions; additionally gear shifting, inertia, and the steering/velocity curve can further complicate things. To avoid over complicating the velocity controller, and since the velocity controller is used universally among all protocols, a PID controller has been used for the velocity control.

10-4-4-1 OA-ADMM Model Predictive Control

The OA-ADMM MPC approach utilizes Model Predictive Control for trajectory optimization, using a kinematic bicycle model as given in Section 2-3-2. Given the nonlinear nature of the kinematic model, linearization has to be performed to get reasonable results. We repeat the

kinematic bicycle model here for convenience:

$$\dot{x} = v\cos(\theta + \psi) \tag{10-11a}$$
$$\dot{y} = v\sin(\theta + \psi) \tag{10-11b}$$
$$\dot{v} = a \tag{10-11c}$$
$$\dot{\theta}_i = \frac{v}{l^r}\sin\psi \tag{10-11d}$$
$$\psi_i = \tan^{-1}\left(\tan\phi\frac{l^r}{l^r + l^f}\right), \tag{10-11e}$$

Given that the control inputs in CARLA are the throttle and the steering angle the state vector is $\boldsymbol{x} = \begin{bmatrix} x, y, v \end{bmatrix}^\top$ and the input vector is $\boldsymbol{u} = \begin{bmatrix} a, \phi \end{bmatrix}^\top$. To linearize these, the small angle approximation around $\bar{\boldsymbol{x}} = \begin{bmatrix} \bar{x}, \bar{y}, \bar{v} \end{bmatrix}^\top$, $\bar{\boldsymbol{u}} = \begin{bmatrix} \bar{a}, \bar{\phi} \end{bmatrix}^\top$, and $\bar{\psi} = \tan^{-1}\left(\tan\bar{\phi}\frac{l^r}{l^r+l^f}\right)$,is used. Additionally, a local coordinate system is used, resulting in $\theta = 0$, taking these steps into account the linearized equations are:

$$\dot{\bar{x}} = v\cos(\bar{\psi}) \tag{10-12a}$$
$$\dot{\bar{y}} = v\sin(\bar{\psi}) \tag{10-12b}$$
$$\dot{\bar{v}} = a \tag{10-12c}$$

where the equations are linearized around the state vector $\bar{\boldsymbol{x}}$ and a local coordinate system is used. The linearized formulation is only accurate near the linearization point, for this reason adaptive MPC is used. Adaptive MPC linearizes the system at each control step, allowing the linearized system to maintain accuracy whilst avoiding having to deal with difficult nonlinear systems. The final step is to discretize the linearized equations, which results in the following state space matrices:

$$\begin{aligned} \boldsymbol{A}_d &= \begin{bmatrix} 1 & 0 & \cos(\bar{\psi})\mathrm{dt} \\ 0 & 1 & \sin(\bar{\psi})\mathrm{dt} \\ 0 & 0 & 1 \end{bmatrix} \\ \boldsymbol{B}_d &= \begin{bmatrix} 0 & -\bar{\phi}\cdot\mathrm{dt}/2 \\ 0 & \bar{\phi}\cdot\mathrm{dt}/2 \\ \mathrm{dt} & 0 \end{bmatrix} \end{aligned} \tag{10-13}$$

10-4-5 OA-ADMM Functions

The implementation of OA-ADMM MPC requires the design of two functions, the adaptation function ϕ and the similarity function μ. Given that the goal is to achieve safe and robust conflict resolution the adaptation function is using a similar approach to the function used in Section 9-1-2. The functions used in CARLA are

$$\phi(\cdot,k)_{ij} = \begin{cases} w_i\boldsymbol{\phi}^{min}, \text{ if } \left(\frac{D_{min}}{\mathrm{dist}_{cap}(\boldsymbol{x}_i^k,\boldsymbol{x}_j^k)}\right)^a < \boldsymbol{\phi}^{min} \\ w_i\boldsymbol{\phi}^{max}, \text{ if } \left(\frac{D_{min}}{\mathrm{dist}_{cap}(\boldsymbol{x}_i^k,\boldsymbol{x}_j^k)}\right)^a > \boldsymbol{\phi}^{max} \\ w_i\left(\frac{D_{min}}{\mathrm{dist}_{cap}(\boldsymbol{x}_i^k,\boldsymbol{x}_j^k)}\right)^a, \text{ otherwise} \end{cases} \tag{10-14}$$

and

$$\phi(\cdot,k)_{ii} = w_i \frac{1}{N_i} \sum_{j \in \mathcal{N}_i} \left(\phi(\cdot,k)_{ij}\right)^a, \tag{10-15}$$

which only differs from Equations (9-21) and (9-22) in the used of the capsule based distance function $dist_{cap}$ and the bounded value of ϕ_{ij} to improve the robustness. The parameters used in the simulation are $a = 1.75, D_{min} = 1.25 \cdot 2cap_r, \boldsymbol{\phi}^{max} = 2 \cdot \mathbf{1}$, and $\boldsymbol{\phi}^{min} = 0.01 \cdot \mathbf{1}$, where cap_r is the capsule radius and $\mathbf{1}$ is a vector containing only ones. The minimum bound serves as a guarantee that $\rho > 0$, preventing problems related to ill-conditioning in the solver, the maximum bound serves to limit the possibility for disturbances to destabilize the system.

The similarity function implemented is a simple one, based on the idea that the relevance of the previous $\boldsymbol{\lambda}$ is positively correlated to the value of $\boldsymbol{\rho}$. An intuitive explanation for the conflict resolution case is to view $\boldsymbol{\lambda}$ as a penalty by OA-ADMM aiming to enforce the collision avoidance constraint. When a collision is close to occurring, the value of $\boldsymbol{\rho}$ will be higher due to the design of the adaptation function ϕ. In this case, it is desirable that to increase the penalty $\boldsymbol{\lambda}$ to enforce the constraint. However, when the system is far from collision, the relevance of the previous $\boldsymbol{\lambda}$ is diminished and can result in unnecessary suboptimality. This simple idea is implemented as follows:

$$\mu(\cdot,k) = \eta\mu(\cdot,k-1) + (1-\eta)\min(\rho_{JI}\frac{1}{w_i}, \mathbf{1}), \tag{10-16}$$

where the elements of μ are bounded to be less or equal to one, along with a weighted average between the current and the previous value of μ. The weighted average is scaled with $0 \leq \eta \leq 1$ acting as a simple filter to reduce the effects of disturbances. Also note that μ returns a vector, allowing the element-wise update of the Lagrange multiplier $\boldsymbol{\lambda}$.

10-5 Simulation Results

The simulations are carried out in CARLA using the benchmarking tool described in Section 10-3. The metrics measured are the total travel time per vehicle for their respective cases, these are compared against the no conflict case for each respective protocol to get the added delay caused by each protocol. The no conflict case for each protocol simulates the same amount of vehicles with the same exact reference velocities, ensured by the identical random seeds, in order to reduce the effect of the different control approaches. A major difference between OA-ADMM MPC and the traditional methods of AMP-IP and TDCR lies that OA-ADMM MPC does not require the map of the environment beforehand. In order to attempt to show the effects of this prior knowledge, the traditional approaches are simulated for the three different fidelity cases mentioned in Section 10-4-1. Note that AMP-IP and TDCR have been tuned to be optimal where possible, except for the prior knowledge. Additionally, these simulations are focused on the performance of one implementation of OA-ADMM MPC, with potential changes in adaptation and similarity functions being beyond the scope of this thesis.

