Thesis for the Degree of Doctor of Philosophy

Optimization-based coordination strategies for connected and autonomous vehicles

Robert Hult

CHALMERS

Department of Electrical Engineering
Chalmers University of Technology

Göteborg, Sweden 2019

Optimization-based coordination strategies for connected and autonomous vehicles

Robert Hult

ISBN: 978-91-7905-108-2

Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie nr 4575
ISSN 0346-718X

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE–412 96 Göteborg
Sweden
Telephone: +46 (0)31 – 772 1000
Email: robert.hult@chalmers.se; roberthult84@gmail.com

Front Cover:
Illustration of vehicles crossing an intersection without traffic-lights.
Artwork by the author.

Typeset by the author using LaTeX.

Chalmers Reproservice
Göteborg, Sweden 2019

To Elin and Gustav

Abstract

Automated vehicles (AV) are expected to reach the consumer market within the next decade. Once AVs become ubiquitous, they could resolve difficult traffic situations through communication-based cooperation. Intersections are of particular interest in this context, as they form bottlenecks in the traffic system and are responsible for a large share of all accidents. Rather than relying on traffic lights, road signs and rules, AVs could employ cooperative strategies to decide how an intersection should be crossed safely and efficiently. However, designing efficient coordination strategies for AVs at intersections is challenging, as computationally hard problems are involved, with a safety-critical dependence on both wireless communication and imprecise sensing.

This thesis treats control algorithms for cooperative coordination of AVs at intersections. The proposed algorithms are based on Optimal Control (OC) formulations of the coordination problem and aim at finding the optimal control commands for each vehicle through a two-stage approximation procedure. In the first stage, the order in which the vehicles cross the intersection is determined using a heuristic based on Mixed-Integer Quadratic Programming (MIQP). In the second stage, the optimal control commands for each vehicle are found under a fixed crossing order. Two algorithms are presented that solves the problem of the second stage in a communication efficient, distributed fashion.

In the first algorithm, the problem is decomposed into one master-problem and one sub-problem for each vehicle. The master-problem is solved using a Sequential Quadratic Programming (SQP) algorithm, where most computations are performed in parallel on-board the vehicles.

In the second algorithm, the problem is solved using a Primal-Dual Interior Point (PDIP) method. The computations involved are separable so that the largest part can be performed in parallel on-board the vehicles, a lesser part in parallel on lead-vehicles for each lane, and a small part at a central network node.

The two-stage approximation procedure is used in a Model Predictive Controller (MPC), and conditions for persistent feasibility and stability are derived. Performance of the MPC-based closed-loop controller is assessed

in simulation, and compared to traffic-lights and alternative coordination algorithms. The results demonstrate that the two-stage approach outperforms existing alternatives, with almost zero average travel-time delay and a marginal increase in energy consumption compared to cruising at constant speed.

An MPC controller based on the SQP algorithm is verified experimentally at a test-track with three real vehicles. The results demonstrate that efficient coordination is practically realizable through communication-based optimization and MPC. In particular, the experiments show that the MPC algorithm performs well under adverse conditions with significant sensor noise, communication impairments and external perturbations.

Keywords: Connected Automated Vehicles, Intersection Coordination, Model Predictive Control, Optimal Control, Optimization

Acknowledgments

As I'm approaching the end of my journey at Chalmers, I'm starting to realize what privilege it has been to work with, learn from, and make friends with so many great people. I would therefore like to take a moment to acknowledge those that directly and indirectly made my journey possible.

First and foremost, my supervisor Professor Paolo Falcone. Thank you Paolo for giving me the possibility to pursue a doctoral degree, for pushing me when needed and for always being supportive.

I also want to thank my co-supervisor Professor Henk Wymeersch for all his support and Dr. Gabriel De Campos for all his help during my first year.

Professors Mario Zanon and Sébastien Gros deserve particular recognition. I've learned an enormous amount from both of you and consider myself lucky to call you my friends. Without your help and expertise, this thesis would have been very different, and without all the discussions and laughs over coffee, the last years much less enjoyable.

I am also deeply grateful for all the great colleagues at the department. In particular, I want to thank Giuseppe, who started his Ph.D.-journey at the same time as I started mine. Thank you Giuseppe for all the good times and for being a great friend.

I also want to thank Emil, Simon, Fredrik, Ankit, Elena, Anton, Ivo and the rest of the past and present SYSCON PhD students and Post-Docs. You truly made the last five years great and I will miss all of you!

A big thanks also goes to Erik, Christopher, Markus and the other guys in Henks research-group, to Marco, Albin, Arun and the other students who participated in GCDC with me and to Arpit and Fredrik at the REVERE lab for their assistance in the experimental part of my work.

I am also thankful for my family and friends outside Chalmers. Thank you for showing patience and supporting me, now and always.

Finally, my wife deserves the biggest recognition of all, for the endless support she has given me through the ups-and-downs of the last five years. I love you Elin.

Robert Hult
Göteborg, April, 2019

List of Publications

This thesis is based on the following appended papers:

Paper A
R. Hult, M. Zanon, S. Gros and P. Falcone, "Optimal Coordination of Automated Vehicles at Intersections: Theory and Experiments", to appear in *IEEE Transactions on Control Systems Technology*

Paper B
R. Hult, M. Zanon, G. Frison, S. Gros and P. Falcone, "Experimental Validation of a Semi-Distributed SQP Method for Optimal Coordination of Automated Vehicles at Intersections", submitted for possible publication in *Optimal Control Applications and Methods*.

Paper C
R. Hult, M. Zanon, S. Gros and P. Falcone, "An MIQP-based Heuristic for Optimal Coordination of Vehicles at Intersections", presented at the *Conference on Decision and Control*, 2018

Paper D
R. Hult, M. Zanon, S. Gros and P. Falcone, "Optimization-based Coordination of Connected, Automated Vehicles at Intersections", submitted for possible publication in *Vehicle System Dynamics*,

Paper E
R. Hult, M. Zanon, S. Gros and P. Falcone, "Energy-Optimal Coordination of Automated Vehicles at Intersections", presented at *European Control Conference*, 2018

Paper F
R. Hult, M. Zanon, S. Gros and P. Falcone, "A Distributed Interior Point Algorithm for Optimal Coordination of Automated Vehicles at Intersections", to be submitted.

Other publications

Related publications by the author not included in the thesis

[i] **R. Hult**, M. Zanon, S. Gros and P. Falcone, "Optimal coordination of Automated Vehicles at Intersections with turns", to be presented at *European Control Conference*, 2019

[ii] **R. Hult**, F. E. Sancar, M. Jalalmaab, A. Vijayan, A. Severinson, M. di Vaio, P. Falcone, B. Fidan and S. Santini, "Design and Experimental Validation of a Cooperative Driving Control Architecture for the Grand Cooperative Driving Challenge 2016", in *Transactions on Intelligent Transportation Systems*, Volume 19, Issue 4, 2018.

[iii] **R. Hult**, M. Zanon, S. Gros and P. Falcone, "Primal decomposition of the optimal coordination of vehicles at traffic intersections", in *Proceedings of the 55th IEEE Conference on Decision and Control*, 2016

[iv] **R. Hult**, G.R. Campos, E. Steinmetz, L. Hammarstrand, P. Falcone and H. Wymeersch, "Coordination of Cooperative Autonomous Vehicles: Toward safer and more efficient road transportation", in *Signal Processing Magazine*, Volume 33, Issue 6, 2016.

[v] **R. Hult**, G.R. Campos, H. Wymeersch and P. Falcone, "An approximate solution to the optimal coordination problem for autonomous vehicles at intersections", in *Proceedings of the American Control Conference*, 2015.

[vi] E. Steinmetz, **R. Hult**, Z. Zou, R. Emardson, F. Brännström, P. Falcone and H. Wymeersch, "Collision-Aware Communication for Intersection Management of Automated Vehicles", in *IEEE Access*, 2018.

[vii] M. Zanon, **R. Hult**, S. Gros and P. Falcone, "Experimental Validation of Distributed Optimal Vehicle Coordination", in *European Control Conference*, 2018

[viii] E. Steinmetz, **R. Hult**, G.R. Campos, M. Wildemeersch, P. Falcone and H. Wymeersch, "Communication analysis for centralized intersection crossing coordination", in *11th International Symposium on Wireless Communication Systems*, 2014.

[ix] G.R. Campos, P. Falcone, **R. Hult**, H. Wymeersch and Jonas Sjöberg, "Traffic coordination at road intersections: Autonomous decision-making algorithms using model-based heuristics", in *Intelligent Transportation Systems Magazine*, Volume 9, Issue 1, 2017.

[x] M. Di Vaio, P. Falcone, **R. Hult**, A. Petrillo, A. Salvi, S. Santini, "Design and experimental validation of a distributed interaction protocol for connected autonomous vehicles at road intersection", submitted to *Transactions on Vehicular Technology.*

[xi] Y. Jiang, M. Zanon, **R. Hult** and B. Houska, "Distributed Algorithm for Optimal Vehicle Coordination at Traffic Intersections", in *IFAC World Congress*, 2017.

[xii] G.R. Campos, P. Falcone, H. Wymeersch, **R. Hult** and J. Sjöberg, "Cooperative receding horizon conflict resolution at traffic intersections", in *Proceedings of the 53rd IEEE Conference on Decision and Control*, 2014.

[xiii] J. Shi, Y. Zheng, Y. Jiang, M. Zanon, **R. Hult** and B. Houska, "Distributed control algorithm for vehicle coordination at traffic intersections", in *European Control Conference*, 2018.

Acronyms

AV	Automated Vehicle
CAV	Connected/Cooperative Automated Vehicle
CQ	Constraint Qualification
FONC	First-Order Necessary Conditions
GNSS	Global Navigation Satelite System
KKT	Karush-Kuhn Tucker (conditions)
LICQ	Linear Independence Constraint Qualification
LP	Linear Program
MINLP	Mixed Integer Nonlinear Program
MIQP	Mixed Integer Quadratic Program
MPC	Model Predictive Control
NLP	Nonlinear program
OC(P)	Optimal Control (Problem)
PDIP	Primal-Dual Interior Point
QP	Quadratic Program
SOSC	Second-Order Sufficient Conditions
SQP	Sequential Quadratic Programming

Contents

Part I

Introductory Chapters

Chapter 1

Background and Outline

The idea of self-driving vehicles has received increasing attention in recent years, and the technology for automated vehicles (AV) is developing rapidly. Several automakers are already field-testing highly automated systems, and they are expected to reach the consumer market within in a few years. Considering the profound impact this could have on society, it is possible that we are witnessing the beginning of a technological revolution.

The work presented in this thesis concerns an application of significance in this context: *the coordination of automated vehicles at intersections*. The idea is that once vehicles are fully automated, they can circumvent the need for traffic lights and rules, and instead jointly decide how intersections should be crossed. This could potentially reduce unnecessary stop-and-go traffic and improve the performance of the overall traffic system.

1.1 Why automated vehicles?

Two often quoted arguments in favor of AVs are increased safety and productivity. In respect of the former, the potential for improvement is large, since more than 90% of all traffic accidents are attributed to human factors [1]. The economic effects are more difficult to assess at this time, but some general predictions have been made. For instance, AVs will allow those currently occupied with driving to instead engage in more fruitful activities, which could be a productivity boost both directly and indirectly. For some commercial applications, e.g. logistics and public transport, the possibility to remove the driver could also lower the demand of human labor [2]. Moreover, since AVs are not limited by the perception and decision-making processes of humans, they could be operated in a more energy efficient manner. Even though the complex behavior on an aggregated level is to be determined, this could have a large impact on the traffic system's energy demand [3].

Figure 1.1: Automated Vehicles enable other activities than driving.

Additionally, since AVs could operate without a human present, the same vehicle could, after it has been used by one person, move on to be used by another. This would make it possible to re-purpose much of the area now devoted to parking, which could have a dramatic impact on how cities are built [4]. The possibility of car-sharing also puts private vehicle ownership into question, and makes AVs possible platforms for a resource efficient transportation-as-a-service paradigm.

1.2 Where are we now?

While the current state of technology puts vehicle automation within reach, the allure of self driving cars has been with the automobile almost since the beginning. For instance, far beyond the technological limitations of its time, a remote-controlled automated traffic system was demonstrated in 1939, during General Motors *Futurama* exhibit at the World Fair [5]. With some experiments during the 60s [6], a few demonstrations of automated functionality were performed on specifically prepared roads in the 70's [7]. Development continued during the 80s, which saw some tests on public roads [8]. In the 90's, a number of research projects were conducted that demonstrated that vehicles with automated functionalities were able to drive 1000s of kilometers on various types of public roads [9, 10]. However, it was not until the mid 2000s that vehicles with capabilities that we would recognize as highly automated started to emerge. The decade saw several demonstrations with fully automated vehicles traveling extended distances in complex environments [11]. In this context, the 2007 Urban Challenge is usually considered a landmark, as it for the first time showed interactions between different automated vehicles, constrained by traffic rules [12].

During the late 2000s, partial driving automation in the form of Advanced

Figure 1.2: Automated Vehicle from Uber

Figure 1.3: Platooning

Driver Assistance Systems (ADAS), such as Adaptive Cruise Control and Lane Keeping Aids, were introduced to the consumer market by a number of automakers. From the mid 2010's the ADAS evolved, and started allowing monitored "hands-and-feet off" automation in low-speed situations (e.g. traffic-jams). With more recent systems like Tesla's Autopilot, Volvo's Pilot Assist or Nissan's ProPilot, supervised partial automation has been made available at higher speeds outside urban environments. At present, several major automakers as well as companies like Waymo, Zenuity, Uber and Lyft, are developing automated systems where little or no driver supervision will be required. Currently, millions of kilometers have been driven on all types of public roads [13], and the projections by the industry leaders are that these systems will reach the consumer market between 2020 and 2030 [14].

Motivated by the ubiquitous availability of wireless communication and potential improvements of AV performance through information-sharing, technologies for *connected* automated vehicles (CAV) have been developed in parallel. Possible applications include cooperative perception, where the vehicles leverage each others sensors to form more accurate representations of the environment [15], and cooperative control, where decision-systems are supported by shared information.

Perhaps the most known example of the latter is *platooning*, where the CAVs are driven with small inter-vehicle distances on highways in order to reduce air-drag and thereby energy demand. A number of demonstrations of platooning have been carried out, including the American PATH program in the 90s [16], the European Sartre Project in the late 2000s [17] and the Grand Cooperative Driving Challenges in the 2010s [18, 19].

A second example of cooperative control is coordination of CAVs at intersections, where the vehicles cooperate in order to improve both safety and efficiency. This application is the focus of this thesis, and the problem is detailed next.

(a) Cause (b) Effect

Figure 1.4: Inefficient use of infrastructure leads to congestion.

1.3 The intersection problem

An intersection is a place where one or more roads cross or merge, typically with inbound traffic from different directions. This creates a shared space that all vehicles must utilize for some time. The necessary use of the shared space increases the risk of accidents, and the situation is complicated further by the presence of vulnerable road-users, such as pedestrians and bicyclists.

Due to the many possible types of accidents, intersections are currently one of the most dangerous components of the traffic system. It has for instance been reported that more than 20% of the traffic related fatalities and 40% of the injuries in the EU occur at intersections [20], with similar numbers found in the US [21]. Therefore, intersections are also one of the most regulated parts of the traffic system, often governed by a combination of traffic-lights, signs and right-of-way rules. As such, they tend to induce lower average speeds and form bottlenecks for the traffic flow, which causes congestion, increased emissions and energy-waste (e.g. through deceleration/acceleration and idling [22]). Besides the impact on the environment and quality of life, the U.S. Treasury has estimated that the annual loss due to congestion to 100 billion dollars in the US alone [23].

A common strategy to resolve congestion problems is to expand the infrastructure by e.g. adding more lanes, tunnels or overpasses. However, since the road-traffic system already today claims a significant part of the exploitable ground surface in urban areas, such expansions are undesirable and not even realizable at many places.

The introduction of CAVs presents a potential remedy: rather than relying on traffic lights, road signs and right-of-way rules, the intersections could be managed completely by coordination algorithms. The central idea is to find control commands for each vehicle such that the intersection is crossed efficiently and without collisions. This is in contrast to the strategies used today, where the traffic flows on the incoming roads are

controlled. Since this would allow the shared area in the intersection to be used more efficiently, overall performance could be improved. In the full CAV penetration case, the vehicles could be controlled to cross the intersection in virtually uninterrupted, tightly packed streams. This would enable safer operation at higher speeds with less energy waste, and increase the capacity of existing infrastructure. The design of algorithms that generate the control commands which achieves this goal is thus central.

1.3.1 Challenges

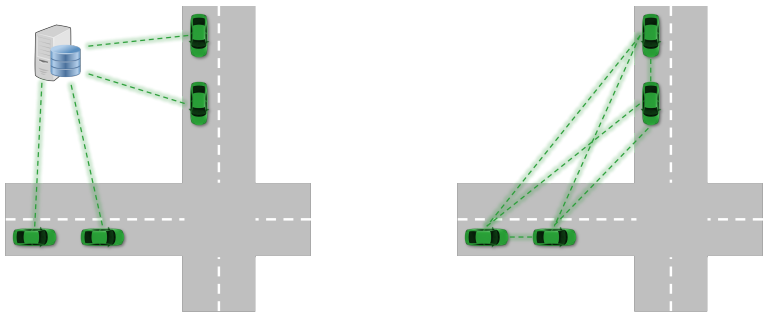
The design of coordination algorithms for the intersection problem is associated with a number of challenges [24]. First, the computational problem of finding collision free motion profiles is combinatorial and hard [25]. This is particularly relevant if more than a few vehicles are involved and the controller is designed to be optimal in some sense. Second, it is not evident how a control system should be designed and where computation should take place. Fully and partly centralized, distributed or decentralized approaches for computation and control are plausible, but differ in various critical respects [24]. Third, all interactions must be performed over wireless channels, and are restricted by their limitations [26]. This is particularly relevant for dense scenarios, as congestion of the communication channel is likely. Finally, a number of difficult uncertainties must be handled. One example is sensor inaccuracies, in particular poor positioning. Another is the presence of non-cooperative entities (e.g., non-cooperative vehicles, pedestrians or bicyclists).

1.3.2 Common approaches

The intersection coordination problem has attracted significant attention from the research community. While there are some early contributions (e.g. [27]), most developments followed after the seminal publications [28, 29], which popularized the problem. The existing results are thus relatively recent, with a significant part published the last five years alone. A detailed review of the literature can be found in the two surveys [30, 31].

Simplifying assumptions

Almost all existing algorithms rely on "idealized" problem formulations, where all vehicles are automated, communication impairments neglected and neither sensing inaccuracies nor vulnerable road users are considered. That is, the focus has so far been on the formulation of the problem, its solution and how control strategies could be implemented.



(a) Centralized (b) Decentralized

Figure 1.5: Classification of intersection coordination schemes

Topologies and classification of existing work

The existing algorithms can in general be classified as either *centralized* or *decentralized*. The former are characterized by the use of a central network node, or Intersection Manager (IM) [28, 29, 32–52]. In decentralized approaches on the other hand, the coordination is carried out using only vehicle-to-vehicle interactions and computation on-board the vehicles [53–68].

The existing centralized approaches differ greatly in the extent of centralization. On one end, the problem is formulated and solved centrally and the control commands sent to the vehicles, which essentially are remotely-controlled [40–42]. On the other end, the IM takes a minor role, and almost all computation is performed in a decentralized fashion. Many centralized schemes are positioned in-between, and use the IM as an arbiter to resolve conflicts, leaving the actual control commands to be computed "locally" by the vehicles [28, 29, 32–39, 51, 52]. Some approaches use a centralized problem formulation, but apply iterative procedures to obtain a solution, where most computation takes place on-board the vehicles [43–48].