All the protocols are tested by spawning vehicles at equal distance to the intersection center with a reference velocity of 4 m/s, with uniformly distributed variations between -0.15 m/s and 0.15 m/s. All the possible cases depicted in Figure 10-5, except for the vehicle follower case, are simulated in threefold and averaged out. The simulator is ran at a frequency of

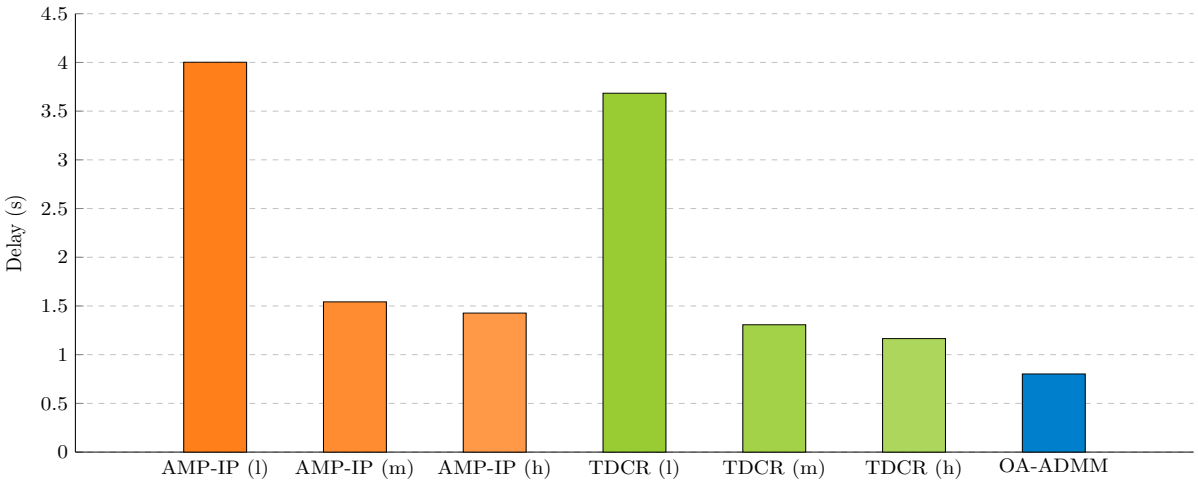


Figure 10-11: Comparison of the delay values for the decentralized protocols.

160 Hz, with the vehicles running the protocols at 20 Hz, however, vehicles perform the low level control at 40 Hz to reduce instability in the low level control.

The average delay for the AMP-IP, TDCR, and OA-ADMM is shown in the form of a bar figure in Figure 10-11, where AMP-IP and TDCR are shown separately for each of their grid fidelity cases.

Protocol (case)	Average Time (s)	Average Delay (s)	Average Added Delay
AMP-IP (no conflicts)	17.804	(−)	(−)
AMP-IP (low fidelity)	21.806	4.002	3.593
AMP-IP (medium fidelity)	19.346	1.542	1.133
AMP-IP (high fidelity)	19.231	1.427	1.018
TDCR (no conflicts)	17.809	(−)	(−)
TDCR (low fidelity)	21.493	3.684	3.275
TDCR (medium fidelity)	19.117	1.307	0.898
TDCR (high fidelity)	18.974	1.165	0.756
OA-ADMM (no conflicts)	18.158	(−)	(−)
OA-ADMM	18.960	0.802	0.394

Table 10-1: Summary of AMP-IP, TDCR, and OA-ADMM results for the Carla simulations using the MCR benchmark. The average delay is measured relative to the no conflict case, the average added delay is measured against the average estimated delay (0.4089 s) from the benchmark.

The average time and average delay for all the protocols are given in Table 10-1. Each protocol has its average delay measured against their respective 'no conflict' cases. The 'no conflict' cases are unique for each protocol because of their different control methods. The average added delay is measured against the average estimated delay, which is defined in Section 10-3-2. The average estimated delay for the tested cases is found to be 0.4089 s using a sampling-based method.

Compared with AMP-IP, OA-ADMM MPC is found to have a 79.95 %, 47.95 %, and 43.74 %

decrease in average delay for the low, medium, and high fidelity cases respectively. The percentage decrease in average added delay regarding the low, medium, and high fidelity cases for AMP-IP are 89.04 %, 65.25 %, and 61.32 % respectively.

Compared with TDCR, OA-ADMM MPC is found to have a 78.21 %, 38.61 %, and 31.11 % decrease in average delay for the low, medium, and high fidelity cases respectively. The percentage decrease in average added delay regarding the low, medium, and high fidelity cases for AMP-IP are 87.98 %, 56.18 %, and 47.93 % respectively.

	L,L	L,F	L,R	F,L	F,F	F,R	R,L	R,F	R,R
Est. Delay L	0.80	0.80	0.00	0.00	0.72	0.83	0.80	0.75	0.00
Est. Delay F	0.75	0.66	0.00	0.72	0.00	0.00	0.80	0.66	0.96
Est. Delay R	0.00	0.96	0.00	0.83	0.00	0.00	0.00	0.00	0.00
AMP-IP (h) L	1.66	1.55	0.86	1.15	1.40	1.33	1.66	1.92	0.00
AMP-IP (h) F	1.92	0.59	0.00	1.40	0.00	0.00	1.55	0.59	1.01
AMP-IP (h) R	0.00	1.01	0.00	1.33	0.00	0.00	0.86	0.00	0.00
TDCR (h) L	1.26	1.60	1.22	1.31	1.45	0.35	1.26	1.21	0.00
TDCR (h) F	1.21	0.56	0.00	1.45	0.00	0.00	1.60	0.56	0.20
TDCR (h) R	0.00	0.20	0.00	0.35	0.00	0.00	1.22	0.00	0.00
OA-ADMM L	0.71	0.55	0.00	0.02	0.15	0.00	0.71	1.91	0.00
OA-ADMM F	1.91	0.69	0.00	0.15	0.00	0.00	0.55	0.69	0.02
OA-ADMM R	0.00	0.02	0.01	0.00	0.01	0.00	0.00	0.00	0.01

Table 10-2: Detailed table of AMP-IP, TDCR, and OA-ADMM results for the Carla simulations using the MCR benchmark, showing the delays for each possible combination depicted in Figure 10-5 excluding the follower vehicle cases. The protocols along with their respective action is listed in the leftmost column, e.g. "AMP-IP (h) L" refers to a vehicle (A) making a left turn using the AMP-IP algorithm with a high fidelity grid map. The column names refer to the actions made by the other vehicle, e.g. "L,F" refers to a vehicle (B) arriving from the left, going forward (relative to B). This delay is then referred to as the delay for "AMP-IP (h) L/L,F", with the conflict case being referred to as solely "L/L,F". This specific case is shown in the third row and third column of Figure 10-5, along with the cases "L/L,R" and "L/L,L".

A detailed overview of the delays for the protocols is given in Table 10-2, where the delay for each individual conflict case is given separately. Note that only the high fidelity cases are shown, the delays for the lower fidelity cases are deemed less relevant for the detailed comparison as they are always higher than the delays for the high fidelity cases.

Chapter 11

Discussion

This chapter contains the discussion for the result of OA-ADMM, AMP-IP and TDCR performed using the Modular Conflict Resolution (MCR) benchmark in the CARLA simulations. The chapter is separated in the following sections: Section 11-1, where the benchmarking tool is discussed, Section 11-2, which analyzes the application of OA-ADMM MPC to a more realistic vehicle model, Section 11-3 which compared OA-ADMM MPC against the AMP-IP method, Section 11-4, which covers the differences in execution and results between OA-ADMM MPC and TDCR, finally Section 11-5, summarizes the most important points from the discussion.

11-1 Benchmarking Tool Discussion

The main goal of the benchmark is to provide a consistent and insightful results for the comparison of decentralized conflict resolution protocols. With a focus on the efficiency of the conflict resolution, which is measured using the added delay metric. The discussion on the benchmark is covered in three sections: firstly the metric and its advantages is discussed in Section 11-1-1; secondly, the implementation of the benchmark and its effects on the results is discussed in Section 11-1-2; thirdly, the design of the OA-ADMM functions and how to tune them is discussed in Section 11-1-2-1.

11-1-1 Metric

The purpose of the estimated delay metric is to provide a direct comparison point for the measured delays, instead of relying on the comparison between protocols. This allows protocols from various works to be compared directly, even when the works do not compare the same protocols. Without the estimated delay metric this would not be possible, as a metric such as the delay compared with the no conflict case is highly dependent on the implementation, such as the environment, the reference velocities, etc. Since the definition of the estimated delay is

given as a function of two trajectories, an analytical solution can be found for all cases which involve two trajectories.

Another benefit of the estimated delay is that it is no longer necessary to perform simulations of a centralized solution to compare against, when comparing between decentralized solutions. The main case for decentralized protocols is its use in environments where a centralized solution is not available, comparing the results against the centralized solution, whose results are once again dependent on the implementation of the centralized solution, does not serve as a consistent and insightful metric for decentralized protocols.

11-1-2 Implementation

The implementation of the benchmarking tool in CARLA has vehicles spawning at the intersection with equal distance to the intersection center. The vehicles are assigned a random reference velocity, which results in different arrival times at the intersection and different contact locations between two trajectories. For that reason, it is possible for vehicles to not have a conflict in trajectories in some cases. For example, the conflict case depicted in the top left intersection in Figure 10-5, shows conflicts between a vehicle traveling forward and a vehicle approaching from its right. When the magenta vehicle is traveling significantly faster than the blue vehicle, it is possible that no conflict occurs between the two vehicles: the magenta vehicle will have passed the conflicting part of the path by the time that the blue vehicle arrives there.

Another part of the benchmark implementation that affects the obtained results is the control methods of the various protocols. As the AMP-IP protocol suggests a simple algorithm where vehicles simply stop before a certain area, and otherwise travel at their reference velocity, a simple PID controller for its velocity is sufficient. The TDCR protocol on the other hand requires a constrained motion planner to ensure that the time slot constraints are not violated, this is implemented through a Model Predictive Control velocity planner, which implements the time slot constraints as a displacement constraint on the planned path. The OA-ADMM MPC problem is formulated as a general motion planning problem, which results in a Model Predictive Controller that plans a trajectory which can deviate from the reference path. The ability to adjust the trajectory of the vehicle can result in a difference in delays, as certain conflict can be avoided by deviating slightly from the planned paths. One example of this is the simultaneous left turns from opposing lanes, depicted in the third row, second column in Figure 10-5. The default planned trajectories given by the A* planner in CARLA has a minor conflict between these two trajectories, this can only be resolved without invoking a passing order by adjusting the trajectories. Due to the difference in motion planning and control solutions, the delays are compared with the no conflict case for their respective motion planning and control solutions.

The chosen environment of the benchmark is also has a significant effect on the obtained results. Being a small intersection with narrow lanes provides an environment where deviation in the trajectories can be difficult to realize. This can result in OA-ADMM MPC obtaining results which do not properly show the benefit of the ability to adjust trajectories. However, if the benefit can be shown, it is likely that the benefit will be more significant when the environment is more spacious.

11-1-2-1 OA-ADMM functions design and tuning

Given that OA-ADMM introduces two new functions it is important to discuss the approach used to tune the functions and parameters. Starting with the adaptation function ϕ, the intuition behind the designed functions Equation (10-14) and Equation (10-15) is to penalize the deviation from the collision free copies $\boldsymbol{z}$ as the possibility of collision increases. The goal therefore is to implement a metric which is represents some form of collision probability. The easiest solution is to simple implement the ratio between a desired distance between vehicles, and the actual distance. This function increases as the distance between the samples decreases, resulting in the ρ increasing for the constraint between those two samples. To give a short example of how this works in practice we take the following $\boldsymbol{x}$ trajectories:

$$\begin{aligned} \boldsymbol{x}_i &= \Big[2, 0, 3, 0\Big]^\top \\ \boldsymbol{x}_j &= \Big[3, 2, 3, 1\Big]^\top, \end{aligned} \tag{11-1}$$

where the $\boldsymbol{x}$ vectors include the x and y coordinates for the first two samples. Suppose that the minimum distance constraint D_{min} is 2 meters, a possible collision free solution $\boldsymbol{z}$ could be

$$\begin{aligned} \boldsymbol{z}_i &= \Big[2, 0, 3, -0.5\Big]^\top \\ \boldsymbol{z}_j &= \Big[3, 2, 3, 1.5\Big]^\top. \end{aligned} \tag{11-2}$$

Using the proposed design approach a possible ϕ can be

$$\phi(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{D_{min}}{dist(\boldsymbol{x}_i, \boldsymbol{x}_j)}, \tag{11-3}$$

which results in the following penalty vector:

$$\boldsymbol{\rho} = \Big[\tfrac{2}{\sqrt{5}}, \tfrac{2}{\sqrt{5}}, 2, 2\Big]^\top. \tag{11-4}$$

The simple adaptation function increases the penalty for deviation from the collision free trajectories by increasing the step size for the λ update and by increasing the squared L2-Norm penalty for the x update.

The implementation of OA-ADMM MPC in CARLA also utilizes a parameter a to scale the distance ratio along with a lower and upper bound for the penalty vector. The parameter a can be used to decrease the value of the penalty vector for ratios < 1 and increase the value of the penalty vector for ratios > 1. The lower bound $\boldsymbol{\phi}^{min}$ is implemented to prevent dual infeasibility in the solver due to ill-conditioned problems. The minimum value might not be necessary depending on the used solver, however if necessary it can be tuned by increasing its value until the solver no longer results in dual infeasibility or maximum iterations reached. The upper bound $\boldsymbol{\phi}^{max}$ can be used to prevent the time-dependent dynamics from causing oscillations in the penalty parameter. In the case that the value of $\boldsymbol{\rho}$ is allowed to increase by a large amount, the result is that λ will be increase significantly the next timestep. As a result, the next planned trajectory will overcompensate and over prioritize the collision-free trajectory compared with the reference trajectory. This in turn results in a very small $\boldsymbol{\rho}$ which causes in the next planned trajectory to over-prioritize the reference trajectory rather than the collision free trajectory. This is essentially what happens when the step-size of gradient descent is too large. The upper bound $\boldsymbol{\phi}^{max}$ should be tuned by evaluating the oscillations in $\boldsymbol{\rho}$ and $\boldsymbol{\lambda}$, if the oscillations are too large, the bound should be lowered.

11-2 OA-ADMM MPC Results Discussion

The OA-ADMM MPC strategy has been proposed and evaluated in Chapter 9, showing the ability for OA-ADMM MPC to achieve priority based conflict resolution in a decentralized fashion with holonomic vehicles. To contribute to these results the strategy is applied to a more difficult problem in terms of vehicle dynamics. This is done by applying OA-ADMM to simulated vehicles in CARLA, where it is compared against the AMP-IP and TDCR decentralized protocols. These results are focused on the conflict resolution efficiency, and therefore do not focus on deadlocks. However, given that we have shown that OA-ADMM MPC is capable of following a priority order, a simple priority based deadlock resolution method can be applied. Is it also possible that a more directly integrated approach to resolving deadlocks is possible with OA-ADMM MPC, however that is beyond the scope of this thesis.

The results from the simulations in Section 10-5 show that OA-ADMM MPC can resolve conflicts without primal constraint violation for non-holonomic vehicles with nonlinear dynamics. The use of adaptive-MPC, which linearizes the MPC state matrices at each control input, reduces the prediction error. However, the prediction error for longer horizons is still significant, especially when vehicles are to make a turn, where the prediction error can induce issues in the conflict resolution algorithm. Given that OA-ADMM MPC utilizes the predicted trajectory from MPC to adjust the trajectories to be collision free, it is important that the predicted trajectory represents the actual planned trajectory to a certain degree. For example, a vehicle intending to turn left may not consistently have a predicted trajectory going straight or turning right, as this will lock the vehicle into resolving the conflict for a trajectory which it is not intending to follow. Whilst there is no clear requirement for the prediction error, it is always desirable to reduce it as much as possible. The use of Nonlinear model predictive control can be a potential solution to this problem. Despite the importance of the prediction error, the results show that OA-ADMM MPC is able to function with a simple linearized kinematic bicycle model using adaptive MPC. This shows that OA-ADMM does not require (close to) perfect prediction to achieve convergence. If it is known to the user that the prediction error will be significant beyond a certain horizon length it is advisable to reduce the horizon or to take the errors into account in the design of the similarity function.