Methodologies

Much of the early work consisted of centralized reservation-based algorithms [28, 29, 32, 33], where a vehicle first would send a reservation request, detailing when it wishes to be inside the intersection. The IM thereafter examines the request and only grants it if no conflicts with previous reservations are detected. If granted, it is thereafter the responsibility of each vehicle to use the reservation, and if denied, to re-send a different request. Other contributions have considered the reservation concept in a decentralized setting, where collision free solutions are constructed using rule-based interaction protocols [27, 56–58].

A number of alternative methods to find the equivalent of reservations

have been proposed, using e.g., mixed-integer optimization [34, 35], polling-systems theory [36] or scheduling [37–39]. In these algorithms, the solution is found in two steps, where potential collisions are resolved in the first (i.e. reservations are made), and the vehicle motion profiles are computed second.

A different approach has relied on extensions of platooning, [53–55]. In these algorithms, virtual platoons are created, consisting of vehicles on different lanes. With properly selected coordinate transformations, collisions are avoided if the vehicles in the virtual platoon are kept at specified inter-vehicle distances, thus solving the coordination problem with standard, decentralized platooning controllers.

Recently, a number of algorithms based on Optimal Control (OC) techniques have been presented [40–52, 59–68] that aim at optimizing the motion profiles of the vehicles. Noted benefits are the ability to incorporate various constraints, e.g., actuator limitations and traffic rules, and the ability to optimize explicitly stated performance objectives. The latter is in contrast to many non OC-based methods, where performance metrics often only are used to heuristically derive algorithms. However, since the optimization problem is hard to solve, most OC-based methods rely on heuristics to some extent. In this respect, a sub-division can be made into *Sequential/Parallel* [59–68] and *Simultaneous* [40–52] approaches.

In the Sequential/Parallel algorithms, the vehicles are first ordered in a priority list. Each vehicle then optimizes its own motion such that collisions are avoided with higher priority vehicles only, using predictions of those vehicles' future behavior. These methods are typically decentralized.

In contrast, Simultaneous methods jointly optimize the motion of all vehicles. As a consequence, cooperative, rather than selfish behaviors could emerge, improving the overall system performance. For instance, a compact car might speed up to let a truck cross without slowing down. However, simultaneous optimization typically requires iterative algorithms and either rely on fully centralized computation [40–42], or centralized formulations with distributed computation [43–52].

1.4 Contributions

This thesis adopts the OC perspective on the intersection coordination problem. The presence of a central network node or decision maker is assumed, and the proposed OC formulations involve the simultaneous optimization of all vehicles' behavior around the intersection. The thesis focuses on (i) the solution of optimal coordination problems with the objective of balancing the computation between a central node and the vehicles, in order for both computation and communication to scale well with the problem size and (ii) its execution in a closed-loop stable and persistently feasible receding horizon scheme. The following contributions are given:

1. The derivation of a general OC formulation of the intersection problem.
2. An optimization-based heuristic for solving the combinatorial part of the problem (finding the crossing order for the vehicles).
3. Two algorithms that solve the OCP for a fixed crossing order where most computations are parallelized and performed by the vehicles
   (a) One Sequential-Quadratic Programming (SQP)-algorithm applied to a primal decomposition of the OC-problem.
   (b) One Primal-Dual Interior Point-algorithm applied to the centralized problem formulation.
4. The receding horizon application of the OC-formulation in an MPC for intersection coordination, and the derivation of conditions for persistent feasibility and stability.
5. A performance evaluation that establishes the benefits of an optimal control-based approach with simultaneous optimization.
6. The experimental validation of the distributed algorithms and MPC on a real test setup.

1.5 Outline

This thesis is written in the form of a "shortened monograph", with papers appended as support. It is the Author's intention to thereby provide a self-contained summary in Part I, where the specifics are found in Part II. The material is organized as follows:

Part I

Chapter 2 Introduces notation, and summarizes results on Optimization, Optimization Algorithms, Optimal Control and Model Predictive Control that are used in Parts I and II.

Chapter 3 States the assumptions used throughout the thesis, and introduces the vehicle and intersection models used.

Chapter 4 Introduces continuous and discrete-time OC formulations of the intersection coordination problem.

Chapter 5 Introduces tailored distributed numerical methods for solving the OC formulation of the intersection coordination problem.

Chapter 6 Introduces the receding horizon application of the OC formulation (MPC-based coordination), presents conditions for stability and persistent feasibility and discusses results from both simulated and experimental evaluations.

Chapter 7 Provides a summary of the appended papers.

Chapter 8 Concludes the thesis and discuss future research directions.

Part II

Paper A:

R. Hult, M. Zanon, S. Gros and P. Falcone, "Optimal Coordination of Automated Vehicles at Intersections: Theory and Experiments", to appear in *IEEE Transactions on Control Systems Technology*.

Paper B:

R. Hult, M. Zanon, G. Frison, S. Gros and P. Falcone, "Experimental Validation of a Semi-Distributed SQP Method for Optimal Coordination of Automated Vehicles at Intersections", submitted to *Optimal Control Applications and Methods*.

Paper C:

R. Hult, M. Zanon, S. Gros and P. Falcone, "An MIQP-based Heuristic for Optimal Coordination of Vehicles at Intersections", presented at the *Conference on Decision and Control*, 2018.

Paper D

R. Hult, M. Zanon, S. Gros, H. Wymeersch and P. Falcone, "Optimization-based Coordination of Connected, Automated Vehicles at Intersections", submitted to *Vehicle System Dynamics*.

Paper E:

R. Hult, M. Zanon, S. Gros and P. Falcone, "Energy-Optimal Coordination of Automated Vehicles at Intersections", presented at *European Control Conference*, 2018.

Paper F

R. Hult, M. Zanon, S. Gros and P. Falcone, "An Interior Point Algorithm for Optimal Coordination of Automated Vehicles at Intersections", to be submitted.

Chapter 2

Preliminaries

This chapter recalls the tools used in the thesis. While Section 2.1 introduces the notation used, Sections 2.2 and 2.3 provides an overview of some concepts and algorithms from continuous optimization and Section 2.4 introduces Optimal Control and Model Predictive Control. A reader familiar with these topics can therefore skip Sections 2.2-2.4 and read the next chapter without loss of information.

2.1 Notation

Analysis For a function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$, the total derivative of f with respect to x is written as $\frac{\mathrm{d}f}{\mathrm{d}x} \in \mathbb{R}^{m \times n}$, and partial derivative as $\frac{\partial f}{\partial x} \in \mathbb{R}^{m \times n}$. We denote $\frac{\partial f}{\partial x} \in \mathbb{R}^{m \times n}$ the Jacobian of f, $\nabla_x f = \frac{\partial f}{\partial x}^\top$ the Gradient and $\nabla_x^2 f = \frac{\partial}{\partial x} \nabla_x f = \frac{\partial^2 g}{\partial x^2}$ the Hessian.

Vectors The concatenation $z = [x^\top, y^\top]^\top$ of two column vectors $x \in \mathbb{R}^n, y \in \mathbb{R}^m$ is also denoted $z = (x, y)$, so that $z \in \mathbb{R}^{n+m}$. The elements of a vector $x \in \mathbb{R}^n$, are written x_i, so that $x = (x_1, \ldots, x_n)$. We denote by $\mathbf{1}$ and 0, the vector with all 1s and l zeros respectively, and let the sizes be dependent on the context they are used. If nothing is stated otherwise, scalar operators on vectors denotes element-wise application, i.e. $\min(x, 0) = (\min(x_1, 0), \ldots, \min(x_n, 0))$. Equalities and inequalities with vectors of same size are applied element-wise, so that e.g., $x \leq y$ denotes $x_i \leq y_i$, for $x, y \in \mathbb{R}^n$. Similarly, the element-wise addition and subtraction of two vectors $x, y \in \mathbb{R}^n$ of the same size is written $x + y$ and $x - y$, respectively. For $x \in \mathbb{R}^n$ the p norm is written $||x||_p = \sqrt[p]{\sum_{i=1}^n x_i^p}$, where in particular $||x||_\infty = \max(|x_1|, \ldots, |x_n|)$ and $||x||_1 = \sum_{i=1}^n |x_i|$. When the sub-script p is omitted, $||\cdot||$ should be understood as the 2-norm. Finally, for a vector $x \in \mathbb{R}^n$, we let $\mathrm{sizeof}(x) = n$.

Matrices For sub-matrices A_i, $i = 1, \ldots, N$, we write

$$A = \text{blockdiag}(A_1, \ldots, A_N) = \begin{bmatrix} A_1 & & \\ & \ddots & \\ & & A_N \end{bmatrix}. \tag{2.1}$$

That is, A is the block-diagonal matrix with blocks A_i, not necessarily of same size, on the main diagonal, all other elements zero. $D(q) = \text{diag}(x) \in \mathbb{R}^{n \times n}$ is the matrix with the elements of $x \in \mathbb{R}^n$ on the main diagonal and all other elements zero. The identity and zero matrices are written I and 0 respectively, where we let the size be dependent on the context they are used. The positive-definiteness of a square matrix M is written $M \succ 0$, positive semi-definiteness as $M \succeq 0$. The inverse signs are used for negative (semi) definiteness. We write the element-wise addition and subtraction of two matrices A, B using $A + B$ and $A - B$.

Sets and intervals We denote the set of non-zero integers $\{0, \ldots, n\}$ as $\mathcal{I}_n$. The cardinality of a discrete set $\mathcal{X}$ is written $|\mathcal{X}|$, so that e.g. $|\mathcal{I}_n| = n+1$. ForÂ $a, b \in \mathbb{R}$, we let $[a, b] = \{x \mid a \leq x \leq b\}$, $]a, b[= [a, b] = \{x \mid a < x < b\}$, $[a, b[= [a, b] = \{x \mid a \leq x \ < b\}$ and $]a, b] = [a, b] = \{x \mid a < x \leq b\}$.

2.2 Optimization

This section introduces optimization, also known as mathematical programming, provides definitions and recalls some results that are central to the development of optimization algorithms.

2.2.1 A general problem statement

We consider mathematical programs on the form

$$\min_x \quad f(x), \tag{2.2a}$$
$$\text{s.t.} \quad g_i(x) = 0, \qquad i = 1, \ldots, m, \tag{2.2b}$$
$$h_i(x) \leq 0, \qquad i = 1, \ldots, q \tag{2.2c}$$

where $x \in \mathbb{R}^n$, $f : \mathbb{R}^n \mapsto \mathbb{R}$, $g_i : \mathbb{R}^n \mapsto \mathbb{R}$ and $h_i : \mathbb{R}^n \mapsto \mathbb{R}$. The solution to (2.2) is the *decision variable* x such that the *objective function* $f(x)$ assumes its minimal value over the *set of feasible solutions* $\mathcal{X} = \{x \mid g(x) = 0,\ h(x) \leq 0\}$, defined by the *constraints* $g(x) = (g_1(x), \ldots, g_m(x))$ and $h(x) = (h_1(x), \ldots, h_q(x))$.

Depending on the properties of the functions f, g and h, Problem (2.2) has different characteristics. Of particular relevance to this thesis are problems where f, g and h are twice continuously differentiable and g, h are such that x is not bound to lie in a discrete set. We denote these problems *nonlinear programs (NLP)*[1]. In the sequel, we assume that the problems discussed are NLPs when not otherwise stated. NLPs where both $\mathcal{X}$ and f are convex are denoted *convex programs*. These include (but are not restricted to) *linear programs (LP)*, where f, g and h are linear, and convex *quadratic programs (QP)*, in which g, h are linear and f is quadratic and positive semi-definite[2].

We also highlight the problems where all or some elements of x additionally are constrained to assume integer values. These problems are called *integer* and *mixed-integer programs (IP/MIP)*, respectively. Depending on f, g and h, IP/MIPs can be further subdivided as, e.g., *integer linear programs (ILP)*, *mixed-integer NLP (MINLP)* or *mixed-integer QPs (MIQPs)*.

Global and local optima

We say that $x^* \in \mathcal{X}$ is a global minimizer to (2.2) if

$$f(x^*) \leq f(x)\ \forall x \in \mathcal{X}, \tag{2.3}$$

[1] We note that in some texts, the term NLP is used to denote a wider problem class, including problems with integer variables. We stress that this is not the case here.

[2] There are also non-convex QPs, with indefinite or negative definite f. However, these have no relevance to the thesis and are not discussed.

and that x^* is a strict global minimizer if (2.3) only holds with equality at x^*. Similarly, x^* is a local minimizer to (2.2) if

$$\exists \epsilon > 0 \ : \ f(x^*) \leq f(x) \ \forall x \in \mathcal{B}_\epsilon(x^*) \cap \mathcal{X}, \tag{2.4}$$

where $\mathcal{B}_\epsilon(x) = \{ y \in \mathbb{R}^n \mid ||x - y|| < \epsilon \}$. We say that x^* is a strict local minimizer if (2.4) only holds with equality at x^*. Similarly, $f(x^*)$ is denoted the (strict) Global/Local minimum of (2.2). A global minimizer is thus always a local minimizer, but the converse does not hold in general. In the sequel we use minimum/minimizer/minimize and optimum/optimizer/optimize interchangeably to denote $f(x^*)$, x^*, and the process of finding a (local) minimizer of (2.2) respectively.

Lagrange duality

The Lagrange function associated with NLP (2.2) is defined as

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^\top g(x) + \mu^\top h(x), \tag{2.5}$$

where $\lambda \in \mathbb{R}^m$ and $\mu \in \mathbb{R}^q, \mu \geq 0$ are known as the *Lagrange multipliers* or simply the *multipliers* associated with constraints $g(x) = 0$ and $h(x) \leq 0$, respectively. Since $\mu \geq 0$, we have that $\mathcal{L}(x, \lambda, \mu) \leq f(x)$ on $\mathcal{X}$, i.e. (2.5) is a relaxation of $f(x)$ on $\mathcal{X}$. The solution to the *Dual problem*

$$d(\lambda^*, \mu^*) = \max_{\lambda, \mu \geq 0} \quad d(\lambda, \mu), \qquad d(\lambda, \mu) = \min_x \quad \mathcal{L}(x, \lambda, \mu), \tag{2.6}$$

therefore bounds this solution to NLP (2.2), i.e. $d(\lambda^*, \mu^*) \leq f(x^*)$. In this context, the Lagrange multipliers are also known as the *dual variables*, differentiating them from the *primal variables* x of the *primal problem* (2.2).

2.2.2 Necessary conditions for optimality

Provided that constraints g and h satisfies conditions known as *constraint qualifications (CQ)*, the necessary conditions for optimality takes an algebraic form. We highlight in particular the following CQ:

Definition 2.1 (Linear Independence Constraint Qualification (LICQ))**.** *If $\nabla_x g(x)$ and all $\nabla_x h_i(x)$ for which $h_i(x) = 0$ are linearly independent, the linear independence constraint qualification holds at x.*

The first order necessary conditions (FONC) for optimality are summarized in the following classical result, where a proof can be found in e.g. [69]

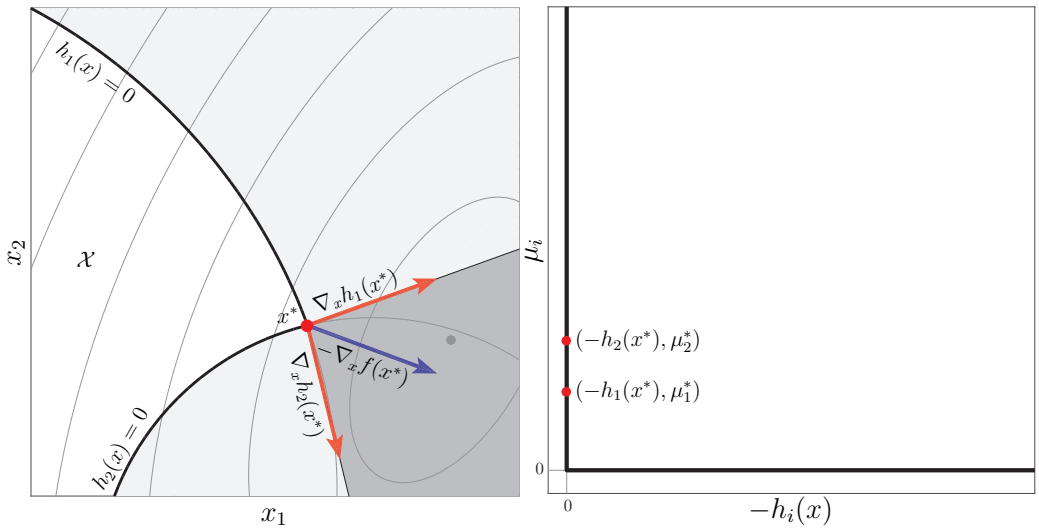


Figure 2.1: Illustration of the KKT conditions (2.7) for an NLP without equality constraints. In (a), the contours of f are shown, with the minimum marked with a gray dot. Satisfaction of (2.7a) implies that $-\nabla_x f(x^*)$ lies in the cone spanned by $\nabla_x h(x^*)$, satisfaction of (2.7c) that $x^* \in \mathcal{X}$. In (b), Satisfaction of (2.7c),(2.7d) and (2.7e) implies that (x^*, s^*) lie in the set drawn in thick black.

Theorem 2.1 (Karush-Kuhn-Tucker (KKT) Necessary Conditions)**.** *If x^* is a local minimizer to NLP (2.2) and constraint qualification holds, $\exists \lambda^*, \mu^*$ such that*

$$\nabla_x \mathcal{L}(x^*, \lambda^*, \mu^*) = 0, \tag{2.7a}$$
$$g(x^*) = 0, \tag{2.7b}$$
$$h(x^*) \leq 0, \tag{2.7c}$$
$$h_i(x^*)\mu_i^* = 0, \qquad i = 1, \ldots, m, \tag{2.7d}$$
$$\mu_i^* \geq 0, \qquad i = 1, \ldots, m \tag{2.7e}$$

In particular, (2.7b),(2.7c) require that x^* is feasible in the primal problem (2.2), and are known as the *primal feasibility* conditions. Similarly, (2.7d) requires that μ^* is feasible in the dual problem and is known as the *dual feasibility* condition. The conditions (2.7a) and (2.7e) are known as the *stationarity* and *complementarity* conditions, respectively, and a vector (x^*, λ^*, μ^*) that satisfies (2.7) is known as a *KKT-point*.

A graphical illustration of the KKT conditions is provided in Figure 2.1.

Sufficient conditions

Since the KKT conditions only are necessary for (local) optimality, we note that all local optima where CQ holds are KKT points but that, in general,

not all KKT points are local optima. Theorem 2.2 establishes sufficient conditions for local optimality, where the proof can be found in e.g. [69], and uses the following definition

Definition 2.2. *A constraint $h_i(x)$ is* active *when $h_i(x) = 0$ and* inactive *when $h_i(x) < 0$. Specifically, the active constraints are* weakly active *when $h_i(x) = 0$ and $\mu_i = 0$, and strictly active when $h_i(x) = 0$ and $\mu_i > 0$. We collect the indices of all active, strictly active and weakly active constraints in $\mathbb{A}, \mathbb{A}^+$ and $\mathbb{A}^0$, respectively, and collect the corrsponding constraints and multipliers in $h_{\mathbb{A}}, h_{\mathbb{A}^+}, h_{\mathbb{A}^0}, \mu_{\mathbb{A}}, \mu_{\mathbb{A}^+}, \mu_{\mathbb{A}^0}$.*

Theorem 2.2 (Sufficient conditions for optimality)**.** *If (x^*, λ^*, μ^*) is a KKT-point for NLP* (2.2) *and CQ holds, x^* is a local minimum if*

$$v^\top \nabla_x^2 \mathcal{L}(x, \lambda, \mu) v \geq 0, \tag{2.8}$$

$$\forall v \in \left\{ v \neq 0 \;\middle|\; \begin{array}{l} \nabla_x g(x)^\top v = 0 \\ \nabla_x h_{\mathbb{A}^+}(x)^\top v = 0, \\ \nabla_x h_{\mathbb{A}^0}(x)^\top v \leq 0, \end{array} \right\}. \tag{2.9}$$

If the inequality (2.8) *is strict, x^* is a strict local minimum.*

That is, $\mathcal{L}(x^*, \lambda^*, \mu^*)$ must be positive-definite in all directions of the argument x that are not blocked by any of the constraints. Is this not the case, there are potentially neighboring points to x^* in $\mathcal{X}$, which achieves lower values of f.

A brief note on convexity

It can be shown that the KKT conditions also are sufficient conditions for optimality for convex problems [70]. In fact, any x^* for which there exists λ^*, μ^* that satisfies (2.7) is also a global minimizer of $f(x)$ on $\mathcal{X}$. This has implications for algorithm design, and makes many convex problems easier to solve than similar NLPs.

2.3 Algorithms for solving NLPs

The focus in this brief overview is on what commonly is called *second-order* methods for NLPs as (2.2). In particular, we discuss the fundamentals of the line-search varieties of the Sequential Quadratic Programming (SQP) and the Primal-Dual Interior Point (PDIP) algorithms. Both methods operate by iteratively updating a primal-dual solution candidate to NLP (2.2). In particular, the methods are "KKT-solvers" as the updates are made by taking Newton-type steps on the KKT-equations (2.7).

2.3.1 Newton's method

Newton's method is a classical procedure for finding the root $y^* \in \mathbb{R}^n$ of a once continuously differentiable function $F : \mathbb{R}^n \mapsto \mathbb{R}^n$. It operates by iteratively updating the *solution candidate* $y^{[k]}$ towards the root to the linearization of F, i.e, by performing the update

$$y^{[k+1]} = y^{[k]} + \alpha^{[k]} \Delta y^{[k]}, \tag{2.10}$$

where

$$\frac{\partial F(y^{[k]})}{\partial y} \Delta y^{[k]} = -F(y^{[k]}). \tag{2.11}$$

Here, Δy is the *Newton direction* and $\alpha^{[k]} \in]0, 1]$ is the *step-size*. Provided that a solution y^* exists, that $\frac{\partial F(y^{[k]})}{\partial y}$ is full rank $\forall k$, that $\alpha^{[k]}$ is chosen appropriately and that $y^{[0]}$ is chosen close enough to y^*, $y^{[k]}$ converges to y^* [71]. Newton's method consequently consists of two major components: the computation of the search direction $\Delta y^{[k]}$ and the determination of the step size $\alpha^{[k]}$.

Newton's method for optimization

In the context of optimization, a Newton-type method can be applied to the KKT equations (2.7) in order to find a KKT point. However, due to the presence of inequalities (2.7c),(2.7d) and the resulting non-smooth complementarity condition (2.7e), the basic Newton scheme cannot be applied directly. This serves as a motivation for the SQP and PDIP algorithms introduced in the following two sections, which both can be interpreted as Newton's methods for inequality constrained programs.

Moreover, for both algorithms, the Newton direction and step-size are selected so that progress towards a feasible local minimum of (2.2) is ensured. Progress is in this context measured through a *merit function* $\phi(x)$, which combines the objective function with penalization of constraint violations. In particular, $\phi(x)$ is selected such that local minimas of (2.2) also are local minimas to $\phi(x)$. Under suitable modifications, $\Delta x^{[k]}$ is found which ensures descent on $\phi(x)$, such that a step size $\alpha^{[k]}$ can be found that ensures progress towards a solution. However, rather than solving the *line-search* problem

$$\alpha^{[k]} = \arg \min_{\alpha \in]0,1]} \quad \phi(x^{[k]} + \alpha \Delta x^{[k]}), \tag{2.12}$$

for the optimal step-size $\alpha^{[k]}$, it is common to instead find "good-enough" $\alpha^{[k]}$ through an inexact, *backtracking* line-search. In particular, starting at some $\alpha^{[k]} \leq 1$, $\alpha^{[k]}$ is reduced until sufficient decrease has been made on ϕ.

An often used criterion for sufficient decrease is the Armijo condition [71], where $\alpha^{[k]}$ is decreased until

$$\phi(x^{[k]} + \alpha^{[k]}\Delta x^{[k]}) < \phi(x^{[k]}) + \gamma\nabla_x\phi(x^{[k]})^\top\Delta x^{[k]}\alpha^{[k]}, \tag{2.13}$$

for $\gamma \in]0, 1/2]$.

2.3.2 SQP

In Sequential Quadratic Programming (SQP), the primal-dual Newton direction is obtained by solving the (strictly) convex QP model of NLP (2.2)

$$\begin{aligned} \min_{\Delta x} \quad & \frac{1}{2}\Delta x^\top H^{[k]}\Delta x + \nabla_x f(x^{[k]})^\top\Delta x && \text{(2.14a)} \\ \text{s.t.} \quad & g(x^{[k]}) + \nabla_x g(x^{[k]})^\top\Delta x = 0, && \text{(2.14b)} \\ & h(x^{[k]}) + \nabla_x h(x^{[k]})^\top\Delta x \leq 0. && \text{(2.14c)} \end{aligned}$$

where $H^{[k]} \succ 0$ is such that

$$v^\top H^{[k]} v > 0, \quad \forall v : [\nabla_x g, \nabla_x h_{\mathbb{A}}]^\top v = 0. \tag{2.15}$$

The primal-dual solution candidates are subsequently constructed through the recursion

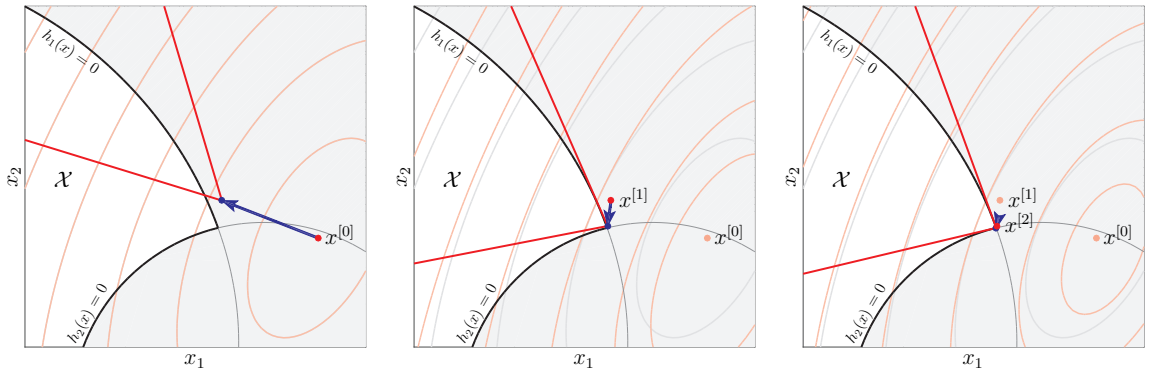
$$\begin{aligned} x^{[k+1]} &= x^{[k]} + \alpha^{[k]}\Delta x^{[k]}, && \text{(2.16a)} \\ \lambda^{[k+1]} &= \lambda^{[k]} + \alpha^{[k]}\Delta\lambda^{[k]}, && \text{(2.16b)} \\ \mu^{[k+1]} &= \mu^{[k]} + \alpha^{[k]}\Delta\mu^{[k]}, && \text{(2.16c)} \end{aligned}$$

where $\Delta\lambda^k = \lambda^k_{\text{QP}} - \lambda^{[k]}$, $\Delta\mu^k = \mu^k_{\text{QP}} - \mu^{[k]}$ and Δx, $\lambda^{[k]}_{\text{QP}}$ and $\mu^{[k]}_{\text{QP}}$ are the primal-dual solutions to (2.14). This Newton-type algorithm thus operates by solving a sequence of quadratic programs. When $\nabla^2_x\mathcal{L}$ satisfies (2.15), $H^{[k]} = \nabla^2_x\mathcal{L}$ can be used, and in other cases a modification $H^{[k]} = \nabla^2_x\mathcal{L} + S^{[k]}$ can be employed, where $S^{[k]} \succeq 0$ is such that (2.15) is satisfied. Provided that the penalty parameter ν is chosen large enough, $\Delta x^{[k]}$ is a descent direction on the ℓ_1 merit function, defined as

$$\phi(x) = f(x) + \nu\left(||g(x)||_1 + ||\max(h(x), 0)||_1\right). \tag{2.17}$$

As discussed above, the step size $\alpha^{[k]}$ be found by a back-tracking line-search on $\phi(x)$ until (2.13) is satisfied.

The basic line-search SQP scheme is summarized Algorithm 1, and an illustrative example is provided in Figure 2.2. Note that at each iterate, the search direction is based on a minimization of a quadratic model of $f(x)$ at $x^{[k]}$, over the set defined by the linearization of constraints (2.2b),(2.2b).



(a) (b) (c)

Figure 2.2: Illustration of SQP for an NLP with inequality constraints. The gray contours show $f(x)$, the red contours the quadratic objective (2.14a), the thick red lines the linearized constraints (2.14c) and the arrows show $\Delta x^{[k]}$

Algorithm 1 A Basic SQP Algorithm. Here, $\epsilon > 0$ is the optimality and feasibility tolerance and the parameter $\beta \in]0, 1]$ determines the rate of reduction in $\alpha^{[k]}$ during the backtracking line-search.

procedure BasicSQP($x^{[0]}$, $\lambda^{[0]}$, $\lambda^{[0]}$)
  $k \leftarrow 0$
  while $\max(||\nabla_x \mathcal{L}||, ||g(x)||, ||\max(h(x), 0)||) > \varepsilon$ **do**
    Solve QP (2.14), set $\alpha^{[k]} := 1$
    while $\phi(x^{[k]} + \alpha^{[k]} \Delta x^{[k]}) \geq \phi(x^{[k]}) + \gamma \nabla_x \phi(x^{[k]})^\top \Delta x^{[k]} \alpha^{[k]}$ **do**
      $\alpha^{[k]} \leftarrow \beta \alpha^{[k]}$.
    end while
    Perform update (2.16)
  end while
end procedure

2.3.3 PDIP algorithms

Primal-dual interior point algorithms are based on the following two modifications of the KKT conditions (2.2)

$$\nabla_x \mathcal{L}(x, \lambda, \mu) = 0, \tag{2.18a}$$
$$g(x) = 0, \tag{2.18b}$$
$$h(x) + s = 0, \tag{2.18c}$$
$$s_i \mu_i - \tau = 0, \tag{2.18d}$$
$$s_i \geq 0, \tag{2.18e}$$
$$\mu_i \geq 0. \tag{2.18f}$$

That is, the primal feasibility condition $h(x) \leq 0$ is replaced with $h(x)+s = 0$, $s \geq 0$, where $s \in \mathbb{R}^q$ is known as the *slack variables* associated with $h(x) \leq 0$, and the complementarity condition is rewritten in terms of s and relaxed using the *barrier-parameter* $\tau \in \mathbb{R}$.

The solution $(x^*_\tau, \lambda^*_\tau, \mu^*_\tau)$ to (2.18) for some fixed τ constitutes an approximate solution to the KKT conditions (2.7) such that $||(x^*_\tau, \lambda^*_\tau, \mu^*_\tau) - (x^*, \lambda^*, \mu^*_i)|| = \mathcal{O}(\tau)$ [71]. When $\tau = 0$, a solution to (2.18) thus coincides with a solution to the KKT conditions (2.7). The fundamental idea in PDIP methods is therefore to solve (2.18a)-(2.18f) as $\tau \to 0$, and thereby obtain a sequence of solution candidates that approach a KKT point of NLP (2.2). In the basic version, a Newton-type method is applied to (2.18a)-(2.18f) for τ fixed until the equations are satisfied to some (τ-dependent) accuracy ε_τ, after which τ is reduced. Collecting $z^{[k]}_\tau = (x^{[k]}_\tau, \lambda^{[k]}_\tau, \mu^{[k]}_\tau, s^{[k]}_\tau)$ and $\Delta z^{[k]}_\tau = (\Delta x^{[k]}_\tau, \Delta\lambda^{[k]}_\tau, \Delta\mu^{[k]}_\tau, \Delta s^{[k]}_\tau)$, the sequence of solution candidates is given by

$$z^{[k+1]} = z^{[k]} + \alpha^{[k]}\Delta z^{[k]}, \tag{2.19}$$

$$M(z^{[k]})\Delta z^{[k]} = -r_\tau(z^{[k]}), \tag{2.20}$$

with

$$M(z^{[k]}) = \begin{bmatrix} H^{[k]} & \nabla_x g & \nabla_x h & \\ \nabla_x g^\top & & & I \\ \nabla_x h^\top & & & \Sigma \\ & & I & \end{bmatrix}, \quad r_\tau(z^{[k]}) = \begin{bmatrix} \nabla_x \mathcal{L}(x^{[k]}, \lambda^{[k]}, \mu^{[k]}) \\ g(x^{[k]}) \\ h(x^{[k]}) + s^{[k]} \\ \mu^{[k]} - D(s^{[k]})^{-1}\mathbf{1}\tau \end{bmatrix}, \tag{2.21}$$

where $D(v) \in \mathbb{R}^{l\times l}$ is the matrix with the elements of $v \in \mathbb{R}^l$ on the main diagonal, $\Sigma = D(s)^{-1}D(\mu)$ and $H^{[k]}$ is a matrix such that

$$v^\top \begin{bmatrix} H^{[k]} & \\ & \Sigma \end{bmatrix} v > 0, \quad \forall v \ : \ \begin{bmatrix} \nabla_x g^\top & \\ \nabla_x h^\top & I \end{bmatrix} v = 0. \tag{2.22}$$

Since all inequalities are simple bounds, a tentative $\alpha^{[k]}$ can easily be selected such that (2.18e),(2.18f) holds at $x^{[k+1]}$. In particular, if $s^{[k]} > 0, \mu^{[k]} > 0$, enforcing $\alpha^{[k]} < \min(\alpha^{\max_s}, \alpha^{\max_\mu})$, where $s^{[k]} + \alpha^{\max_s}\Delta s^{[k]} = 0$, $\mu^{[k]} + \alpha^{\max_\mu}\Delta\mu^{[k]} = 0$ ensures that $s^{[k+1]} > 0$, $\mu^{[k+1]} > 0$. Consequently, selecting $s^{[0]} > 0, \mu^{[0]} > 0$ ensures that the solution candidates remain strictly in the *interior* of the set defined by inequalities (2.18e),(2.18f). The final selection of an improving $\alpha^{[k]}$ is thereafter made using a backtracking line-search on a merit function.

Algorithm 2 A basic PDIP Algorithm. Here, $\gamma \in [0, 1[$

procedure BasicPDIP($x^{[0]}$, $\lambda^{[0]}$, $\mu^{[0]}$, $s^{[0]}$, τ)
  $k \leftarrow 0$
  while $\tau > \varepsilon$ **do**
    while $||r_\tau(z^{[k]})|| > \varepsilon_\tau$ **do**
      Solve (2.20) for $\Delta z^{[k]}$
      Find $\alpha^{\max} < \min(\alpha^{\max_s}, \alpha^{\max_\mu})$
      Perform line-search on merit function to find $\alpha^{[k]} \leq \alpha^{\max}$
      Perform update (2.19)
    end while
    $\tau \leftarrow \min(\gamma\tau, \epsilon)$
  end while
end procedure

A basic PDIP procedure is summarized in Algorithm 2, with an illustration of the solution process shown in Figure 2.3. Here we have used a simple update rule for τ, while practical PDIP algorithms, among other things, employ more sophisticated procedures. In particular, most PDIP algorithms are what is known as path-following methods, where accurate solution of (2.18) is avoided for large τ. Instead, these algorithms inexactly tracks the *central path* $p(\tau) = z^*_\tau$ towards the solution.

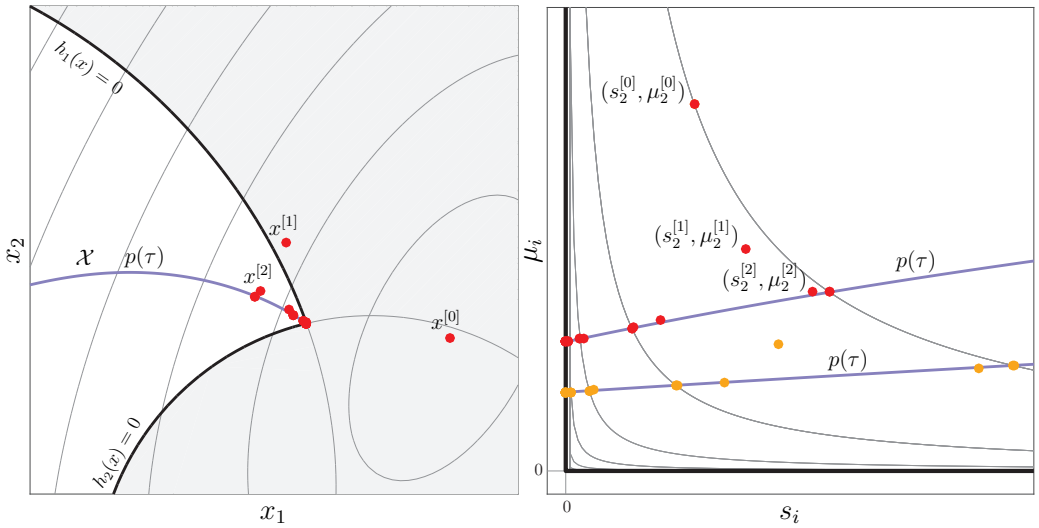
2.3.4 Sensitivity analysis

For the *parametric* NLP

$$\min_x \quad f(x, p) \tag{2.23a}$$
$$\text{s.t.} \quad g(x, p) = 0, \tag{2.23b}$$
$$h(x, p) \leq 0, \tag{2.23c}$$

with parameter $p \in \mathbb{R}^l$, both the (local) primal-dual solution z^* and *optimal value function* $f(x^*)$, are functions of p, i.e. $z^*(p) = (x^*(p), \lambda^*(p), \mu^*(p))$, $f^*(p)$. Moreover, if at $x^*(p)$, f,g,h are twice continuously differentiable in p, LICQ and SOSC holds, and there are no weakly active constraints ($\mathbb{A}^0 = \emptyset$), the sensitivity to variations in p can be obtained through the Implicit Function Theorem [72]. In particular, collecting the KKT equality conditions for the set of strictly active constraints in $r_{\mathbb{A}^+}(z^*(p), p) = (\nabla_x \mathcal{L}, g(x^*), h_{\mathbb{A}^+}(x^*), D(\mu^*_{\mathbb{A}^+})h_{\mathbb{A}^+}(x^*))$, where $z^*(p) = (x^*(p), \lambda^*(p), \mu^*_{\mathcal{A}^+}(p))$, $\frac{\mathrm{d}z^*}{\mathrm{d}p}$ is given by

$$\frac{\partial r_{\mathbb{A}^+}}{\partial p}\frac{\mathrm{d}z^*}{\mathrm{d}p} + \frac{\partial r_{\mathbb{A}^+}}{\partial p} = 0, \tag{2.24}$$



(a) (b)

Figure 2.3: Illustration of a basic PDIP method where (a) shows the primal iterates and the primal central path and (b) shows the iterates and central path in (s, μ). The parabolas are $\mu_i s_i = \tau$ for different values of τ.