11-3 OA-ADMM MPC vs AMP-IP Results Discussion

The results from Section 10-5 show a clear performance increase for OA-ADMM compared with AMP-IP in terms of travel time, delay, and added delays. AMP-IP is a heuristic conflict resolution method with discrete conflict zones in form of cells in a grid, whereas OA-ADMM is an MPC-based conflict resolution method, which samples the trajectory on the $\mathbb{R}^2$ plane. The MPC collision avoidance constraint between two samples is comparable to a grid with infinite resolution. The only prior knowledge required by OA-ADMM MPC is the relative position of the two vehicles, which can be easily obtained even without communication, and the predicted trajectories, which is also required for AMP-IP. On the other hand, for AMP-IP to obtain a resolution similar to that of OA-ADMM MPC, a very high resolution map is needed of the intersection, which is not reasonable to expect to always be available. In the case that a map is made with the on-board sensors, a solution must be provided to prevent disagreements in the map and the cell division of the map.

Another significant difference between OA-ADMM MPC and AMP-IP is the ability for OA-ADMM to deviate from the reference trajectory, whereas AMP-IP only can affect the velocity of the vehicle. If the vehicles plan their trajectory in a way that causes conflicts with many other trajectories they cannot adjust their trajectories to avoid these conflict, and are instead forced to simply wait their turn. Meanwhile, OA-ADMM MPC is not constrained to only adjust its velocity as utilizes MPC for its trajectory optimization. This enables these conflicts to be resolved without the need for a passing order, enabling faster conflict resolution than AMP-IP can obtain, even if it has a map of infinite resolution.

The advantages of the MPC-based OA-ADMM approach is easily visible in the conflict resolution efficiency results found in Section 10-5. In the case that AMP-IP has a low fidelity map, the average delay is around 4 seconds, which is about five times as high as the average delay of OA-ADMM, which is around 0.8 seconds. The main reason for this is that the low fidelity map enables only one vehicle to enter the entire intersection at once, similar to how a 4-way stop sign functions. This can be seen as the worst case scenario for AMP-IP, with the exception of a disagreement in maps leading to collisions. When AMP-IP upgrades ot a medium fidelity map the delay reduces to around 1.5 seconds, which is further reduces to around 1.4 seconds with a high fidelity map. The substantial reduction in delay between the low fidelity map and medium fidelity map is entirely the result of the improved resolution of collision avoidance. The high fidelity is however close to the optimal performance of AMP-IP, as the delay decrease between the medium fidelity and the high fidelity map is only 7.45 %, compared with the decrease of 61.47 % between the low and medium fidelity maps. The remaining difference between AMP-IP and OA-ADMM is likely caused by the ability to deviate from the reference trajectory and the use of a predictive control method, compared with the reactive velocity control method employed by AMP-IP.

11-4 OA-ADMM MPC vs TDCR Results Discussion

TDCR can essentially be seen as AMP-IP with a predictive control method, which should reduce delays by allowing vehicles to adjust their velocities to enter a conflict zone the moment the previous vehicle is planned to leave it. AMP-IP, conversely, only allows vehicles to start accelerating the moment the other vehicle has already left the conflict zone. Comparing the delays for TDCR with the delays from AMP-IP, TDCR decreases the average delay by 7.9 % for the low fidelity case, 15.2 % for the medium fidelity case, and 18.4% for the high fidelity case. When looking at the absolute decreases, the decrease is 0.32 s, 0.235 s, and 0.262 s for the low, medium, and high fidelity cases respectively. This seems to suggest that the absolute performance improvement caused by the predictive control method is around 0.25 seconds in terms of the average delay.

Assuming that the high fidelity map achieves most of the improvements in terms of collision avoidance resolution, it is likely that the reduction in delay between TDCR and OA-ADMM is mainly caused by the ability to deviate from the reference trajectories. To support this claim we can investigate the delays found in the detailed overview of conflict cases given in Table 10-2. The row containing the conflict cases for a right turning vehicle show that there is essentially no delay for OA-ADMM MPC, with the variance likely due to numerical errors in the simulation and/or optimization. For the case of TDCR on the other hand, some significant delays are found for the R/L,F case, the R/F,L case and the R/R,L case. The absence of these

delays in OA-ADMM MPC can be attributed to the deviation from the reference trajectory, as this is the major difference between the two approaches. The estimated delays are also higher than the delays obtained by TDCR and OA-ADMM for the right turning vehicles, this is likely due to the assumption that vehicles arrive at the conflict point at the same time for the estimated delay, whereas the benchmark has both vehicles arriving at the intersection at the same time.

Comparing the delays for the forward traveling vehicle between OA-ADMM and TDCR also shows that OA-ADMM in general has lower delays, with the exception for the F/L,L case and the F/L,F case. In these cases OA-ADMM obtained a delay of 1.91 seconds and 0.69 seconds respectively, whereas TDCR obtained delays of 1.21 seconds and 0.56 seconds respectively. It is likely that the optimal solution is for one vehicle to adjust its velocity and simply wait its turn, which TDCR is effective at solving. However, the current implementation of OA-ADMM MPC has no way to enforce certain vehicles to follow a passing order as this is not easily defined in the trajectory MPC problem. It is possible to utilize a hybrid between the TDCR approach and the OA-ADMM approach, in cases where it is known that the optimal solution involves awaiting a passing order.

When comparing the results for the left turning vehicle, TDCR seems to consistently have delays of above 1 second, except for two cases: L/F,R and L/R,R. OA-ADMM on the other hand has only one case with a delay above 1 second: the L/R,F case. Note that the L/R,F case is identical to the F/L,L case mentioned in the previous paragraph[28]. Left turns using a passing order based solution results in significant delays for the system. These delays, however, can be reduced significantly by allowing vehicles to adjust their trajectories. The cases where this seems the most applicable are: L/L,F, L/L,R, L/F,L, L/F,F: these cases all show a decrease in delay of over 1 second for OA-ADMM compared with TDCR.

11-5 Summary

OA-ADMM MPC generally achieves shorter delays than competing standards like AMP-IP and TDCR. OA-ADMM MPC utilizes MPC based conflict resolution whereas AMP-IP and TDCR utilize a reservation based conflict resolution method. The differences in delays between OA-ADMM and AMP-IP suggests that the ability to deviate from trajectories, along with the MPC based controller, is able to reduce delays significantly, whilst having fewer requirements in terms of prior knowledge. When analyzing the delays for specific cases between OA-ADMM and TDCR, the results suggest that the ability to deviate from the reference trajectories is a major contributor to increase conflict resolution efficiency.

However, the results also indicate that for certain cases, the OA-ADMM MPC method results in higher delays than TDCR, suggesting that in certain cases enforcing a passing order can be more beneficial than the MPC based approach utilized by OA-ADMM MPC. It could be possible to utilize a hybrid of these methods to achieve the best of both worlds.

It is also possible that this issue is the result of the prediction error caused by the linearized kinematic bicycle model used by the adaptive MPC implementation for OA-ADMM MPC. If a nonlinear MPC model is used instead, it is possible that this behavior is no longer found.

[28] L/R,F and F/L,L both refer to the same situation, with the difference being in which vehicle is taken as the observer.

Chapter 12

Conclusion

This chapter includes the conclusion of the Online Adaptive Alternating Direction Method of Multipliers (OA-ADMM), along with its application to decentralized conflict resolution. The conclusion is separated into three sections, first a general summary of the thesis is given in Section 12-1; second, the limitations of OA-ADMM and OA-ADMM MPC is given in Section 12-2; finally, potential future work to solve these limitations is given in Section 12-3.

12-1 Summary

This thesis started by introducing the decentralized conflict resolution problem along with the problem statement. The necessary background of the Alternating Direction Method of Multipliers was explored extensively, including the proof of convergence for conventional ADMM. Afterwards, the application of an adaptive penalty parameter was explored, along with the application of ADMM to online/real-time problems. On that foundation, the novel Online Adaptive-ADMM method was proposed, which introduces two novel functions: the adaptation function ϕ and the similarity function μ. The adaptation function forms the adaptive part of OA-ADMM, allowing the framework to adjust the priority of certain constraint depending on the systems needs. The similarity function is what allows the system to function in a time-dependent environment, where the optimal value of the Lagrange multiplier will shift over time. This method is extensively proposed and analyzed, with convergence proven for time-independent systems. Arguments for online convergence are given and analyzed.