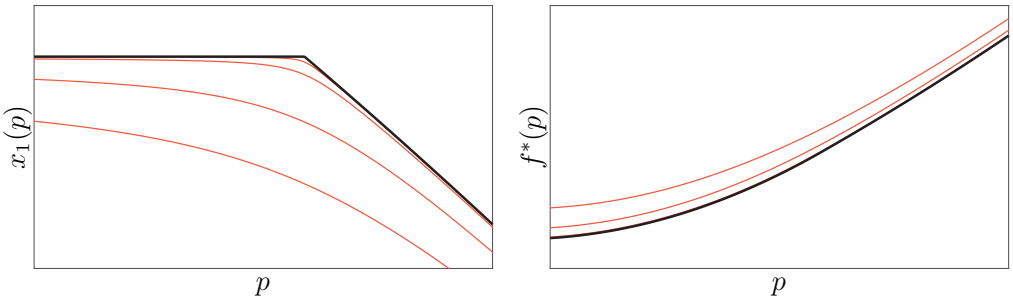
which together with the KKT conditions gives that

$$\frac{\mathrm{d}f^*}{\mathrm{d}p} = \frac{\partial \mathcal{L}}{\partial p}. \tag{2.25}$$

The derivatives exist whenever the system (2.24) has a solution, i.e. when $\frac{\partial r_{\mathbb{A}^+}}{\partial p}$ has full rank, which holds true under LICQ and SOSC. Consequently, if LICQ and SOSC holds at $x^*(p)$, $\forall p \in \mathcal{P} \subseteq \mathbb{R}^l$ we have that (a) $w(p)$ is continuous and $f^*(p)$ is once continuously differentiable on $\mathcal{P}$ (b) that both $z^*(p)$ and $f^*(p)$ are piece-wise twice continuously differentiable on $\mathcal{P}$, with undefined derivatives for p such that $\mathbb{A}^0 \neq \emptyset$.

It is worth pointing out that since there is no notion of "active constraints" in PDIP methods, both $z^*(p)$ and $f^*(p)$ are twice continuously differentiable $\forall p \in \mathcal{P}$ when NLP (2.23) is solved with Algorithm 2. Instead of (2.24), the sensitivities are given by $\frac{\partial r_\tau}{\partial p}\frac{\mathrm{d}w}{\mathrm{d}p} + \frac{\partial r_\tau}{\partial p} = 0$, and rather than "kinks" in $z^*(p)$ and $\nabla_p f^*(p)$ at p where $\mathbb{A}^0 \neq \emptyset$, the functions are strongly non-linear in regions around such p. As illustrated in Figure 2.4, both $w(p)$ and $f^*(p)$ can therefore be made "smoother" by keeping ϵ large in Algorithm 2.

It is also worth pointing out that the derivatives of $z^*(p)$ and $f^*(p)$ are cheap to compute when SQP or PDIP algorithms are used to solve (2.23). In both cases, the factorization of $\frac{\partial r_{\mathbb{A}^+}}{\partial p}$ or $\frac{\partial r_\tau}{\partial p}$ needed to solve (2.24) is computed at the algorithms final iterate and can consequently be reused.



(a) (b)

Figure 2.4: Illustration of the functions $x^*(p)$ and $f^*(p)$ for the optimization problem solved in Figure 2.3. The parameter is the x_2-position of the unconstrained minimum of f, marked with $x^{[0]}$ in Figure 2.3. As p is increased, the optimal solution $x^*(p)$ moves and the active set changes so that $h_2(x) < 0$. The resulting kink in $x^*(p)$ is seen in (a). The parabolic lines show $x_1^*(p)$ and $f^*(p)$ obtained through Algorithm 2 with different ϵ, illustrating the lack of discontinuity in PDIP methods.

2.3.5 Distribution methods

Some NLPs can be decomposed into a set of sub-problems, which are coupled through shared variables and/or constrains. In many cases, the original problem has a hierarchical structure, such as the one shown in Figure 2.5. By utilizing this structure, some degree of parallelism is often enabled and computational savings achieved. An example is problems of the type

$$\min_{y,x} \quad \sum_{i=1}^{N} f_i(x_i, y), \tag{2.26a}$$

$$\text{s.t.} \quad g_i(x_i, y) = 0, \; h_i(x_i, y) \leq 0, \quad i = 1, \ldots, N \tag{2.26b}$$

which consists of N separate components, coupled by the *complicating variable* y, This problem can be re-written as

$$\min_{y} \quad \sum_{i=1}^{N} f_i^*(y) \tag{2.27}$$

$$f_i^*(y) = \min_{x} \quad f_i(x_i, y) \tag{2.28a}$$

$$\text{s.t.} \quad g_i(x_i, y) = 0, \tag{2.28b}$$

$$h_i(x_i, y) \leq 0, \tag{2.28c}$$

which is known as a *primal decomposition* of NLP (2.26) consisting of the master (2.27) and sub-problems (2.28). The name is due to the master problem's manipulation of the original problem's primal variables. Similar decompositions can be made when the sub-problems instead are coupled through complicating constraints on, e.g., the form $\sum_{i=1}^{N} g_i(x_i) = 0$, $\sum_{i=1}^{N} h_i(x_i) \leq 0$. In these cases the master problem instead manipulates corresponding dual variables λ, μ of the original problem, whereby such schemes are known as *dual decomposition*.

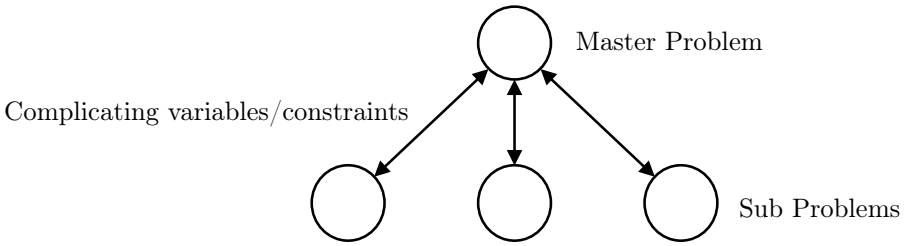


Figure 2.5: Typical structure of problems where distribution methods can be applied.

An archetypal solution procedure for the decomposed problem is

1. $\forall i$: Given y, solve (2.28) and compute derivatives of $f_i^*(y)$.
2. Use derivatives to update y towards a local minimum of (2.27). Repeat until a solution to (2.27) is found.

When a local minimizer y^* is found, it is together with the x^* obtained by solving (2.28) with y^* the solution to (2.26). Schemes for dual decomposition operate similarly.

Since $f_i^*(y)$ is the optimal value function of an optimization problem, its derivatives are obtained using the results of Section 2.3.4. Consequently, $f^*(y)$ is only piece-wise continuously differentiable when inequality constraints are present in the sub-problems. For this reason, various first order methods for non-smooth optimization are often employed on primal and dual decomposition formulations. However, Newton-type methods can be employed, provided that the step size is selected to ensure progress, and that the evaluation of derivatives at points where differentiability is lost (active set changes in the sub problem) is handled (as in, e.g., [73]). It should also be noted that when a PDIP method is used to solve (2.28), $f_i^*(y)$ is twice continuously differentiable, but possibly strongly nonlinear on subsets of its domain, which leads to difficulties for a Newton-type method.

An alternative approach

In some cases, an alternative strategy can be employed to solve (2.26). Adopting the PDIP perspective for simplicity, we note that besides evaluating the involved functions and their derivatives, the computationally expensive part of Algorithm 2 is the solution of the linear system $M(z^{[k]})\Delta z^{[k]} = -r_\tau(z^{[k]})$ (i.e. (2.20)). When this can be parallelized, savings can be achieved. Collecting the primal-dual variables of (2.27) in z_m and those of sub-problem (2.28) in $z_{s,i}$, and define $z_s = (z_{s,1}, \ldots, z_{s,N})$, (2.20) can be written as

$$\begin{bmatrix} M_m & M_{ms} \\ M_{ms}^\top & M_s \end{bmatrix} \begin{bmatrix} \Delta z_m \\ \Delta z_s \end{bmatrix} = - \begin{bmatrix} r_m \\ r_s \end{bmatrix}. \tag{2.29}$$

Using the Schur complement, (2.29) can be solved as

$$\left(M_m - M_{ms}M_s^{-1}M_{ms}^\top\right)\Delta z_m = -r_m + M_{ms}M_s^{-1}r_s, \tag{2.30}$$

$$M_s\Delta z_s = -r_s - M_{ms}^\top\Delta z_m, \tag{2.31}$$

When the sub-problems only are coupled through z_m (as in (2.26)) we have that 1) $M_{ms}M_s^{-1}M_{ms}^\top$ and $M_{ms}M_s^{-1}r_s$ are sums with terms from the different sub-problems 2) (2.31) can be solved as N independent sets of equations. This enables parallelism but avoids the non-smooth optimization of decomposition-based approaches.

2.4 Optimal control

The goal in optimal control (OC) is to find trajectories of a dynamical system such that a performance metric is optimized. The continuous-time variety of OC thus considers the optimization of an objective functional over a function space, contrary to the optimization of functions over vector spaces in the previous section. While Optimal Control Problems (OCP) can take different forms, we focus on problems of the type

$$\min_{\mathrm{u}(t)} \quad V(\mathrm{x}(t_f)) + \int_{t_0}^{t_f} l(\mathrm{x}(t),\mathrm{u}(t))\mathrm{d}t \tag{2.32a}$$

$$\text{s.t.} \quad \mathrm{x}(t_0) = \hat{x}_0 \tag{2.32b}$$

$$\dot{\mathrm{x}}(t) = f(\mathrm{x}(t),\mathrm{u}(t)), \tag{2.32c}$$

$$h(\mathrm{x}(t),\mathrm{u}(t)) \leq 0, \tag{2.32d}$$

$$h^f(\mathrm{x}(t_f)) \leq 0. \tag{2.32e}$$

Here, $\mathrm{x}(t)$,$\mathrm{u}(t)$ are the *state* and *control* trajectories of the *dynamical system* $\dot{\mathrm{x}}(t) = f(\mathrm{x}(t),\mathrm{u}(t))$ respectively, which are required to satisfy the *path constraint* $h(\mathrm{x}(t),\mathrm{u}(t)) \leq 0$ for $t \in [t_0, t_f]$, and the *terminal constraint* $h^f(\mathrm{x}(t_f)) \leq 0$, starting from the *initial state* $\hat{x}_0$, where t_0 and t_f are the initial and *final time*s respectively.

2.4.1 Solving OCPs

The classical approach to solve (2.32) has conceptual similarities with the methods of Section 2.3, in that a solution is found through the necessary conditions for optimality. These give a multi-point boundary value problem (BVP), which is solved to obtain the optimal state and control trajectories $\mathrm{x}^*(t)$,$\mathrm{u}^*(t)$ [74]. For all but a class of simple problems, the BVP must be discretized and solved approximately (e.g., with a shooting-method). This is

in many cases difficult, and is complicated further by the presence of (2.32d) and (2.32e). Since an expression for the optimal control first is found and thereafter is solved through discretization, this is sometimes called the *"first optimize then discretize"* or *indirect* approach to Optimal Control [75].

Direct Optimal Control

An alternative to finding $\mathrm{u}^*(t)$ through the optimality conditions of OCP (2.32) is to instead solve a "discretized", finite-dimensional problem whose optimal solution approximates $\mathrm{u}^*(t)$. This method is known as *Direct Optimal Control (DOC)* and is sometimes referred to as the *"first discretize then optimize"* approach. The construction of the "discretized" problem is known as the *transcription* of OCP (2.32), and is often performed by assuming a parametrized form of the input, i.e. $\mathrm{u}(t) := \mathrm{u}(t, u)$ with parameter vector u. Since $\mathrm{x}(t)$ is a function of u(t), the OCP (2.32) can thereby be cast as an optimization problem in u. Since explicit solutions to (2.32c) in many cases are hard to find, transcription typically involves numerical integration (using e.g. explicit or implicit Runge-Kutta or collocation-based integrators [76])

A DOC formulation of Problem (2.32) A common input parametrization is piece-wise constant $\mathrm{u}(t)$ on a uniform time grid $t_0, t_1, \ldots, t_N$, i.e. $\mathrm{u}(t, u) = u_k$, $t \in [t_k, t_{k+1}[$ with $u = (u_0, \ldots, u_{N-1})$ and $t_k = k\Delta t$. This can be related to $\mathrm{x}(t)$ using, e.g., multiple shooting [77], i.e. by finding $x = (x_1, \ldots, x_N)$ such that

$$x_{k+1} - F_k(x_k, u_k) = 0, \quad k = 0, \ldots, N-1, \tag{2.33}$$

where $x_0 = \hat{x}_0$ and $F_k(x_k, u_k)$ denotes the (numerical) solution to (2.32c) at t_{k+1}, when $\mathrm{x}(t_k) = x_k$ and $\mathrm{u}(t) = u_k$, $t \in [t_k, t_{k+1}[$. If the path constraints (2.32d) only are enforced at $t_0, \ldots, t_k$, a transcription of OCP (2.32) reads

$$\begin{aligned}
\min_{x,u} \quad & V(x_N) + \sum_{k=0}^{N-1} \ell_k(x_k, u_k) && \text{(2.34a)}\\
\text{s.t.} \quad & x_0 = \hat{x}_0, && \text{(2.34b)}\\
& x_{k+1} - F_k(x_k, u_k) = 0, \quad k = 1, \ldots, N-1 && \text{(2.34c)}\\
& h(x_k, u_k) \leq 0, && \text{(2.34d)}\\
& h^f(x_N) \leq 0, && \text{(2.34e)}
\end{aligned}$$

where $\ell_k(x_k, u_k)$ is the (numerical) integration of $l(x(t), u(t))$ over $[t_k, t_{k+1}]$. Problem (2.34) is thus an optimization problem in x, u on the form (2.2), which can be classified as discussed in Section 2.2.1. When (2.34) is an NLP,

the algorithms discussed in Section 2.3 can be employed. A special case is quadratic positive-definite l, linear f and affine h. In this case, (2.32c) admits an exact solution and NLP (2.34) is a convex QP.

2.4.2 Model predictive control

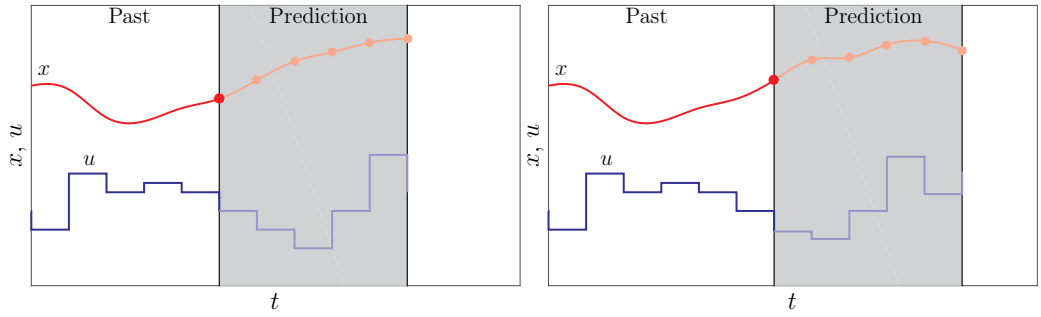
While the OCP (2.32) gives optimal state and control trajectories for a dynamical system described by (2.32c), applying $\mathrm{u}^*(t)$ directly to a real system often leads to poor performance. The reason is that although $\mathrm{u}^*(t)$ is applied, the actual system trajectory $\mathrm{x}^a(t)$ will in general be different from $\mathrm{x}^*(t)$. This occurs when, e.g., the actual system differs from the model (2.32c), when the system state ($\hat{x}_0$) is known with uncertainty or when the system is subjected to external perturbations. This serves as a motivation for Model Predictive Control (MPC), which is a practical realization of optimal control that includes a feedback component.

Rather than applying $\mathrm{u}^*(t)$ in open-loop, in MPC, the (discretized) OCP is resolved periodically to update $\mathrm{u}^*(t)$, in a procedure which can be summarized as follows

1. At time t, estimate the current system state $\hat{x}$.
2. Solve OCP (2.34) with initial state set to $\hat{x}$ for the optimal control u^*.
3. Apply the first part of u^* (i.e. u_0^*) to the plant.
4. Wait until $t + \Delta t$, go to 1.

Through the solution of OCP (2.34), an MPC consequently uses the model (2.34c) to form a prediction of the optimal action to take over a future time window, which results in the predicted optimal control commands u^*. However, since the prediction will be inaccurate, only the first part of u^* is applied, whereafter the system response is evaluated, and the process repeated. The latter introduces feedback, and allows MPC to compensate for model/plant mismatches and perturbations. The time window is of size $N\Delta t$ and is commonly referred to as the *prediction horizon*. The prediction horizon length N is typically held constant so that its endpoint moves forward in time together with the actual system time. Due to this MPC is also known as *receding horizon control*. An illustration of the operating principle of MPC is given in Figure 2.6.

MPC comes in a number of different varieties, and can be classified based on the characteristics of ℓ_k, V, F, h, h^f. Perhaps the most common is *linear MPC*, where F is linear, h, h^f are affine and ℓ_k, V are convex-quadratic so that (2.34) is a QP. However, *Nonlinear MPC* (NMPC) exists, where some functions are nonlinear and (2.34) is an NLP. Problems where ℓ_k, V are



(a) at t_k (b) at t_{k+1}

Figure 2.6: Illustration of the receding horizon principle. The current state is marked with a red dot, state trajectories in red, control inputs in blue. The prediction window (gray) is pushed forward in time as the system evolves.

indefinite provides a separate class, which sometimes is denoted *Economic MPC*, different from *Tracking MPC* with positive-definite ℓ_k, V.

Stability of tracking MPC is established using Lyapunov arguments [78] relying on the positive definiteness of ℓ_k, V, whereas Economic MPC require further assumptions and is more involved [79].

Another important property of MPC is *persistent feasibility* . Letting $\mathcal{F}$ be the set of $\hat{x}_0$ such that (2.34) has a solution, and $\kappa^*(x)$ be the optimal u_0^* for $\hat{x}_0 = x$, we define

Definition 2.3 (Persistent Feasibility)**.** *The MPC based on* (2.34) *is persistently feasible if* $F(x, \kappa^*(x)) \in \mathcal{F}$, $\forall x \in \mathcal{F}$.

That is, a persistently feasible MPC does not take the system to a state x where no control commands exists that satisfy (2.34b)-(2.34e).

Chapter 3

Modeling Intersection Scenarios

In this chapter, we discuss modeling aspects of the intersection coordination problem, with focus on vehicle motion models and conditions for collision avoidance.

3.1 Motion models

There are a number of motion models that describes the dynamics of ground vehicles with different levels of accuracy. The more detailed combines lateral- and longitudinal motion with models of the power-train, suspension and tires, and provides high accuracy under a wide range of conditions [80, 81].

However, since intersections normally are placed on roads with low curvature, and racing-like driving often is not the objective, the lateral vehicle dynamics are of less importance in the problem at hand. It is therefore common to decouple the lateral- and longitudinal dynamics in the intersection coordination context. In particular, in thesis and most other work on the topic (e.g. [29,33,40–52,59–61,63–68]), the following assumption is employed:

Assumption 3.1 (Vehicles on Rails)**.** *The vehicles in the vicinity of the intersection move along predefined paths and do not change lanes. The lateral tracking of the path is handled by a low-level controller.*

As illustrated in Figure 3.1, the vehicles are consequently treated as if they were on rails. This enables simplified models that only describe the motion of the vehicles along their paths. The assumption is not restrictive, as the dominant mode of motion for a vehicle typically is oriented with the road.

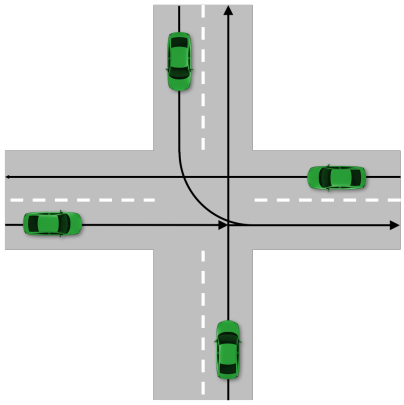


Figure 3.1: Illustration of the Assumption 3.1. The vehicles are assumed to move along the paths illustrated by the lines, where the arrows indicate the direction of motion.

3.1.1 General form

With Assumption 3.1, the state $\mathrm{x}_i(t) : \mathbb{R} \mapsto \mathbb{R}^{n_i}$ of the vehicle with index i is such that $\mathrm{x}_i(t) = (\mathrm{p}_i(t), \mathrm{v}_i(t), \tilde{\mathrm{x}}_i(t))$, where $\mathrm{p}_i(t)$ is the scalar position of the vehicle's geometrical center on its path, $\mathrm{v}_i(t)$ the scalar velocity along the path and $\tilde{\mathrm{x}}_i(t)$ other states (e.g. acceleration and/or internal states of the power-train). Using the control function $\mathrm{u}_i(t) : \mathbb{R} \mapsto \mathbb{R}^{m_i}$, the vehicle motion is described by a constrained ordinary differential equation (ODE)

$$\dot{\mathrm{x}}_i(t) = f_i(\mathrm{x}_i(t), \mathrm{u}_i(t)), \tag{3.1a}$$

$$0 \geq h_i(\mathrm{x}_i(t), \mathrm{u}_i(t)), \tag{3.1b}$$

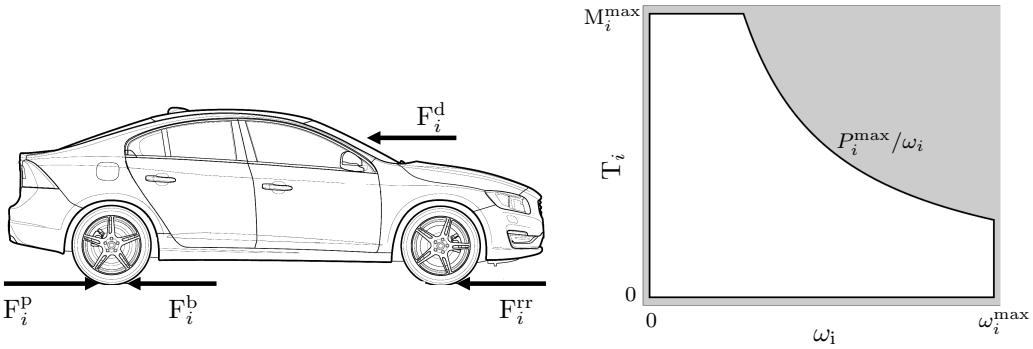
where both $f_i : \mathbb{R}^{n_i \times m_i} \mapsto \mathbb{R}^{n_i}$ and $h_i : \mathbb{R}^{n_i \times m_i} \mapsto \mathbb{R}^{c_i}$ are continuously differentiable. As in most other work on the topic, (3.1) is assumed to be such that $\mathrm{v}_i(t) \geq 0, \forall t$, i.e., such that no vehicles ever reverses.

3.1.2 Dynamical models

As illustrated in Figure 3.2(a), models on the form (3.1) arise from the application of Newtons second law to the vehicle in the longitudinal direction. In particular, (3.1a) then includes

$$\dot{\mathrm{p}}_i(t) = \mathrm{v}_i(t), \quad \dot{\mathrm{v}}_i(t) = \frac{1}{m_i}(\mathrm{F}_i^{\mathrm{c}}(\mathrm{u}(t), \mathrm{x}(t)) + \mathrm{F}_i^{\mathrm{r}}(\mathrm{x}(t))), \tag{3.2}$$

where m_i is the vehicle mass, $\mathrm{F}_i^{\mathrm{c}}(\mathrm{u}(t), \mathrm{x}(t))$ are the forces which are manipulable through the control input and $\mathrm{F}_i^{\mathrm{r}}(\mathrm{x}(t))$ are the other forces acting on the vehicle. The latter is often a sum of resistive forces, which typically includes the air-drag $\mathrm{F}_i^{\mathrm{d}}(\mathrm{x}_i(t))$ and rolling resistance $\mathrm{F}_i^{\mathrm{rr}}$. Even though not considered in this thesis, the effects of gravity on non-flat surfaces can be



(a) Free-body diagram of a vehicle on level ground.

(b) Illustration of the operating points (white) of an asynchronous electric motor

Figure 3.2: Modeling illustrations

included as $\mathrm{F}_i^{\mathrm{g}}(\mathrm{x}_i(t)) = -m_i g \cos(\alpha(\mathrm{p}_i(t)))$, where $\alpha(s)$ is the slope of the road at position s and g is the acceleration due to gravity.

The controllable forces are those generated by the propulsion system, $\mathrm{F}_i^{\mathrm{p}}(\mathrm{u}_i(t), \mathrm{x}_i(t))$, and those generated from the braking system, $\mathrm{F}_i^{\mathrm{b}}(\mathrm{u}_i(t), \mathrm{x}_i(t))$ [81]. Both typically have internal dynamics, but these are often fast enough to be ignored in this application. However, there are often relevant limitations, such as a maximum motor torque and brake force. These are included in constraints (3.1b).

Vehicles with electric power-trains

While vehicles with all types of power-trains could be considered, Electric Vehicles (EV) are of particular interest, since these are expected to dominate future vehicle fleets [82]. For EVs, the propulsive force is often $\mathrm{F}_i^{\mathrm{p}}(\mathrm{u}_i(t), \mathrm{x}_i(t)) = G_i \mathrm{M}_i(\mathrm{x}_i(t), \mathrm{u}_i(t))/r_{\mathrm{w},i}$ where G_i is the (typically fixed) total gear ratio, $r_{\mathrm{w},i}$ the wheel radius and $\mathrm{M}_i(\mathrm{x}_i(t), \mathrm{u}_i(t))$ the torque generated by the electric motor. The latter is constrained by limitations on the power electronics and mechanical subsystems (i.e. bearings and axles). A simplified representation of these constraints for asynchronous electric motors is shown in Figure 3.2(b) and expressed as

$$\mathrm{M}_i(\mathrm{x}_i(t), \mathrm{u}_i(t)) \leq \min(\mathrm{M}_i^{\max}, \mathrm{P}_i^{\max}/\omega_i(t)), \tag{3.3a}$$

$$\omega_i(t) \leq \omega_i^{\max}, \tag{3.3b}$$

where $\mathrm{P}_i^{\max}$, $\mathrm{M}_i^{\max}$ and $\omega_i^{\max}$ are the maximum power, torque and motor speed respectively, with $\omega_i = G\mathrm{v}_i(t)/r_{\mathrm{w},i}$ [81].

The dynamics of an EV's power-train are typically fast enough to be ignored, due to which M_i can be considered a directly manipulable input.

A simplified model of the EV's motion is thus (3.2), with state $\mathrm{x}_i(t) = (\mathrm{p}_i(t), \mathrm{v}_i(t))$ and control $\mathrm{u}_i(t) = (\mathrm{M}_i(t), \mathrm{F}_i^{\mathrm{b}}(t))$.

Vehicles with lower-level controllers

A different type of models can be used when lower level, power-train controllers are present. An example which is often used in platooning and has been proposed for use the intersection coordination context [65], is

$$\dot{\mathrm{p}}_i(t) = \mathrm{v}_i(t), \quad \dot{\mathrm{v}}_i(t) = \mathrm{a}_i(t), \quad \dot{\mathrm{a}}_i(t) = \frac{1}{\tau_i}\left(\mathrm{a}_i^r(t) - \mathrm{a}_i(t)\right), \tag{3.4}$$

where τ_i is a constant, $\mathrm{a}_i(t)$ the vehicle acceleration, and $\mathrm{a}_i^r(t)$ is the *commanded* acceleration, which is the control input.

Kinematic models

An alternative often found in the literature (e.g. [33, 40–49, 51, 52, 59–61, 63, 64, 68]) is to only consider kinematics and model the vehicle through a chain of integrators where the (scalar) input $u_i(t)$ enters in the end. A particularly common choice is the Double Integrator (DI) model

$$\dot{\mathrm{p}}_i(t) = \mathrm{v}_i(t), \quad \dot{\mathrm{v}}_i(t) = \mathrm{u}_i(t), \tag{3.5}$$

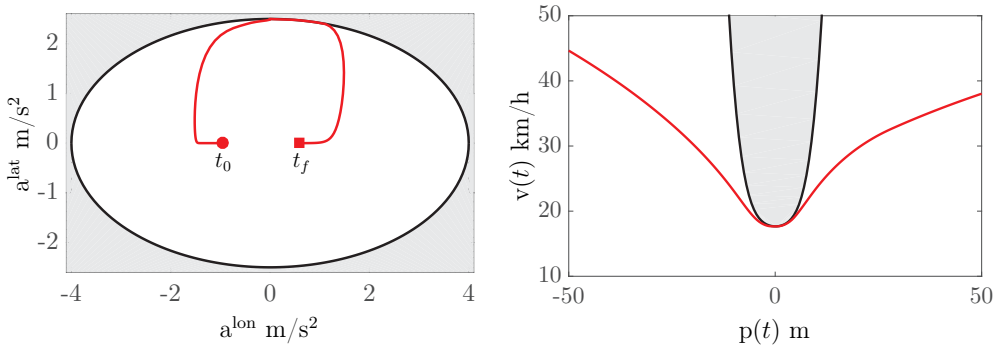
with bounds $\mathrm{v}_i^{\min} \leq \mathrm{v}_i(t) \leq \mathrm{v}_i^{\max}$ and $a_i^{\min} \leq \mathrm{u}_i(t) \leq a_i^{\max}$.

3.1.3 Motion on curved paths

While vehicles that goes straight through the intersection move on paths with small curvature, this is not the case for vehicles that perform left and right turns. A turning vehicle will therefore be subjected to non-negligible lateral forces, which must be constrained to lie in an acceptable range. Ultimately, this can be derived from the conditions under which the tires adhere to the road surface, but even tighter bounds results when passenger comfort is considered. However, since the one-dimensional models discussed above does not include lateral motion, a strategy that enforces such constraints indirectly must be employed.

To this end, we proposed an extension in [83]. In particular, since the path that a vehicle traverses is fixed and known, the curvature $\kappa_i(s)$ is available at every point s along the path. When a vehicle follows the path exactly it will thus experience the lateral acceleration

$$\mathrm{a}_i^{\mathrm{lat}}(t) = \kappa_i(\mathrm{p}_i(t))\mathrm{v}_i(t)^2. \tag{3.6}$$



(a) Acceleration. (b) Velocity

Figure 3.3: Example with constraint by (3.7).

Since both passenger comfort and the tire adhesion depend on the force *vectors* that act on the car, we suggested the following condition in [83]

$$\left(\frac{\dot{\mathrm{v}}_i(t)}{a_i^{\mathrm{lon,max}}}\right)^2 + \left(\frac{\kappa_i(\mathrm{p}_i(t))\mathrm{v}_i(t)^2}{a_i^{\mathrm{lat,max}}}\right)^2 \leq 1, \tag{3.7}$$

where $a_i^{\mathrm{lat,max}}$ and $a_i^{\mathrm{lon,max}}$ are the lateral and longitudinal acceleration limits respectively, and $\dot{\mathrm{v}}_i(t)$ can be evaluated through the right-hand side of (3.1a).

An example of $(\mathrm{a}_i^{\mathrm{lon}}(t), \mathrm{a}_i^{\mathrm{lat}}(t))$ and $(\mathrm{p}_i(t), \mathrm{v}_i(t))$ of a turning vehicle under constraint (3.7) is provided in Figure 3.3. Note how the vehicle is forced to slow down when the turn is negotiated.

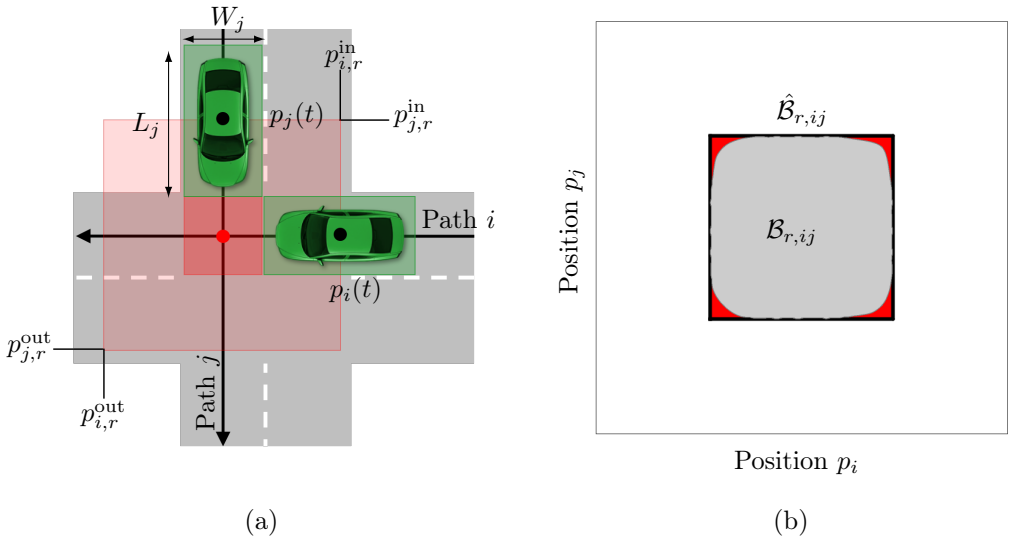
3.2 Conditions for collision avoidance

Two types of collisions can occur between the vehicles at an intersection: 1) side collisions between vehicles on different lanes, and 2) rear-end collisions between vehicles on the same lane. In this section, we discuss how to form conditions which, if satisfied, ensure that collisions are avoided.

We first note that in general, a collision between two vehicles i and j is characterized by the occupation of the same space by parts of their respective geometries $\mathcal{G}_i(\mathrm{p}_i(t))$ and $\mathcal{G}_j(\mathrm{p}_j(t))$. Consequently, the least restrictive condition for any type of Collision Avoidance (CA) is that

$$(\mathrm{x}_i(t), \mathrm{x}_j(t)) \notin \mathcal{B}_{r,ij} = \{(x_i, x_j) \mid \mathcal{G}_i(p_i) \cap \mathcal{G}_j(p_j) \neq \emptyset\}, \tag{3.8}$$

for all vehicle pairs (i, j). For detailed models with a two-dimensional position coordinate and orientation, this corresponds to the avoidance of a complex object in the 6-dimensional joint configuration space of the two vehicles. However, Assumption 3.1 enables slightly conservative conditions that are significantly easier to handle.



(a) (b)

Figure 3.4: Illustration of the SICA condition (3.9). In (a), $\hat{\mathcal{G}}_i(p_i)$ are shown in green, the CZ in light red and the dark red is the area where the outer approximations can overlap. In (b), the black square marks $\hat{\mathcal{B}}_{r,ij}$, the gray area $\mathcal{B}_{r,ij}$ and the red areas are configurations that are unnecessarily excluded.

3.2.1 Side collision avoidance (SICA)

Side collisions can only occur in areas around the points where the paths of two vehicles intersect. We denote these areas *conflict zones* (CZ), and note that more than one vehicle pair (i, j) can collide at the same CZ. Rather than deriving a CA condition from the complicated vehicle geometry directly, a slightly conservative condition can be obtained using rectangular outer approximations $\hat{\mathcal{G}}_i(\mathrm{p}_i(t)) \supseteq \mathcal{G}_i(\mathrm{p}_i(t))$. In particular, due to Assumption 3.1, a collision corresponds to a set of positions in their two-dimensional joint configuration space. A requirement which ensures that $\mathcal{G}_i(\mathrm{p}_i(t)) \cap \mathcal{G}_j(\mathrm{p}_j(t)) \neq \emptyset$ at the r:th CZ can thereby be formulated as

$$(\mathrm{x}_i(t), \mathrm{x}_j(t)) \notin \hat{\mathcal{B}}_{r,ij} = \left\{(x_i, x_j) \mid \ p_i \in [p^{\text{in}}_{r,i}, p^{\text{out}}_{r,i}],\ p_j \in [p^{\text{in}}_{r,j}, p^{\text{out}}_{r,j}]\right\}, \quad (3.9)$$

where $p^{\text{in}}_{r,i}$ and $p^{\text{out}}_{r,i}$ are the first and last positions p_i for which $\hat{\mathcal{G}}_i(p_i) \cap \hat{\mathcal{G}}_j(p_j) \neq \emptyset$, $\forall p_j$ at CZ r. An illustration of (3.9) is provided in Figure 3.4(a) and a visualization of the conservativeness induced with the tightest approximation $\hat{\mathcal{G}}_i(p_i)$ is given in Figure 3.4(b). As the latter highlights, little is lost when the vehicle geometry is roughly rectangular. Several algorithms in the literature use variations of (3.9) to represent collisions [25, 61].

A reformulation of (3.9) that is sometimes used [36, 41, 84] employs auxiliary variables that describe the time of entry ($t^{\text{in}}_{r,i}$) and departure ($t^{\text{out}}_{r,i}$)

of CZ r, defined implicitly through

$$\mathrm{p}_i(t_{i,r}^{\mathrm{in}}) = p_{r,i}^{\mathrm{in}}, \quad \text{and} \quad \mathrm{p}_i(t_{i,r}^{\mathrm{out}}) = p_{r,i}^{\mathrm{out}}.^{1} \tag{3.10}$$

Since by assumption $\mathrm{v}_i(t) \geq 0$, $p_{r,i}^{\mathrm{in}} < p_{r,i}^{\mathrm{out}} \Rightarrow t_{r,i}^{\mathrm{in}} < t_{r,i}^{\mathrm{out}}$ and therefore $\left(t_{r,i}^{\mathrm{out}} \leq t_{r,j}^{\mathrm{in}}\right) \Rightarrow \neg \left(t_{r,j}^{\mathrm{out}} \leq t_{r,i}^{\mathrm{in}}\right)$, the following statement is equivalent to (3.9) and assures Side Collision Avoidance (SICA):

$$\left(t_{r,i}^{\mathrm{out}} \leq t_{r,j}^{\mathrm{in}}\right) \vee \left(t_{r,j}^{\mathrm{out}} \leq t_{r,i}^{\mathrm{in}}\right). \tag{3.11}$$

In words, (3.11) states that vehicle j must leave CZ r before vehicle i enters or vice-versa.