The OA-ADMM method is then combined with MPC to achieve decentralized conflict resolution. This method applies OA-ADMM to the centralized MPC based conflict resolution, decomposing the problem into two parallelizable optimization problems. This method is shown to be able to resolve a conflict using a priority based method for holonomic vehicles in a Matlab simulation. The method was then compared against AMP-IP and TDCR in CARLA simulations using the Modular Conflict Resolution (MCR) benchmarking tool. These simulations were focused on the applicability of OA-ADMM MPC to the more difficult problem involving non-holonomic vehicles and nonlinear dynamics. The benchmarking tool is focused

on the travel delays caused by the conflict resolution, providing a metric for the estimated delay for the tested conflict cases.

The results indicate that OA-ADMM is able to achieve lower delays for almost all conflict cases, with three potential reasons. The first being the essentially infinite resolution achieved by the sample based collision avoidance constraints in MPC compared with the grid based collision avoidance methods utilized by AMP-IP and TDCR. The second possible reason is only relevant to AMP-IP, as both OA-ADMM and TDCR use predictive controllers instead of reactive controllers. The last possible reason is the ability for OA-ADMM to deviate from the reference trajectories, whereas AMP-IP and TDCR can only vary their velocities. These three reasons add up to OA-ADMM performing better, whilst having fewer requirements in terms of prior knowledge.

12-2 Limitations

The major limitations of OA-ADMM is related to the novelty of the approach. Whilst the convergence is proven for the time-independent case, the method is designed for time-dependent systems. The inability to prove convergence for time-dependent cases is however not unique to OA-ADMM, as this is a difficult problem to solve for all real-time optimization methods. The adaptation and similarity functions of OA-ADMM however do allow the user to increase the robustness of OA-ADMM, if the functions are designed properly.

Whilst a few examples are provided for the two functions, along with the analysis of their effects, the exact design of these functions are not fully explored. When the method is applied to a physical system an adaptation function related to the physical safety can be a decent approach, however the mathematical aspects of the function design require more research.

The OA-ADMM MPC method proposed in this work is able to efficiently resolve conflicts using a priority protocol. However, a priority based method still relies on a passing order based deadlock detection method, which can lead to efficiency reductions in OA-ADMM MPC. Currently, no deadlock detection or resolution method exists which is closely integrated with an MPC-based conflict resolution or multi-agent motion planning and collision avoidance approach.

As OA-ADMM MPC utilizes a predictive model, prediction errors affect the effectiveness and robustness of the system. The simulations have shown that OA-ADMM MPC is able to function even with significant prediction errors caused by the linearized bicycle model. However, the exact bounds necessary to have good results are not known.

12-3 Future Work

Given that OA-ADMM is a novel method, a lot of work can be done to further explore the proposed adaptation function and similarity function. It is important to set more exact mathematical requirements and provide more mathematically founded design strategies to allow the application of OA-ADMM to as of yet unexplored problems.

For OA-ADMM MPC it is important to consider to possibility of a more closely integrated deadlock detection and resolution method. It is possible that this can be integrated in the

adaptation function as the adaptation function is what enables OA-ADMM MPC to have priority based conflict resolution. It is perhaps possible to adapt the values of ρ such when a potential deadlock is deemed more likely.

Better guarantees and arguments for online convergence of OA-ADMM is needed. Whilst the method is shown to function through simulations, a challenge for real-time optimization in general is the lack of mathematical convergence proof. If this can be provided the direct application of OA-ADMM to more safety critical systems can be considered. As of now, OA-ADMM should be accompanied by a low level controller with safety guarantees for safety critical applications.

Bibliography

[1] D. Schrank, B. Eisele, T. Lomax, and J. Bak, "2015 Urban Mobility Scorecard," Aug. 2015.

[2] J. I. Levy, J. J. Buonocore, and K. von Stackelberg, "Evaluation of the public health impacts of traffic congestion: A health risk assessment," *Environmental Health*, vol. 9, no. 1, p. 65, Oct. 2010.

[3] M. Barth and K. Boriboonsomsin, "Real-World Carbon Dioxide Impacts of Traffic Congestion," *Transportation Research Record*, vol. 2058, no. 1, pp. 163–171, Jan. 2008.

[4] J. Walker, "The Self-Driving Car Timeline – Predictions from the Top 11 Global Automakers," https://emerj.com/ai-adoption-timelines/self-driving-car-timeline-themselves-top-11-automakers/.

[5] Y. Rahmati and A. Talebpour, "Towards a collaborative connected, automated driving environment: A game theory based decision framework for unprotected left turn maneuvers," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2017, pp. 1316–1321.

[6] J. A. Khan, L. Wang, E. Jacobs, A. Talebian, S. Mishra, C. A. Santo, M. Golias, and C. Astorne-Figari, "Smart Cities Connected and Autonomous Vehicles Readiness Index," in *ACM SCC*, Portland, OR, United States, 2019.

[7] Y. Liu, F. Dion, and S. Biswas, "Dedicated Short-Range Wireless Communications for Intelligent Transportation System Applications: State of the Art," *Transportation Research Record*, vol. 1910, no. 1, pp. 29–37, Jan. 2005.

[8] L. Xiao and F. Gao, "A comprehensive review of the development of adaptive cruise control systems," *Vehicle System Dynamics*, vol. 48, no. 10, pp. 1167–1192, Oct. 2010.

[9] Y. W. Sugimoto and C. W. Sauer, "Effectiveness Estimation Method for Advanced Driver Assistance System and Its Application to Collision Mitigation Brake System," /paper/Effectiveness-Estimation-Method-for-Advanced-Driver-Sugimoto-Sauer/5fa2d6979465924295584415 1956d5664e92e920, 2005.

[10] D. Bevly, X. Cao, M. Gordon, G. Ozbilgin, D. Kari, B. Nelson, J. Woodruff, M. Barth, C. Murray, A. Kurt, K. Redmill, and U. Ozguner, "Lane Change and Merge Maneuvers for Connected and Automated Vehicles: A Survey," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 105–120, Mar. 2016.

[11] K. Dresner and P. Stone, "Multiagent Traffic Management: A Reservation-Based Intersection Control Mechanism." in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2004*, vol. 2, Jan. 2004, pp. 530–537.

[12] ——, "A Multiagent Approach to Autonomous Intersection Management," *Journal of Artificial Intelligence Research*, vol. 31, pp. 591–656, Mar. 2008.

[13] Minjie Zhu, Xu Li, Hongyu Huang, Linghe Kong, Minglu Li, and Min-You Wu, "LICP: A look-ahead intersection control policy with intelligent vehicles," in *2009 IEEE 6th International Conference on Mobile Adhoc and Sensor Systems*, Oct. 2009, pp. 633–638.

[14] N. Murgovski, G. R. de Campos, and J. Sjöberg, "Convex modeling of conflict resolution at traffic intersections," in *2015 54th IEEE Conference on Decision and Control (CDC)*, Dec. 2015, pp. 4708–4713.

[15] L. Riegger, M. Carlander, N. Lidander, N. Murgovski, and J. Sjöberg, "Centralized MPC for autonomous intersection crossing," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Nov. 2016, pp. 1372–1377.

[16] R. Naumann, R. Rasche, J. Tacken, and C. Tahedi, "Validation and simulation of a decentralized intersection collision avoidance algorithm," in *Proceedings of Conference on Intelligent Transportation Systems*, Nov. 1997, pp. 818–823.

[17] L. Makarem and D. Gillet, "Decentralized Coordination of Autonomous Vehicles at intersections," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 13 046–13 051, Jan. 2011.

[18] G. R. de Campos, P. Falcone, and J. Sjöberg, "Autonomous cooperative driving: A velocity-based negotiation approach for intersection crossing," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, Oct. 2013, pp. 1456–1461.