Adding Margins

It is often desirable to introduce margins to the SICA conditions. For the conditions derived from outer approximations of the vehicle geometry detailed above, this can be done by either enlarging the CZ definition as $[p_{r,i}^{\mathrm{in}} - \delta p_{r,i}, p_{r,i}^{\mathrm{out}} + \delta p_{r,i}]$, $\delta p_{r,i} > 0$, or by introducing time margins in (3.11), so that $\left(t_{r,i}^{\mathrm{out}} + \delta t_{ij} \leq t_{r,j}^{\mathrm{in}}\right) \vee \left(t_{r,j}^{\mathrm{out}} + \delta t_{ij} \leq t_{r,i}^{\mathrm{in}}\right)$, $\delta t_{ij} > 0$.

3.2.2 Rear-End Collision Avoidance (RECA)

Rear end collisions can only occur between two adjacent vehicles on the lane, which translates to simple conditions on the vehicle positions due to Assumption 3.1. Denoting the length of vehicle i as L_i and $\delta_{ij} = L_l/2 + L_j/2$, a necessary condition for Rear-End Collision Avoidance (RECA) is

$$\mathrm{p}_i(t) + \delta_{ij} \leq \mathrm{p}_j(t), \tag{3.12}$$

when vehicle i is behind vehicle j. Condition (3.12) could be extended to include conservative (and more practical) distance keeping policies, e.g., fixed spacing policies with $\delta_{ij} > L_i/2 + L_j/2$, or speed-dependent policies $\delta_{ij} = L_i/2 + L_i/2 + \gamma_i \mathrm{v}_i(t)$, with time-headway γ_i.

Rear-end collisions with turning vehicles

Rear end collisions between vehicles that take turns and the adjacent vehicles on their origin and destination lanes require additional descriptions under Assumption 3.1. In particular, RECA with respect to vehicles on the origin lane should only be enforced until the turning vehicle leaves the origin lane.

[1] If $\mathrm{v}(t_{i,r}^{\mathrm{in}}) = 0$, $t_{i,r}^{\mathrm{in}}$ is not uniquely defined by $\mathrm{p}_i(t_{i,r}^{\mathrm{in}}) = p_{r,i}^{\mathrm{in}}$. A practical remedy is to instead use the definition $t_{r,i}^{\mathrm{in}} = \min t$ s.t. $\mathrm{p}_i(t_{i,r}^{\mathrm{in}}) = p_{r,i}^{\mathrm{in}}$. Since $\mathrm{v}(t_{i,r}^{\mathrm{in}}) = 0$ rarely will be encountered in practice, this is avoided for ease of presentation.

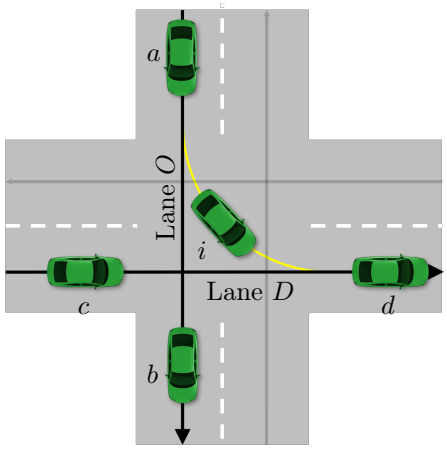


Figure 3.5: Vehicles in (3.13) O and D denotes the origin and destination lanes, respectively.

Similarly, RECA with respect to vehicles on the destination lane should only be enforced from the time that the turning vehicle enters the destination lane. We formalize these requirements as

$$\mathrm{p}_i(t) + \delta_{ib} \leq \mathrm{p}_b(t), \qquad \mathrm{p}_a(t) + \delta_{ai} \leq \mathrm{p}_i(t), \qquad t \leq t_i^O \tag{3.13a}$$

$$\mathrm{p}_i(t) + \delta_{id} \leq \mathrm{p}_d(t), \qquad \mathrm{p}_c(t) + \delta_{ci} \leq \mathrm{p}_i(t), \qquad t \geq t_i^D, \tag{3.13b}$$

where vehicles index a, b, c, d relate to vehicle i as shown in Figure 3.5 and times t_i^O and t_i^D are defined as

$$\mathrm{p}_i(t_i^O) = p_i^O, \tag{3.14a}$$

$$\mathrm{p}_i(t_i^D) = p_i^D, \tag{3.14b}$$

Here, p_i^O is the first position on the path of the turning vehicles such that $\hat{\mathcal{G}}_i(p_i^O) \cap \hat{\mathcal{G}}_a(p_a) \neq \emptyset$, $\hat{\mathcal{G}}_i(p_i^O) \cap \hat{\mathcal{G}}_b(p_b) \neq \emptyset \ \forall p_a, p_b$, and p_i^D is the last position after which $\hat{\mathcal{G}}_i(p_i^D) \cap \hat{\mathcal{G}}_c(p_c) \neq \emptyset$, $\hat{\mathcal{G}}_i(p_i^D) \cap \hat{\mathcal{G}}_d(p_d) \neq \emptyset \ \forall p_c, p_d$.

3.3 Summary

In this chapter discussed modeling aspects of the intersection coordination problem. We presented the Vehicles on Rails assumption the simplified motion models that it enable, and derived slightly conservative conditions for side and rear-end collision avoidance. For turning vehicles, we discussed how the lateral acceleration can be limited to promote comfort and safety, and how their more complicated collision avoidance collisions can be described. In the next chapter, we employ the presented models and collision avoidance conditions in optimal control formulations of the intersection coordination problem.

Chapter 4

Optimal Coordination

In this chapter we introduce an Optimal Control (OC) formulation of the coordination problem for automated vehicles at intersections. Possible performance objectives are discussed in Section 4.1 and a general OC formulation is presented in Section 4.2. The transcription of the infinite-dimensional OCP to a finite-dimensional optimization problem is described in Section 4.3, and possible solution strategies are discussed Section 4.4.

4.1 Performance objectives

Several performance objectives have been proposed in the intersection coordination context. Common choices are energy/fuel consumption minimization [59, 68] or traffic flow maximization/travel time minimization [36, 43]. These are often motivated by the needs of the traffic system (e.g. to avoid congestion) and a reduced economic/environmental impact. Objectives that include the interest of the passengers have also been proposed, as in e.g. [49], where risk-minimization was considered or [83] that factored in comfort. In optimal control formulations, the objectives are typically described using functionals of $\mathrm{x}_i(t)$, $\mathrm{u}_i(t)$ on the form

$$J_i(\mathrm{x}_i(t), \mathrm{u}_i(t)) = V_i^{\mathrm{f}}(\mathrm{x}(t_f)) + \int_0^{t_f} l_i(\mathrm{x}_i(t), \mathrm{u}_i(t))\mathrm{d}t. \tag{4.1}$$

where, $V_i^{\mathrm{f}}(\mathrm{x}(t_f))$ and $l_i(\mathrm{x}_i(t), \mathrm{u}_i(t))$ describe the performance metric to optimize. In particular, l_i is often a weighted combination of different objectives

$$l_i(\mathrm{x}_i(t), \mathrm{u}_i(t)) = \sum_{y=1}^{Y} \mathrm{q}_{i,y} l_i^y(\mathrm{x}_i(t), \mathrm{u}_i(t)), \tag{4.2}$$

where $l_i^y(\mathrm{x}_i(t), \mathrm{u}_i(t))$ is a "sub-objective" and $\mathrm{q}_{i,y} > 0$ is a scalar weight. Selecting the net power delivered to the vehicle, $\mathrm{P}_i(x_i(t), u_i(t))$, as one sub-

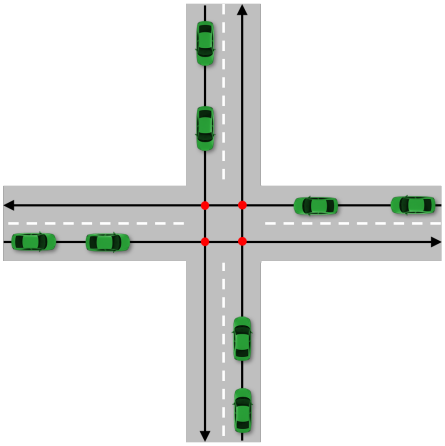


Figure 4.1: Illustration of the scenarios considered

objective captures energy consumption, setting another to $-\mathrm{v}(t)$) captures travel-time. The weights q_i^y determines the trade-off between the two. Comfort measures can be included similarly, as in, e.g., [83], where we proposed to set one l_i^y to $||\mathrm{a}_i(t)||^2$, where $\mathrm{a}_i(t)$ is the acceleration vector.

A common alternative (e.g. [59, 63, 65, 68]), is to use quadratic functions

$$l_i(\mathrm{x}_i(t), \mathrm{u}_i(t)) = \Delta\mathrm{x}_i(t)^\top Q_i \Delta\mathrm{x}_i(t) + \Delta\mathrm{u}_i(t)^\top R_i \Delta\mathrm{x}_i(t), \tag{4.3}$$

with $\Delta\mathrm{x}_i(t) = \mathrm{x}_i(t) - x_i^{\mathrm{ref}}$, $\Delta\mathrm{u}_i(t) = \mathrm{u}_i(t) - u_i^{\mathrm{ref}}$, where x_i^{ref} and u_i^{ref} are reference values for $\mathrm{x}_i(t)$ and $\mathrm{u}_i(t)$ respectively, and $Q_i \succeq 0$, $R_i \succ 0$ are weighting matrices. While (4.3) doesn't necessarily correspond to a particular quantity of interest, Q_i and R_i can be selected to achieve good performance in other metrics indirectly. It is for instance argued in [84] that a quadratic penalization of $a_i^{\mathrm{lon}}(t)$ minimizes transient operation of internal combustion engines, and therefore also the fuel consumed. In this context, we proposed the use of an automated tuning procedure from the literature in Paper E, where Q_i and R_i are selected so that the optimal solution is a first-order approximation of that obtained with a general objective.

4.2 A general optimal control formulation

To state the coordination of automated vehicles within the optimal control framework, we make the following assumption:

Assumption 4.1 (Full Automation)**.** *There are no non-cooperative entities present in the scenario.*

That is, we do not consider scenarios with legacy vehicles, pedestrians or bicyclists. The assumption is restrictive and limits the applicability

to traffic scenarios in a distant future, but is standard in the literature (see e.g. [28, 33, 40, 59, 68]). The optimal coordination of N vehicles at an intersection such as that shown in Figure 4.1 is the solution to the following OCP

$$\min_{\mathrm{x}(t),\mathrm{u}(t),T} \quad \sum_{i=1}^{N} J_i(\mathrm{x}_i(t), \mathrm{u}_i(t)) \tag{4.4a}$$

$$\text{s.t.} \quad \mathrm{x}_i(0) = \hat{x}_{i,0}, \qquad i \in \mathcal{N}, \tag{4.4b}$$

$$\dot{\mathrm{x}}_i(t) = f_i(\mathrm{x}_i(t), \mathrm{u}_i(t)), \qquad i \in \mathcal{N}, \tag{4.4c}$$

$$h_i(\mathrm{x}_i(t), \mathrm{u}_i(t)) \leq 0, \qquad i \in \mathcal{N}, \tag{4.4d}$$

$$\mathrm{p}_i(t_{i,r}^{\mathrm{in}}) = p_{r,i}^{\mathrm{in}},\ \mathrm{p}_i(t_{i,r}^{\mathrm{out}}) = p_{r,i}^{\mathrm{out}}, \qquad i \in \mathcal{N},\ r \in \mathcal{R}_i, \tag{4.4e}$$

$$\left(t_{r,i}^{\mathrm{out}} \leq t_{r,j}^{\mathrm{in}}\right) \vee \left(t_{r,j}^{\mathrm{out}} \leq t_{r,i}^{\mathrm{in}}\right), \qquad (i,j,r) \in \mathcal{C}_S, \tag{4.4f}$$

$$\mathrm{p}_i(t) + \delta_{ij} \leq \mathrm{p}_j(t), \qquad (i,j) \in \mathcal{C}_R, \tag{4.4g}$$

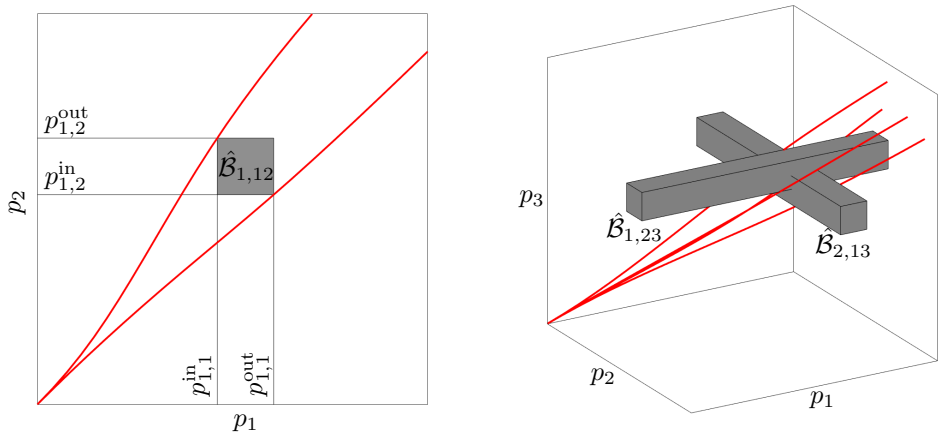
where, $\mathcal{N} = \{1, \ldots, N\}$, $\mathrm{x}(t) = (\mathrm{x}_1(t), \ldots, \mathrm{x}_N(t))$, $\mathrm{u}(t) = (\mathrm{u}_1(t), \ldots, \mathrm{u}_N(t))$, $T = (T_1, \ldots, T_N)$ and T_i collects $T_{r,i} = (t_{r,i}^{\mathrm{in}}, t_{r,i}^{\mathrm{out}})$, $\forall r \in \mathcal{R}_i$, with $\mathcal{R}_i$ being the CZ crossed by vehicle i. Moreover, $\mathcal{C}_S$ collects all vehicle-pairs and conflict zones (i,j,r) where side-collisions are possible and $\mathcal{C}_R$ collects all vehicle pairs (i,j) where rear-end collisions are possible.

Remark 4.1. *While OCP* (4.4) *is derived for a scenario with one lane in each direction, multi-lane problems can be considered with the same formalism. To ease the presentation we have also excluded vehicles that turns, but emphasize that those can be included as discussed in Chapter 3.*

Solving OCP (4.4) involves deciding the *crossing order*, i.e., if vehicle i should cross CZ r before vehicle j or vice-versa, for all $(i,j,r) \in \mathcal{C}_S$. An illustration of the configuration space and example of collision free trajectories for different crossing order in a two and three-vehicle scenario is given in Figure 4.2. As can be seen, the different crossing order corresponds to different paths around the configuration space object that describes collisions. While two crossing orders are possible in the two-vehicle case, the number grows to four for three vehicles and to 16 for four vehicles. While the exact number of orders for N vehicles is dependent on the intersection layout and the distribution of vehicles on different lanes, it is clear that the solution space exhibit rapid growth in N.

4.3 A practical optimal control formulation

To obtain a more practical representation, we use a direct optimal control reformulation of (4.4). This approach is taken in many OC-based schemes



(a) 2 Vehicles (b) 3 Vehicles

Figure 4.2: Illustrations of the configuration space of two simple scenarios. The gray objects show $\hat{\mathcal{B}}_{r,ij}$, and the red lines collision free trajectories for different crossing orders.

on intersection coordination (see e.g. [59, 61, 62, 65]), but it should be noted that indirect methods sometimes are employed as well [68, 84].

We use a piece wise constant input parametrization, i.e. $\mathrm{u}_i(t) = u_{i,k}$, $t \in [t_k, t_{k+1}[$, for $t_k = k\Delta t$ and $u_{i,k} \in \mathbb{R}^{m_i}$, which enables the recursion

$$x_{i,k+1} := \mathrm{x}_i(t_{k+1}) = F_i(x_{i,k}, u_{i,k}, \Delta t), \tag{4.5}$$

from an initial condition $\mathrm{x}_i(0) = \hat{x}_{i,0}$. Here, $F_i(x_{i,k}, u_{i,k}, \Delta t)$ denotes the solution to $\dot{\mathrm{x}}_i(t) = f_i(\mathrm{x}_i(t), \mathrm{u}_i(t))$ at $t = t_k + \Delta t$, with $\mathrm{x}_i(t_k) = x_{i,k}$ and $\mathrm{u}_i(t) = u_{i,k}$, $t \in [t_k, t_k + \Delta t]$. The state at an arbitrary time t can thereby be written $\mathrm{x}_i(t) = F_i(x_{i,k}, u_{i,k}, \delta t)$, with $\delta t = t - t_k$ and $k = \lfloor t/\Delta t \rfloor$, whereby $\mathrm{x}_i(t)$ and $\mathrm{u}_i(t)$ are completely described for all $t \in [0, K\Delta t]$ by the vectors $x_i = (x_{i,0}, \ldots, x_{i,K})$ and $u_i = (u_{i,0}, \ldots, u_{i,K-1})$. In particular, letting $w_i = (x_i, u_i)$, we express the position $\mathrm{p}_i(t)$ at an arbitrary time t as

$$p_i(t, w_i) = F_{i,p}(x_{i,k}, u_{i,k}, t - t_k), \;\; k = \lfloor t/\Delta t \rfloor, \tag{4.6}$$

where $F_{i,p}$ denotes the position component of F_i. Consequently, the time of entry, $t_{i,r}^{\text{in}}$, and time of departure, $t_{i,r}^{\text{in}}$, are well-defined functions of w_i through (3.10) when (4.5) is satisfied.

Finally, we only consider enforcement of the inequality constraints (4.4d) at the times $t_0, \ldots, t_K$, gather $w = (w_1, \ldots, w_N)$ and define

$$J_i(w_i) = V_i^{\text{f}}(x_{i,N}) + \sum_{k=0}^{K-1} \int_{t_k}^{t_{k+1}} l_i(\mathrm{x}_i(t), \mathrm{u}_{i,k}) \mathrm{d}t. \tag{4.7}$$

Using $\mathcal{I}_a = \{0, \dots, a\}$, for $a \in \mathbb{N}$, the direct reformulation of OCP (4.4) reads

$$\min_{w,T} \quad \sum_{i=1}^{N} J_i(w_i) \tag{4.8a}$$

$$\text{s.t.} \quad x_{i,k} = \hat{x}_{i,0}, \qquad i \in \mathcal{N}, \tag{4.8b}$$

$$x_{i,k+1} = F_i(x_{i,k}, u_{i,k}, \Delta t), \qquad i \in \mathcal{N},\ k \in \mathcal{I}_{K-1}, \tag{4.8c}$$

$$h_i(x_{i,k}, u_{i,k}) \leq 0, \qquad i \in \mathcal{N},\ k \in \mathcal{I}_{K-1}, \tag{4.8d}$$

$$p_i(t_{i,r}^{\text{in}}, w_i) = p_{r,i}^{\text{in}}, \qquad i \in \mathcal{N},\ r \in \mathcal{R}_i, \tag{4.8e}$$

$$p_i(t_{i,r}^{\text{out}}, w_i) = p_{r,i}^{\text{out}}, \qquad i \in \mathcal{N},\ r \in \mathcal{R}_i, \tag{4.8f}$$

$$\left(t_{r,i}^{\text{out}} \leq t_{r,j}^{\text{in}}\right) \vee \left(t_{r,j}^{\text{out}} \leq t_{r,i}^{\text{in}}\right), \qquad (i,j,r) \in \mathcal{C}_S, \tag{4.8g}$$

$$p_{i,k} + \delta_{ij} \leq p_{j,k}(t), \qquad (i,j) \in \mathcal{C}_R,\ k \in \mathcal{I}_K. \tag{4.8h}$$

A practical reformulation of the SICA constraints

A common way to handle constraints such as (4.8g) is to introduce auxiliary binary variables and use the "big-M" technique [85]. In particular, with $b_{r,i,j} \in \{0,1\}$ and M sufficiently large, an equivalent formulation of (4.8g) is

$$t_{r,i}^{\text{out}} - t_{r,j}^{\text{in}} \leq b_{r,i,j} M, \tag{4.9a}$$

$$t_{r,j}^{\text{out}} - t_{r,i}^{\text{in}} \leq (1 - b_{r,i,j}) M. \tag{4.9b}$$

For $b_{r,i,j} = 1$, vehicle j is constrained to cross CZ r before vehicle i since (4.9a) cannot be active, with the opposite holding true for $b_{r,i,j} = 0$. Collecting $b_{r,i,j}$ for all $(i,j,r) \in \mathcal{C}_S$ in b, we note that (4.8) can be solved as a Mixed-Integer NLP (MINLP) where b encodes the crossing order.