[19] S. R. Azimi, G. Bhatia, R. R. Rajkumar, and P. Mudalige, "Vehicular Networks for Collision Avoidance at Intersections," *SAE International Journal of Passenger Cars - Mechanical Systems*, vol. 4, no. 1, pp. 406–416, Apr. 2011.

[20] R. Azimi, G. Bhatia, R. Rajkumar, and P. Mudalige, "Intersection Management using Vehicular Networks," in *SAE 2012 World Congress & Exhibition*, Apr. 2012, pp. 2012–01–0292.

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

[22] C. Liu, C. Lin, S. Shiraishi, and M. Tomizuka, "Distributed Conflict Resolution for Connected Autonomous Vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 1, pp. 18–29, Mar. 2018.

[23] R. Van Parys and G. Pipeleers, "Online distributed motion planning for multi-vehicle systems," in *2016 European Control Conference (ECC)*, Jun. 2016, pp. 1580–1585.

[24] ——, "Distributed model predictive formation control with inter-vehicle collision avoidance," in *2017 11th Asian Control Conference (ASCC)*, Dec. 2017, pp. 2399–2404.

[25] H. Zheng, R. R. Negenborn, and G. Lodewijks, "Fast ADMM for Distributed Model Predictive Control of Cooperative Waterborne AGVs," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 4, pp. 1406–1413, Jul. 2017.

[26] F. Rey, Z. Pan, A. Hauswirth, and J. Lygeros, "Fully Decentralized ADMM for Coordination and Collision Avoidance," in *2018 European Control Conference (ECC)*, Jun. 2018, pp. 825–830.

[27] L. Chen, H. Hopman, and R. R. Negenborn, "Distributed model predictive control for vessel train formations of cooperative multi-vessel systems," *Transportation Research Part C: Emerging Technologies*, vol. 92, pp. 101–118, Jul. 2018.

[28] B. S. He, H. Yang, and S. L. Wang, "Alternating Direction Method with Self-Adaptive Penalty Parameters for Monotone Variational Inequalities," *Journal of Optimization Theory and Applications*, vol. 106, no. 2, pp. 337–356, Aug. 2000.

[29] B. E. Jackson, T. A. Howell, K. Shah, M. Schwager, and Z. Manchester, "Scalable Cooperative Transport of Cable-Suspended Loads with UAVs using Distributed Trajectory Optimization," 2019.

[30] R. Firoozi, L. Ferranti, X. Zhang, S. Nejadnik, and F. Borrelli, "A Distributed Multi-Robot Coordination Algorithm for Navigation in Tight Environments," *arXiv:2006.11492 [cs]*, Jun. 2020.

[31] Z. Xu, M. A. T. Figueiredo, and T. Goldstein, "Adaptive ADMM with Spectral Penalty Parameter Selection," *arXiv:1605.07246 [cs]*, Jul. 2017.

[32] Z. Xu, M. A. T. Figueiredo, X. Yuan, C. Studer, and T. Goldstein, "Adaptive Relaxed ADMM: Convergence Theory and Practical Implementation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7389–7398.

[33] A. Kelly, "Essential Kinematics for Autonomous Vehicles," CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, Tech. Rep. CMU-RI-TR-94-14, May 1994.

[34] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2015, pp. 1094–1099.

[35] *ISO Standard for Road vehicles – Vehicle Identification Number (VIN) – Content and Structure*, ISO 3779, 2009.

[36] *ISO Standard for Road vehicles – World Manufacturer Identifier (WMI) Code*, ISO 3780, 2009.

[37] J. Kaths, E. Papapanagiotou, and F. Busch, "Traffic Signals in Connected Vehicle Environments: Chances, Challenges and Examples for Future Traffic Signal Control," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, Sep. 2015, pp. 125–130.

[38] C. Priemer and B. Friedrich, "A decentralized adaptive traffic signal control using V2I communication data," in *2009 12th International IEEE Conference on Intelligent Transportation Systems*, Oct. 2009, pp. 1–6.

[39] H. Rakha and R. K. Kamalanathsharma, "Eco-driving at signalized intersections using V2I communication," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Oct. 2011, pp. 341–346.

[40] F. Ye, M. Adams, and S. Roy, "V2V Wireless Communication Protocol for Rear-End Collision Avoidance on Highways," in *ICC Workshops - 2008 IEEE International Conference on Communications Workshops*, May 2008, pp. 375–379.

[41] P. Gomes, C. Olaverri-Monreal, and M. Ferreira, "Making Vehicles Transparent Through V2V Video Streaming," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, pp. 930–938, Jun. 2012.

[42] "FCC Allocates Spectrum 5.9 GHz Range for Intelligent Transportation Systems Uses," https://transition.fcc.gov/Bureaus/Engineering_Technology/News_Releases/1999/nret9006.html.

[43] D. Jiang and L. Delgrossi, "IEEE 802.11p: Towards an International Standard for Wireless Access in Vehicular Environments," in *VTC Spring 2008 - IEEE Vehicular Technology Conference*, May 2008, pp. 2036–2040.

[44] *IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments*, IEEE 802.11p, 2010.

[45] S. Gräfling, P. Mähönen, and J. Riihijärvi, "Performance evaluation of IEEE 1609 WAVE and IEEE 802.11p for vehicular communications," in *2010 Second International Conference on Ubiquitous and Future Networks (ICUFN)*, Jun. 2010, pp. 344–348.

[46] X. Wang, S. Mao, and M. X. Gong, "An Overview of 3GPP Cellular Vehicle-to-Everything Standards," *GetMobile: Mobile Computing and Communications*, vol. 21, no. 3, pp. 19–25, Nov. 2017.

[47] V. Vukadinovic, K. Bakowski, P. Marsch, I. D. Garcia, H. Xu, M. Sybis, P. Sroka, K. Wesolowski, D. Lister, and I. Thibault, "3GPP C-V2X and IEEE 802.11p for Vehicle-to-Vehicle communications in highway platooning scenarios," *Ad Hoc Networks*, vol. 74, pp. 17–29, May 2018.

[48] *Dedicated short range communications (DSRC) message set dictionary*, SAE Standard J2735, 2009.

[49] *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service*, ETSI EN 302 637-2 V1.4.1, 2019.

[50] E. Young Hyun, I. Kim, H. Yoo, S. Cho, and B. Jeon, "Design of the emergency messages integration for vehicle to everything (V2X) communications of connected cars communications of connected carsV2X)," *International Journal of Engineering and Technology(UAE)*, vol. 7, pp. 12–14, Aug. 2018.

[51] "European Commission - PRESS RELEASES - Press release - Road Safety: New rules clear way for clean, connected and automated mobility on EU roads," https://europa.eu/rapid/press-release_MEMO-19-1649_en.htm.

[52] F. Y. Chee, "EU opens road to 5G connected cars in boost to BMW, Qualcomm," https://www.reuters.com/article/us-eu-autos-tech-idUSKCN1TZ11F, Jul. 2019.

[53] J. Horwitz, "Why Europe just boosted 5G over Wi-Fi for connected cars," https://venturebeat.com/2019/07/05/why-europe-just-boosted-5g-over-wi-fi-for-connected-cars/, Jul. 2019.

[54] A. Bazzi, B. M. Masini, A. Zanella, and I. Thibault, "On the Performance of IEEE 802.11p and LTE-V2V for the Cooperative Awareness of Connected Vehicles," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 11, pp. 10 419–10 432, Nov. 2017.

[55] H. A. Rakha, I. Zohdy, and R. K. Kamalanathsharma, "Agent-Based Game Theory Modeling for Driverless Vehicles at Intersections," Feb. 2013.

[56] M. Elhenawy, A. A. Elbery, A. A. Hassan, and H. A. Rakha, "An Intersection Game-Theory-Based Traffic Control Algorithm in a Connected Vehicle Environment," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, Sep. 2015, pp. 343–347.

[57] Z. Han, *Game Theory in Wireless and Communication Networks: Theory, Models, and Applications*. Cambridge, UK ;: Cambridge University Press, 2012.