4.4 Solution strategies

As noted earlier, the set of possible solutions to the coordination problem grows rapidly growth in N, reflected by the $2^{|\mathcal{C}_S|}$ possible values of b. Regardless of how the problem is solved, finding *a* solution can in the worst case require full exploration of the solution space. Given the nature of the intended application, attempting to solve (4.8) directly in general not a viable option for anything but small problem instances. Any realistic method devised to solve the problem *must* therefore rely on heuristics.

4.4.1 Two-Stage approaches

The heuristic approach taken in this thesis and some other work (e.g. [43, 48, 68]) is to split the solution of OCP (4.8) in two stages, where

1. The crossing order first is found using a heuristic
2. The continuous part of the problem is solved second, giving w.

While several heuristics are possible for the first stage, with a particular choice discussed in Section 5.3, w can be found as the solution to an NLP. In particular, by "freezing" the crossing order in (4.8) we obtain the *fixed order coordination problem*

$$\begin{aligned}
\min_{w,T} \quad & \sum_{i=1}^{N} J_i(w_i) && && \text{(4.10a)}\\
\text{s.t.} \quad & x_{i,k} = \hat{x}_{i,0}, && i \in \mathcal{N}, && \text{(4.10b)}\\
& x_{i,k+1} = F_i(x_{i,k}, u_{i,k}, \Delta t), && i \in \mathcal{N},\ k \in \mathcal{I}_{K-1}, && \text{(4.10c)}\\
& h_i(x_{i,k}, u_{i,k}) \leq 0, && i \in \mathcal{N},\ k \in \mathcal{I}_{K-1}, && \text{(4.10d)}\\
& p_i(t^{\text{in}}_{i,r}, w_i) = p^{\text{in}}_{r,i}, && i \in \mathcal{N},\ r \in \mathcal{R}_i, && \text{(4.10e)}\\
& p_i(t^{\text{out}}_{i,r}, w_i) = p^{\text{out}}_{r,i}, && i \in \mathcal{N},\ r \in \mathcal{R}_i, && \text{(4.10f)}\\
& t^{\text{out}}_{r,i} \leq t^{\text{in}}_{r,j}, && (i,j,r) \in \mathcal{S}, && \text{(4.10g)}\\
& p_{i,k} + \delta_{ij} \leq p_{j,k}, && (i,j) \in \mathcal{C}_R,\ k \in \mathcal{I}_K, && \text{(4.10h)}
\end{aligned}$$

where the difference from (4.8) is the replacement of constraint (4.8g) with the simple inequality (4.10g). Here, the (given) crossing order is denoted by $\mathcal{S}$, and $(i,j,r) \in \mathcal{S}$ means that vehicle i crosses CZ r before vehicle j.

The removal of constraints (4.8g), render (4.10) a continuous program. However, even though the crossing order is provided externally, the exact time of entry $t^{\text{in}}_{i,r}$ and exit $t^{\text{out}}_{i,r}$ for each vehicle i and each CZ r is to be determined. Since these depend nonlinearly on w_i through (4.10e) and (4.10f), (4.10) is a non-convex Nonlinear Program (NLP), even for the case where (4.10c) is linear, (4.10d) affine and (4.10a) convex. The non-convexity is illustrated in Figure 4.3, and can be understood as even though the "side" at $\cup_r \hat{\mathcal{B}}_{i,j,r}$ which is passed is given by $\mathcal{S}$, the w must still be found in a non-convex subset of the system's configuration space.

To solve (4.10), one can deploy a standard algorithm for NLPs (see Section 2.3), but convergence is in general only guaranteed to local optima. However, it is worth noting that if the global optimum (4.10) is found under the "correct" order $\mathcal{S}$, the minimizer of (4.10) is also the global minimizer of (4.8).

4.4.2 Sequential approaches

Rather than finding the crossing order and then solving OCP (4.8), one can construct a (sub-optimal) solution to (4.8) by deciding the action for each

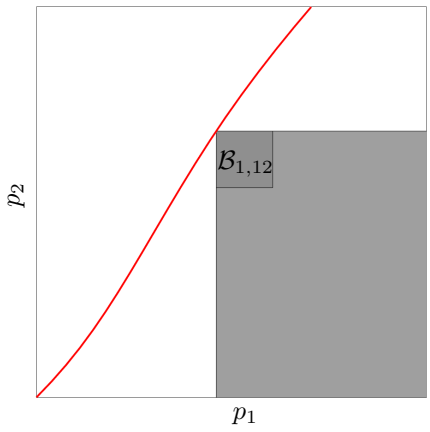


Figure 4.3: Illustration of the inherent non-convexity of the fixed order problem in the configuration space of a 2-vehicle scenario. The light gray area corresponds to the infeasible configurations due to constraint (4.10g) and $v_i(t) \geq 0$.

vehicle in sequence. In particular, a solution can be built up through the following steps:

1. First, find a decision order $\mathcal{O}$ through a heuristic, then
2. Find the uncoordinated, "greedy" solution of the first vehicle in the decision order
3. Find the optimal solution of the second vehicle, such that collisions are avoided with the first vehicle.
4. Find the optimal solution of the next vehicle such that collisions are avoided with previous vehicles. Repeat 4 until last vehicle.

We explored such a scheme in [60][1]. Here, the vehicles are sorted in ascending order of their time-to-react, t_i^{TTR}, defined as the time before a vehicle passes the point of no return, p_i^{NR}, when the current velocity is kept. Beyond p_i^{NR} the vehicle cannot be prevented to enter the intersection, even at maximum braking. That is, $t_i^{TTR} = (p_i^{NR} - p_{i,0})/v_{i,0}$ where

$$p_i^{NR} = \min p_{i,0} \text{ s.t. } x_{i,k+1} = F_i(x_{i,k}, u_{i,k}),\ h_i(x_{i,k}, u_{i,k}) \leq 0,\ p_{i,k} \leq p_{i,r}^{\text{in}}, \forall k$$

The point of no return is illustrated by the boundary between the white and gray areas in Figure 4.4.

Consequently, the vehicle that is closest to its p_i^{NR} gets to decide what action to take first, thereafter the second closest, etc[2]. To find its state

[1] In this paper, we only studied problems without RECA constraints. The extension that include such constraints is however rather simple.

[2] In the original paper, the method was developed for discrete time systems, and even though conceptually the same, t_i^{TTR} was defined using tools from reachability analysis.

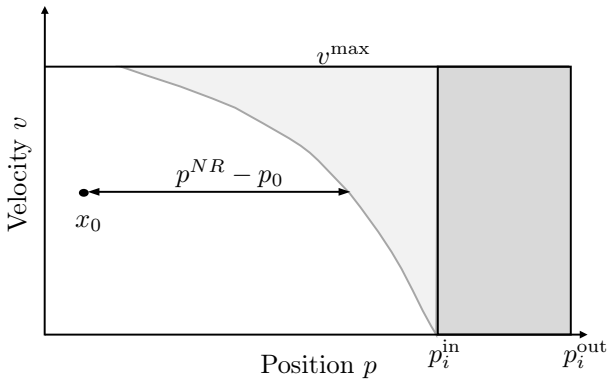


Figure 4.4: Illustration of the construction of the *time-to-react* heuristic. The dark-gray area corresponds to the intersection and the light gray areas to states where no feasible input exists such that the vehicle stops before the intersection.

and control trajectories w_i, each vehicle solves two OCPs where it evaluates the cost of crossing the intersection before or after all vehicles that decided before it, and picks the solution corresponding to the lower cost. In this way, collision free w is obtained by solving $2N - 1$ OCPs. If $p_{i,0} < p_i^{NR}$ for all vehicles, the procedure always finds a solution.

A number of contributions can be found in the literature that propose schemes with the same fundamental principle. A notable OC-based example is found in [68, 84], where the decision order is derived from a First-Come-First-Served type heuristic and the vehicle state- and control trajectories are found using indirect methods. In this context it is also worth mentioning the seminal, non OC-based approach of [28].

4.5 Summary

In this chapter, we introduced an OC formulation of the intersection coordination problem. Besides the restrictive Full Automation assumption, the OC formulation is general and only depends on the non-restrictive Vehicles on Rails assumption and a mild geometry approximation. We presented both the general, fixed order and transcribed/discrete time varieties of the problem and discussed their properties. We also introduced the Two-Stage and Sequential strategies to handle the inherent combinatorial complexity. In the next chapter, we discuss how to solve the OCPs in a practical setting by employing a Two-Stage strategy where most computation necessary to select the crossing order and solve the fixed order problem is distributed to the vehicles.

Chapter 5

Computational Methods

In this chapter we discuss methods with which the OCP of the previous chapter can be solved in a practical context. Given that the intention is to apply the OCP in a receding horizon fashion, we note that it must be solved in "real-time". That is, the time between the measurement of the current state and the application of the control command sufficiently small enough for the scenario not to have changed significantly. However, the OCP is large and because the vehicles are physically dispersed, its solution must rely on communication to some extent. Since communication requires time, both the amount of data exchanged and the number of transmissions must be low. The approach taken to solve the OCP is thus of importance.

Fully centralized approaches

One alternative is to solve the OCP at a central network node, e.g., a Road-Side Unit (RSU), Cloud Server or designated "lead vehicle", and pass the solution to the vehicles. The information sent from the central node could either be the current control command to apply, $u^*_{i,0}$, a sequence of control commands u_i, or a reference trajectory to be tracked (e.g. the optimal x^*_i). The vehicles would in turn send up-to-date state information, motion models, constraints and objectives to the central node. One drawback of such *centralized* schemes is that the central node needs extensive computational capabilities to retain real-time performance with increasing scenario size. Other drawbacks include the introduction of a single point of failure, the necessity to share private or proprietary information and a heavy reliance on the wireless network to transmit large amounts of safety-critical data.

Partly centralized approaches

Another alternative is to distribute the solution of the OCP, and perform the bulk of computations on board the vehicles. This could scale better with

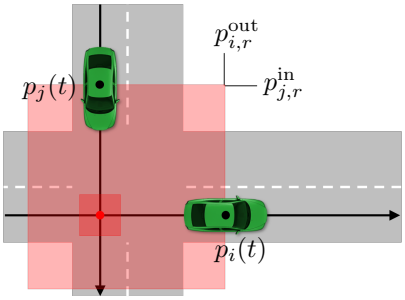


Figure 5.1: Simplified scenario with a single Conflict Zone. The light red area shows the mutual exclusion region and the dark red the zone where collisions can occur.

scenario size and reduce the requirements on the central node, since each vehicle brings additional computational resources. This could also partly circumvent the need to share private data and possibly reduce the amount of information exchanged.

In this chapter, we discuss two such algorithms for the fixed-order problem (4.10) and introduce a heuristic for the crossing order $\mathcal{S}$. These can be used in the two-stage scheme discussed in Section 4.4.1.

5.1 A method for simple fixed-order problems

In this section, we introduce an SQP-based method for fixed-order problems without RECA constraints, which is discussed at length in Paper B. To ease the presentation, the method is described for scenarios with a single CZ, shown in Figure 5.1, but it is applicable to general layouts.

5.1.1 Decomposition

Without the RECA conditions (4.10h), the fixed-order problem (4.10) is reduced to

$$\begin{aligned}
\min_{w,T} \quad & \sum_{i=1}^{N} J_i(w_i) && && \text{(5.1a)}\\
\text{s.t.} \quad & x_{i,k} = \hat{x}_{i,0}, && i \in \mathcal{N}, && \text{(5.1b)}\\
& x_{i,k+1} = F_i(x_{i,k}, u_{i,k}, \Delta t), && i \in \mathcal{N},\ k \in \mathcal{I}_{K-1}, && \text{(5.1c)}\\
& h_i(x_{i,k}, u_{i,k}) \leq 0, && i \in \mathcal{N},\ k \in \mathcal{I}_{K-1}, && \text{(5.1d)}\\
& p_i(t_i^{\text{in}}, w_i) = p_i^{\text{in}},\ p_i(t_i^{\text{out}}, w_i) = p_i^{\text{out}}, && i \in \mathcal{N}, && \text{(5.1e)}\\
& t_i^{\text{out}} \leq t_j^{\text{in}}, && (i,j) \in S. && \text{(5.1f)}
\end{aligned}$$

All inter-vehicles couplings are thus due to constraints (5.1f) through the

timeslots T_i. With this structure, Problem (5.1) can be decomposed into the *timeslot allocation problem*

$$\begin{aligned}
\min_{T} \quad & \sum_{i=1}^{N} V_i(T_i) && && (5.2a)\\
\text{s.t.} \quad & T_i \in \operatorname{dom}\left(V_i(T_i)\right), && i \in \mathcal{N}, && (5.2b)\\
& t_i^{\text{out}} \leq t_j^{\text{in}}, && (i,j) \in S, && (5.2c)
\end{aligned}$$

and the *vehicle problems*

$$\begin{aligned}
V_i(T_i) = \min_{w_i} \quad & J_i(w_i) && && (5.3a)\\
\text{s.t.} \quad & x_{i,k} = \hat{x}_{i,0}, && && (5.3b)\\
& x_{i,k+1} = F_i(x_{i,k}, u_{i,k}, \Delta t), && k \in \mathcal{I}_{K-1}, && (5.3c)\\
& h_i(x_{i,k}, u_{i,k}) \leq 0, && k \in \mathcal{I}_{K-1}, && (5.3d)\\
& p_i(t_i^{\text{in}}, w_i) = p_i^{\text{in}}, && && (5.3e)\\
& p_i(t_i^{\text{out}}, w_i) = p_i^{\text{out}}, && && (5.3f)
\end{aligned}$$

where $\operatorname{dom}(V_i(T_i))$ denotes the set of T_i for which (5.3) is feasible. Rather than solving the (often large) NLP (5.1) directly, one can thereby obtain a solution to (5.1) through the significantly smaller (5.2).

Characterization of $\operatorname{dom}((T_i))$

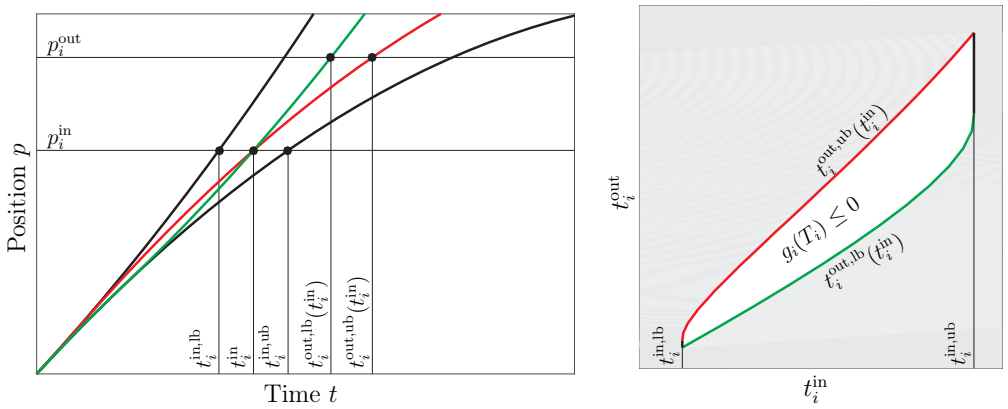
We showed in [52, 86] that $\operatorname{dom}(V_i(T_i))$ can be expressed as

$$g_i(T_i) = \begin{bmatrix} t_i^{\text{in}} - t_i^{\text{in,ub}} \\ t_i^{\text{in,lb}} - t_i^{\text{in}} \\ t_i^{\text{out}} - t_i^{\text{out,ub}}(t_i^{\text{in}}) \\ t_i^{\text{out,lb}}(t_i^{\text{in}}) - t_i^{\text{out}} \end{bmatrix} \leq 0, \tag{5.4}$$

where

$$\begin{aligned}
t_i^{\text{out,lb}}(t_i^{\text{in}}) = \min_{w_i, t_i^{\text{out}}} \quad & t_i^{\text{out}} && && (5.5a)\\
\text{s.t.} \quad & \text{(5.3b)} - \text{(5.3d)}, && && (5.5b)\\
& p_i(t_i^{\text{in}}, w_i) = p_i^{\text{in}},\ p_i(t_i^{\text{out}}, w_i) = t_i^{\text{out}}, && && (5.5c)
\end{aligned}$$

$$t_i^{\text{out,ub}}(t_i^{\text{in}}) = \max_{w_i, t_i^{\text{out}}} \quad t_i^{\text{out}} \quad \text{s.t.} \quad (5.5\text{b}), (5.5\text{b}), \tag{5.6}$$



(a) Solutions to (5.5)-(5.8). (b) Example of $g_i(T_i) \leq 0$

Figure 5.2: Illustration of $g_i(T_i)$ and its definition.

$$t_i^{\text{in,lb}} = \min_{w_i, t_i^{\text{in}}} \quad t_i^{\text{in}} \quad \text{s.t.} \quad (5.3\text{b}) - (5.3\text{d}), \quad p_i(t_{i,r}^{\text{in}}, w_i) = p_{r,i}^{\text{in}}, \tag{5.7}$$

$$t_i^{\text{in,ub}} = \max_{w_i, t_i^{\text{in}}} \quad t_i^{\text{in}} \quad \text{s.t.} \quad (5.3\text{b}) - (5.3\text{d}), \quad p_i(t_{i,r}^{\text{in}}, w_i) = p_{r,i}^{\text{in}}. \tag{5.8}$$

That is, for (5.3) to be feasible, t_i^{in} must lie between the largest, $t_i^{\text{in,ub}}$, and smallest, $t_i^{\text{in,lb}}$ time of entry, which are defined by the trajectories resulting from the maximum retarding and accelerating control, respectively. Similarly, the time of departure t_i^{out} must lie between the largest, $t_i^{\text{out,ub}}(t_i^{\text{in}})$, and smallest, $t_i^{\text{out,lb}}(t_i^{\text{in}})$ time of departure, for a given time of entry t_i^{in}. The trajectories corresponding to solutions of (5.5)-(5.8) and their relation to $g_i(x)$ are illustrated in Figure 5.2.

5.1.2 Solving the decomposed problem using SQP

In Paper A, we solve (5.2) using SQP, as briefly summarized below.