[58] M. Kloock, P. Scheffe, S. Marquardt, J. Maczijewski, B. Alrifaee, and S. Kowalewski, "Distributed Model Predictive Intersection Control of Multiple Vehicles," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Oct. 2019, pp. 1735–1740.

[59] Guangquan Lu, Lumiao Li, Yunpeng Wang, Ran Zhang, Zewen Bao, and Haichong Chen, "A rule based control algorithm of connected vehicles in uncontrolled intersection," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Oct. 2014, pp. 115–120.

[60] N. Li, I. Kolmanovsky, A. Girard, and Y. Yildiz, "Game Theoretic Modeling of Vehicle Interactions at Unsignalized Intersections and Application to Autonomous Vehicle Control," Jun. 2018.

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

[62] H. Rewald and O. Stursberg, "Cooperation of autonomous vehicles using a hierarchy of auction-based and model-predictive control," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2016, pp. 1078–1084.

[63] L. Makarem and D. Gillet, "Fluent coordination of autonomous vehicles at intersections," in *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct. 2012, pp. 2557–2562.

[64] R. C. Holt, "Some Deadlock Properties of Computer Systems," in *Proceedings of the Third ACM Symposium on Operating Systems Principles*, ser. SOSP '71. New York, NY, USA: ACM, 1971, pp. 64–71.

[65] R. Obermarck, "Distributed Deadlock Detection Algorithm," *ACM Trans. Database Syst*, vol. 7, pp. 187–208, 1982.

[66] K. M. Chandy and J. Misra, "A Distributed Algorithm for Detecting Resource Deadlocks in Distributed Systems," in *Proceedings of the First ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, ser. PODC '82. New York, NY, USA: ACM, 1982, pp. 157–164.

[67] K. M. Chandy, J. Misra, and L. M. Haas, "Distributed Deadlock Detection," *ACM Trans. Comput. Syst.*, vol. 1, no. 2, pp. 144–156, May 1983.

[68] A. K. Elmagarmid, "A Survey of Distributed Deadlock Detection Algorithms," *SIGMOD Rec.*, vol. 15, no. 3, pp. 37–45, Sep. 1986.

[69] M. P. Fanti, "Event-based controller to avoid deadlock and collisions in zone-control AGVS," *International Journal of Production Research*, vol. 40, no. 6, pp. 1453–1478, Jan. 2002.

[70] K. Im, K. Kim, Y. Moon, T. Park, and S. Lee, "The deadlock detection and resolution method for a unified transport system," *International Journal of Production Research*, vol. 48, no. 15, pp. 4423–4435, Aug. 2010.

[71] F. Perronnet, A. Abbas-Turki, and A. El Moudni, "Vehicle routing through deadlock-free policy for cooperative traffic control in a network of intersections: Reservation and congestion," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Oct. 2014, pp. 2233–2238.

[72] Y.-T. Lin, H. Hsu, S.-C. Lin, C.-W. Lin, I. H.-R. Jiang, and C. Liu, "Graph-Based Modeling, Scheduling, and Verification for Intersection Management of Intelligent Vehicles," *ACM Trans. Embed. Comput. Syst.*, vol. 18, no. 5s, pp. 95:1–95:21, Oct. 2019.

[73] Shigang Chen, Yi Deng, P. Attie, and Wei Sun, "Optimal deadlock detection in distributed systems based on locally constructed wait-for graphs," in *Proceedings of 16th International Conference on Distributed Computing Systems*, May 1996, pp. 613–619.

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

[75] S. Srinivasan and R. Rajaram, "A Decentralized Deadlock Detection and Resolution Algorithm for Generalized Model in Distributed Systems," *Distrib. Parallel Databases*, vol. 29, no. 4, pp. 261–276, Aug. 2011.

[76] M. Singhal, "Deadlock detection in distributed systems," *Computer*, vol. 22, no. 11, pp. 37–48, Nov. 1989.

[77] A. Kshemkalyani and M. Singhal, "A one-phase algorithm to detect distributed deadlocks in replicated databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 6, pp. 880–895, Nov. 1999.

[78] R. Rocha and B. D. Thatte, "Distributed cycle detection in large-scale sparse graphs," Aug. 2015.

[79] G. Oliva, R. Setola, L. Glielmo, and C. N. Hadjicostis, "Distributed Cycle Detection and Removal," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 1, pp. 194–204, Mar. 2018.

[80] X. Lin and J. Chen, "An optimal deadlock resolution algorithm in multidatabase systems," in *Proceedings of 1996 International Conference on Parallel and Distributed Systems*, Jun. 1996, pp. 516–521.

[81] J. B. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," *Proceedings of the American Mathematical Society*, vol. 7, no. 1, pp. 48–50, 1956.

[82] M. Hashemzadeh, N. Farajzadeh, and A. Haghighat, "Optimal detection and resolution of distributed deadlocks in the generalized model," in *14th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP'06)*, Feb. 2006, pp. 4 pp.–.

[83] D. Dolgov and S. Thrun, "Autonomous driving in semi-structured environments: Mapping and planning," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 3407–3414.

[84] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, Apr. 2010.

[85] L. Pallottino, V. G. Scordio, A. Bicchi, and E. Frazzoli, "Decentralized Cooperative Policy for Conflict Resolution in Multivehicle Systems," *IEEE Transactions on Robotics*, vol. 23, no. 6, pp. 1170–1183, Dec. 2007.

[86] A. A. Malikopoulos, C. G. Cassandras, and Y. J. Zhang, "A decentralized energy-optimal control framework for connected automated vehicles at signal-free intersections," *Automatica*, vol. 93, pp. 244–256, Jul. 2018.

[87] S. Bhattacharya, "Search-Based Path Planning with Homotopy Class Constraints," in *Twenty-Fourth AAAI Conference on Artificial Intelligence*, Jul. 2010.

[88] N. Karabakal and J. C. Bean, "A multiplier adjustment method for multiple shortest path problems with coupling constraints," 1995.

[89] Hong Zhang, V. Kumar, and J. Ostrowski, "Motion planning with uncertainty," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 1, May 1998, pp. 638–643 vol.1.

[90] V. Ivan, D. Zarubin, M. Toussaint, T. Komura, and S. Vijayakumar, "Topology-based representations for motion planning and generalization in dynamic environments with interactions," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1151–1163, Aug. 2013.

[91] M. Kuderer, H. Kretzschmar, and W. Burgard, "Teaching mobile robots to cooperatively navigate in populated environments," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov. 2013, pp. 3138–3143.

[92] E. Schmitzberger, J. L. Bouchet, M. Dufaut, D. Wolf, and R. Husson, "Capture of homotopy classes with probabilistic road map," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, Sep. 2002, pp. 2317–2322 vol.3.

[93] 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*, ser. HSCC '12. New York, NY, USA: ACM, 2012, pp. 145–154.

[94] R. Hult, M. Zanon, S. Gros, and P. Falcone, "Primal decomposition of the optimal coordination of vehicles at traffic intersections," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec. 2016, pp. 2567–2573.

[95] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, Jul. 2011.

[96] G. H. Goluh and C. F. Van Loan, "Matrix computations," *Johns Hopkins*, 1996.

[97] R. Nishihara, L. Lessard, B. Recht, A. Packard, and M. Jordan, "A General Analysis of the Convergence of ADMM," in *International Conference on Machine Learning*, Jun. 2015, pp. 343–352.

[98] J. Eckstein and D. P. Bertsekas, "On the Douglas—Rachford splitting method and the proximal point algorithm for maximal monotone operators," *Mathematical Programming*, vol. 55, no. 1, pp. 293–318, Apr. 1992.

[99] M. Fortin and R. Glowinski, *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems*. Elsevier, Apr. 2000.

[100] ——, "Chapter 1 Augmented Lagrangian Methods in Quadratic Programming," in *Studies in Mathematics and Its Applications*, ser. Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems, M. Fortin and R. Glowinski, Eds. Elsevier, Jan. 1983, vol. 15, pp. 1–46.

[101] D. Gabay, "Chapter IX Applications of the Method of Multipliers to Variational Inequalities," in *Studies in Mathematics and Its Applications*, ser. Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems, M. Fortin and R. Glowinski, Eds. Elsevier, Jan. 1983, vol. 15, pp. 299–331.

[102] C. Liu, C. Lin, Y. Wang, and M. Tomizuka, "Convex feasible set algorithm for constrained trajectory smoothing," in *2017 American Control Conference (ACC)*, May 2017, pp. 4177–4182.

[103] C. Liu, C.-Y. Lin, and M. Tomizuka, "The Convex Feasible Set Algorithm for Real Time Optimization in Motion Planning," *SIAM Journal on Control and Optimization*, vol. 56, no. 4, pp. 2712–2733, Jan. 2018.

[104] J. Barzilai and J. M. Borwein, "Two-Point Step Size Gradient Methods," *IMA Journal of Numerical Analysis*, vol. 8, no. 1, pp. 141–148, Jan. 1988.

[105] A. Themelis and P. Patrinos, "Douglas-Rachford splitting and ADMM for nonconvex optimization: Tight convergence results," *SIAM Journal on Optimization*, vol. 30, no. 1, pp. 149–181, Jan. 2020.

[106] B. Zhou, L. Gao, and Y.-H. Dai, "Gradient Methods with Adaptive Step-Sizes," *Computational Optimization and Applications*, vol. 35, no. 1, pp. 69–86, Sep. 2006.

[107] H. Wang and A. Banerjee, "Online Alternating Direction Method (longer version)," *arXiv:1306.3721 [cs, math]*, Jul. 2013.

[108] T. Suzuki, "Dual averaging and proximal gradient descent for online alternating direction multiplier method," in *In Proceedings of the 30th International Conference on Machine Learning*, 2013, pp. 392–400.

[109] S. Hosseini, A. Chapman, and M. Mesbahi, "Online distributed ADMM via dual averaging," in *53rd IEEE Conference on Decision and Control*, Dec. 2014, pp. 904–909.

[110] S. Liu, J. Chen, P.-Y. Chen, and A. Hero, "Zeroth-Order Online Alternating Direction Method of Multipliers: Convergence Analysis and Applications," in *International Conference on Artificial Intelligence and Statistics*, Mar. 2018, pp. 288–297.

[111] M. S. Darup, G. Book, and P. Giselsson, "Towards real-time ADMM for linear MPC," in *2019 18th European Control Conference (ECC)*, Jun. 2019, pp. 4276–4282.

[112] S. East and M. Cannon, "ADMM for MPC with state and input constraints, and input nonlinearity," *arXiv:1807.10544 [math]*, Jul. 2018.

[113] L. Ferranti, R. R. Negenborn, T. Keviczky, and J. Alonso-Mora, "Coordination of Multiple Vessels Via Distributed Nonlinear Model Predictive Control," in *2018 European Control Conference (ECC)*, Jun. 2018, pp. 2523–2528.

[114] Y. Wang, W. Yin, and J. Zeng, "Global Convergence of ADMM in Nonconvex Nonsmooth Optimization," *Journal of Scientific Computing*, vol. 78, no. 1, pp. 29–63, Jan. 2019.

[115] D. Limon, T. Alamo, F. Salas, and E. Camacho, "On the stability of constrained MPC without terminal constraint," *IEEE Transactions on Automatic Control*, vol. 51, no. 5, pp. 832–836, May 2006.

[116] D. Limon Marruedo, T. Alamo, and E. Camacho, "Stability analysis of systems with bounded additive uncertainties based on invariant sets: Stability and feasibility of MPC," in *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, vol. 1, May 2002, pp. 364–369 vol.1.

[117] J. Löfberg, "Oops! I cannot do it again: Testing for recursive feasibility in MPC," *Automatica*, vol. 48, no. 3, pp. 550–555, Mar. 2012.

[118] M. Alamir, "Stability proof for nonlinear MPC design using monotonically increasing weighting profiles without terminal constraints," *Automatica*, vol. 87, pp. 455–459, Jan. 2018.

[119] Z. Liu and O. Stursberg, "Recursive Feasibility and Stability of MPC with Time-Varying and Uncertain State Constraints," in *2019 18th European Control Conference (ECC)*, Jun. 2019, pp. 1766–1771.

[120] M. N. Zeilinger, C. N. Jones, D. M. Raimondo, and M. Morari, "Real-time MPC - Stability through robust MPC design," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) Held Jointly with 2009 28th Chinese Control Conference*, Dec. 2009, pp. 3980–3986.

[121] Z. Marvi and B. Kiumarsi, "Safety Planning Using Control Barrier Function: A Model Predictive Control Scheme," in *2019 IEEE 2nd Connected and Automated Vehicles Symposium (CAVS)*, Sep. 2019, pp. 1–5.

[122] R. Mangharam, D. Weller, R. Rajkumar, P. Mudalige, and F. Bai, "GrooveNet: A Hybrid Simulator for Vehicle-to-Vehicle Networks," in *2006 Third Annual International Conference on Mobile and Ubiquitous Systems: Networking Services*, Jul. 2006, pp. 1–8.

[123] R. Mangharam, J. J. Meyers, R. Rajkumar, D. D. Stancil, J. S. Parikh, H. Krishnan, and C. Kellum, "A Multi-hop Mobile Networking Test-bed for Telematics," 2005.

[124] I. W. Ho, K. K. Leung, J. W. Polak, and R. Mangharam, "Node Connectivity in Vehicular Ad Hoc Networks with Structured Mobility," in *32nd IEEE Conference on Local Computer Networks (LCN 2007)*, Oct. 2007, pp. 635–642.

[125] U. C. Bureau, "TIGER/Line Shapefiles," https://www.census.gov/geographies/mapping-files/time-series/geo/tiger-line-file.html.

[126] A. Bhat, S. Aoki, and R. Rajkumar, "Tools and Methodologies for Autonomous Driving Systems," *Proceedings of the IEEE*, vol. 106, no. 9, pp. 1700–1716, Sep. 2018.

[127] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," *arXiv:1711.03938 [cs]*, Nov. 2017.

[128] M. Dupuis, M. Strobl, and H. Grezlikowski, "Opendrive 2010 and beyond–status and future of the de facto standard for the description of road networks," in *Proc. of the Driving Simulation Conference Europe*, 2010, pp. 231–242.

[129] C. Liu, W. Zhan, and M. Tomizuka, "Speed profile planning in dynamic environments via temporal optimization," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2017, pp. 154–159.

[130] D. Hrovat, S. Di Cairano, H. Tseng, and I. Kolmanovsky, "The development of Model Predictive Control in automotive industry: A survey," in *2012 IEEE International Conference on Control Applications*, Oct. 2012, pp. 295–302.

Glossary

List of Acronyms

ADMM	Alternating Direction Method of Multipliers
AMP-IP	Advanced Maximum Progression Intersection Protocol
BFS	Breadth-First Search
BSM	Basic Safety Message
C-V2X	Cellular Vehicle to Everything
CAM	Cooperative Awareness Message
CFS	Convex Feasible Set
DFS	Depth-First Search
DRS	Douglas-Rachdord splitting
DSRC	Dedicated Short-Range Communication
FCFS	First Come First Serve
HGCS	Hypothetical Generic Communication Standard
ITS-G5	Intelligent Transport Systems-G5
O-ADMM	Online Alternating Direction Method of Multipliers
OA-ADMM	Online Adaptive Alternating Direction Method of Multipliers
LICP	Look-ahead Intersection Control Policy
MCR	Modular Conflict Resolution
SQP	Sequential Quadratic Programming
TDCR	Timeslot-based Distributed Conflict Resolution
V2D	Vehicle to Device
V2G	Vehicle to Grid
V2I	Vehicle to Infrastructure
V2N	Vehicle to Network
V2P	Vehicle to Pedestrian

V2V	Vehicle to Vehicle
V2X	Vehicle to Everything
VIN	Vehicle Identification Number
WAVE	Wireless Access in Vehicular Environments