The Lagrange function associated with (5.2) is

$$\mathcal{L} = \sum_{i \in \mathcal{N}} \mathcal{L}_i(T_i, \mu_i) + \sum_{(i,j) \in \mathcal{S}} \gamma_{i,j}(t_i^{\text{out}} - t_j^{\text{in}}), \tag{5.9}$$

where $\mathcal{L}_i(T_i, \mu_i) = V_i(T_i) + \lambda_i^\top g_i(T_i)$, and $\mu = (\mu_i, \ldots, \mu_N)$ and $\gamma_{i,j}$ denotes the Lagrange multipliers of constraints (5.2b) and (5.2c), respectively. Collecting $\gamma_{i,j}$, $\forall (i,j) \in \mathcal{S}$ in γ, the SQP performs the iteration

$$T^{[k+1]} = T^{[k]} + \alpha[k] \Delta T^{[k]}, \tag{5.10a}$$

$$\lambda^{[k+1]} = (1 - \alpha^{[k]})\lambda^{[k]} + \alpha^{[k]} \lambda_{QP}^{[k]}, \tag{5.10b}$$

$$\gamma^{[k+1]} = (1 - \alpha^{[k]})\gamma^{[k]} + \alpha^{[k]} \gamma_{QP}^{[k]}, \tag{5.10c}$$

Algorithm 3 Solution of simplified fixed order problem. C and v_i denote the central node and vehicle i respectively.

1: **procedure** IntersectionSQP
2: C : sends initiation command, sets $k = 0$
3: $\forall v_i$: Sends $T_i^{[0]}$ and $\mathcal{D}_i^{[0]}$ to C
4: **while** Exit criteria not met **do**
5: C : Solve QP (5.11), sets $\alpha^{[k]} = 1$
6: C : Forms tentative $T^{[k+1]} = T^{[k+1]} + \alpha^{[k]}\Delta T^{[k]}$, sends it to v_i
7: $\forall v_i$: Computes and sends $\mathcal{D}_i^{[k+1]}$ to C
8: C : Evaluates quality of step using $\mathcal{D}_i^{[k+1]}$
9: **if** Step Accepted **then**
10: C : Sets $k = k + 1$, evaluates (5.10)
11: **else**
12: C : Reduces $\alpha^{[k]}$, goes to line 6
13: **end if**
14: **end while**
15: **end procedure**

from some initial solution candidate $(T^{[0]}, \lambda^{[0]}, \gamma^{[0]})$, until a convergence criteria is met. Here, $\alpha^{[k]} \in]0, 1]$ is a step-size and $(\Delta T_i^{[k]}, \lambda_{QP}^{[k]}, \gamma_{QP}^{[k]})$ is the primal-dual solution to the convex QP

$$\min_{\Delta T} \quad \sum_{i=1}^{N} \frac{1}{2}\Delta T_i^\top H_i^{[k]} \Delta T_i + \nabla_{T_i} V_i^\top \Delta T_i \tag{5.11a}$$

$$\text{s.t.} \quad g_i(T_i^{[k]}) + \nabla_{T_i} g_i^\top \Delta T_i \leq 0, \qquad i \in \mathcal{N}, \tag{5.11b}$$

$$t_i^{\text{out}\,[k]} + \Delta t_i^{\text{out}} \quad \leq t_j^{\text{in}\,[k]} + \Delta t_j^{\text{in}}, \qquad (i,j) \in S, \tag{5.11c}$$

where all derivatives are evaluated at $T_i^{[k]}$ and $H_i^{[k]}$ is the suitably modified Hessian of $\mathcal{L}_i$. The selection of $\alpha^{[k]}$ is detailed in Paper B and [45], but we note that it requires re-evaluation of $V_i(T_i)$ and $g_i(T_i)$ when full steps ($\alpha^{[k]} = 1$) cannot be taken.

5.1.3 Practical application

In a practical setting, the SQP subproblem (5.11) must be solved at a central network node, while the data required to build (5.11) can be computed on-board the vehicles. For each vehicle, this data is obtained by solving (5.3), and the optimization problems defining $g_i(T_i)$ for the given $T_i^{[k]}$, and computing the associated sensitivities. A practical SQP procedure is summarized in Algorithm 3 where the information flow between the central node

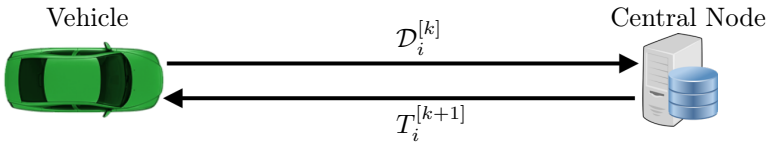


Figure 5.3: Illustration of the information exchange in Algorithm 3.

and the vehicles is shown in Figure 5.3. Here, the data from vehicle i at iteration k is collected in $\mathcal{D}_i^{[k]} = \{V_i, \nabla_{T_i} V_i, \nabla^2_{T_i} V_i, g_i, \nabla_{T_i} g_i, \nabla^2_{T_i} g_i\}$.

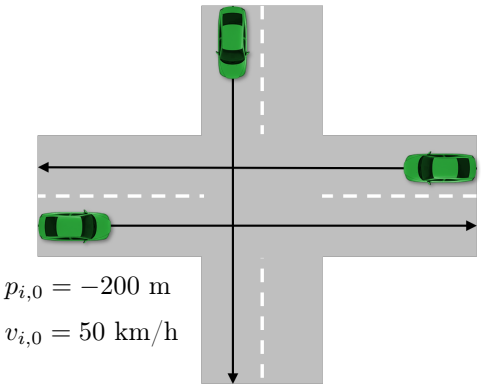
Since the SQP sub-problem (5.11) typically is small compared to (5.3) and the problems defining $g_i(T_i^{[k]})$, the main effort is the evaluation of $\mathcal{D}_i^{[k]}$ on line 7. However, as they are decoupled, evaluation of $V_i(T_i)$ and the elements of $g_i(T_i)$ can also be parallelized. In an ideal setting, the time required at line (7) is consequently the maximum time it takes to solve (5.3),(5.5)-(5.8) among all vehicles.

Moreover, the optimization problems solved for $\mathcal{D}_i$ are structured such that highly efficient solvers for direct optimal control can be used. In particular, when (5.3c) are linear, (5.3d) affine and $J_i(w_i)$ quadratic, (5.3) is a QP, and fast MPC QP solvers, e.g. HPMPC [87], or FORCES [88] can be employed. As discussed in Paper B, linear (5.3c) and affine (5.3d) also enables the constraints $g_i(T_i)$ to be evaluated by solving Linear Programs (LPs) rather than NLPs, allowing additional speed ups.

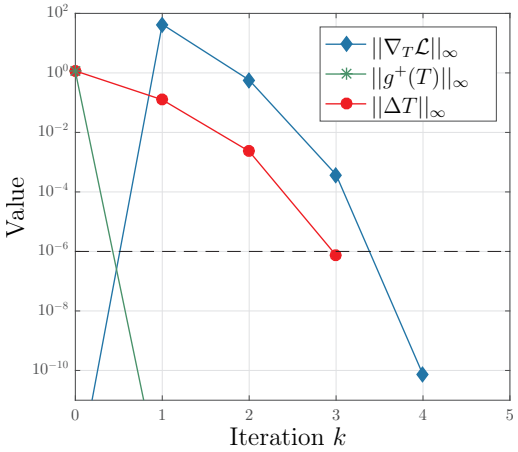
5.1.4 Performance illustration

We demonstrate the performance of Algorithm 3 using a simple example. The scenario is that shown in Figure 5.4(a), where three vehicles start 200 meters before the intersection with a speed of 50 km/h. If no action is taken, all three vehicles reach the intersection center at the same time, leading to a three-way collision. The motion model (5.3c) is the simple double integrator $\dot{p}_i(t) = v_i(t)$, $\dot{v}_i(t) = u_i(t)$, where the input $u_i(t)$ is the acceleration and (5.3d) are bounds $u_i^{\min} \leq u_i(t) \leq u_i^{\max}$, $0 \leq v_i(t)$. The horizon is $K = 200$ and the sample time is $\Delta t = 0.1$ s. The objective is $J(w_i) = 32.12(v_{i,K} - v^{\mathrm{r}})^2 + \sum_{k=0}^{K-1}(v_{i,k} - v^{\mathrm{r}})^2 + 10u_{i,k}^2$, with $v^{\mathrm{r}} = 50$ km/h.

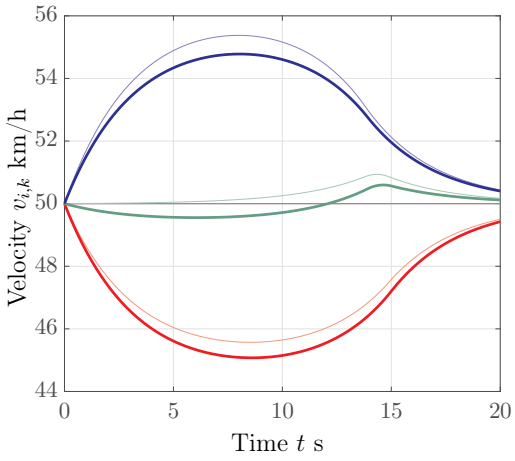
Figure 5.4(b) shows the development of three progress metrics of the SQP: stationarity of $\mathcal{L}$, $||\nabla_T \mathcal{L}||_\infty$, the size of the change in the timeslots, $||\Delta T||_\infty$, and feasibility $||g^+(T)||_\infty = ||\max(g(T), 0)||_\infty$. Here, $g(T)$ collects $g_i(T_i), \forall i \in \mathcal{N}$ and $t_i^{\mathrm{in}} - t_j^{\mathrm{out}}\ \forall (i,j) \in \mathcal{S}$. As the figure shows, primal feasibility (i.e., non-overlapping timeslots that are feasible for all vehicles) is obtained after the first iteration, and the following iterations make adjustments towards optimality. Note in particular the development of $||\Delta T||_\infty$, which is a measure of the change in T between different iterations. As can be seen, the change between iteration 0 and 1 is around 0.1 s, and around 1 ms



(a) Example Configuration



(b) Progress of Algorithm 3. The dashed line shows the set tolerance.



(c) Velocity Profiles v_i corresponding to the iterates $T_i^{[k]}$. The solution is drawn in bold.

Figure 5.4: Configuration Illustration and data from example scenario. Here, $T_i^{[0]}$ was such that all vehicles satisfied (5.3e),(5.3f) by keeping the initial velocity of 50 km/h. Note that the trajectories from all iterates are drawn in (c).

between iteration 1 and 2. Since the vehicles are required to enforce T_i through constraints (5.3e) and (5.3f), it is worth noting that the standard deviation of the error in corrected[1] GNSS positioning typically lie between 0.01-0.5 m [89]. If $\Delta T_i = 1$ ms and a vehicle crosses the intersection at 50 km/h, the corresponding change in the vehicle position at T_i is 0.0139 m. Timeslot updates in this order would therefore be around or below the noise floor, and not give any noticeable difference in the behavior of the actual system. Even more, since feasibility is attained immediately and already $\Delta T_i^{[1]}$ is small, the algorithm could probably be stopped after iteration 1 with

[1]Using Real Time Kinematic (RTK) or Differential GNSS.

Problem Size			V2C Com.		C2V Com.	
			Rounds	#Floats	Rounds	#Floats
Centralized	Central:	1812 Variables 905 Constraints	1	36 (12 p. veh.)	1	see Caption
	Vehicle:	see Caption				
SQP	Central:	6 Variables 26 Constraints	4	120 (10 p. veh./rd.)	3	18 (6 p. rd.)
	Vehicle:	602 Variables 301 Constraints				

Table 5.1: Problem size and communication requirements of a fully and partly centralized solution of the example instance. The communication flows from the central-node to the vehicles in the centralized case are dependent on how the scheme is implemented. It could consist of, e.g., the vehicle control commands, the full solution trajectories or the optimal T_i. In the latter case, (5.3) must be solved on-board the vehicles to obtain u_i.

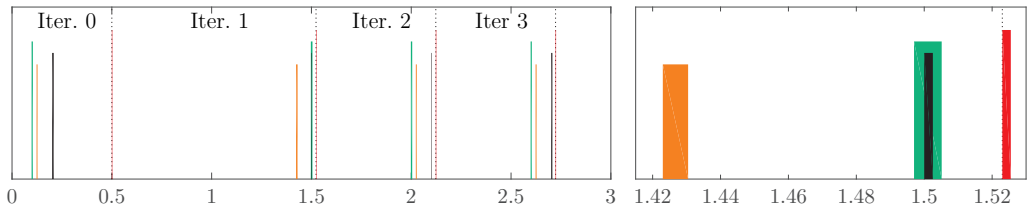
marginal performance loss. This can be understood intuitively by inspecting the trajectories corresponding to $T_i^{[k]}$, illustrated in Figure 5.4(c), and noting the small difference between the result at iteration 1,2 and 3.

From our observations, the behavior exhibited in the example is general for small problems: The algorithm attains feasibility after the first or second iterate, and converge to a relevant tolerance within 3-4 iterations. The behavior also seems to generalize to larger problems, even though a few more iterations typically are needed.

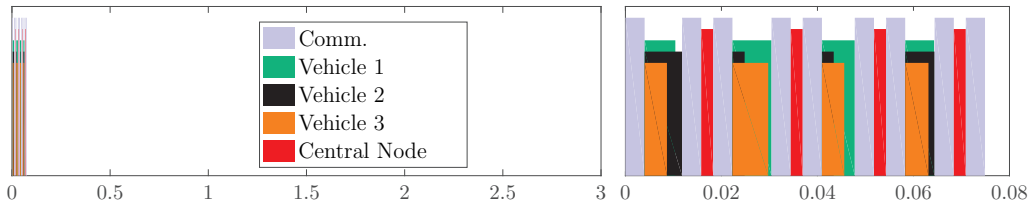
For comparison, we summarize the differences in problem size and communication requirements between the proposed SQP algorithm and a strategy where (5.1) is solved centrally and the solution communicated to the vehicles. As the numbers highlight, the SQP algorithm trade-off reduced computation centrally with higher communication requirements.

5.1.5 Experimental validation

We applied Algorithm 3 in an experiment with three vehicles. Each vehicle was equipped with an 802.11p-based communication system, dSpace MicroAutoBox II (MABx) real-time computers and laptops which communicated internally using UDP over Ethernet, and global-time synchronization was achieved through GPS. The objective function and dynamics were chosen so that (5.3) was a QP and (5.5)-(5.8) could be solved as LPs. The vehicle level problems (5.3), (5.5)-(5.8) had about 600 variables, 600 inequality and 400 equality constraints each, and were solved with HPMPC [87]. The SQP sub-problems (5.11) had 6 variables and 12 inequality constraints and was solved with Matlab's *quadprog*. All problems were solved on standard laptops with solve-times in the 1-2 to ms range. While this should have



(a) Observed in experiment



(b) Expected results with better implementation.

Figure 5.5: Illustration of the time spent on computation, communication and idling in an experimental instance of Algorithm 3. The width of the bars correspond to the time spent whereas the difference in height are for visualization only. For the vehicles, it consists of the time required to solve (5.3), (5.5)-(5.8) in sequence, and for the central node the time required to solve (5.11). Right hand-side plots are close-ups.

given an execution time for Algorithm 3 of less than 0.1 s, we observed numbers in the 1-3 s range. The primary reason was likely issues with the MABx UDP-module which caused long periods of idling. The time spent on different tasks during a scenario similar to that of Figure 5.4 is visualized in Figure 5.5(a). As the figure illustrates, only about 1.6% of the time was spent on solving (5.3), (5.5)-(5.8). While it was not recorded during the experiments, the time-performance of the same communication system was evaluated experimentally in [90], and transmission times in the 2-4 ms range was reported. It is therefore reasonable to assume that with correctly functioning hardware, solution-times of less 5% than that observed could be obtained. For comparison, an illustration of the time-line of such a case is given in Figure 5.5(b), where Algorithm 3 terminates in less than 80 ms.

A lesson learned from these experiments is that communication likely will require more time than computation, and that the performance of the overall system is highly dependent on well functioning communication links. A longer discussion can be found in Paper B.

5.2 A method for fixed-order problems

While the method described in the previous section is applicable to most scenarios, it could introduce problems since the RECA constraints are not considered. Consider, for instance, the case where the optimal solution for

two vehicles on the same lane is to cross the intersection in direct succession. If the first vehicle is significantly slower than the second when crossing, a collision could occur immediately after the intersection. Such issues could be avoided if the RECA constraints are considered when the problem is solved. However, this introduces additional couplings in the problem, and destroys the hierarchical structure used in (5.2),(5.3). Due to this, the distribution scheme of Algorithm 3 cannot be employed.

A remedy is to solve the original, centralized, formulation of the fixed-order problem (4.10), including the RECA constraints, but distribute the operations in the optimization algorithm. In this section, we summarize such a method, where (4.10) is solved using Primal-Dual Interior Point (PDIP) Algorithm. The method is detailed in Paper F.

5.2.1 Interior point-formulation of Problem (4.10)

As discussed in Section 2.3.3, the PDIP algorithm operates by updating the primal-dual solution candidate $z^{[k]}$ and barrier parameter $\tau^{[k]}$ through the iteration

$$z^{[k+1]} = z^{[k]} + \alpha^{[k]} \Delta z^{[k]}, \tag{5.12}$$

$$\tau^{[k+1]} = \xi(\tau^{[k]}, z^{[k]}). \tag{5.13}$$

This is performed until the perturbed KKT conditions $r_{\tau^{[k]}}(z^{[k]}) \approx 0$ (c.f (2.18)) and a small τ has been obtained through the update-strategy ξ. The *search direction* $\Delta z^{[k]}$ is computed by solving the KKT system

$$M(z^{[k]}) \Delta z^{[k]} = -r_{\tau^{[k]}}(z^{[k]}), \tag{5.14}$$

where $M = \frac{\partial r_\tau}{\partial z}$. The *step-size* $\alpha^{[k]}$ can thereafter be selected using a backtracking line-search. A PDIP algorithm can be applied to (4.10), in which case z includes w, T, the multipliers and slack-variables associated with constraints (4.10b)-(4.10h). .

Finding the search direction

While it lacks the simple hierarchical structure of (5.2),(5.3), the fixed-order problem (4.10) has a structure that allows parallel operations in the solution of (5.14). This structure is made visible by the following arrangement of the primal-dual variables, where we collect

- w_i, T_i, the multipliers and slack variables associated with constraints (4.10b)-(4.10f) in z_i

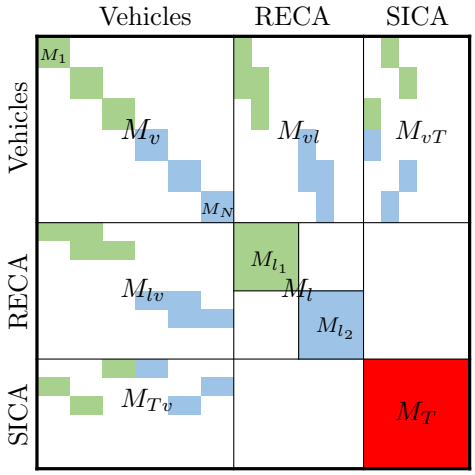


Figure 5.6: Illustration of M. The colors differentiate blocks corresponding to vehicles on different lanes. The various components not are shown to size.

- the multipliers and slack variables associated with the RECA constraints (4.10h) for lane j in z_{l_j}
- the multipliers and slack variables associated with the SICA constraints (4.10g) in z_T.

and let $z = (z_v, z_l, z_T)$, where $z_v = (z_{v_1}, \ldots, z_{v_N})$, $z_l = (z_{l_1}, \ldots, z_{l_L})$, with the number of lanes L. The perturbed KKT conditions are ordered similarly, so that $r = (r_v, r_l, r_T)$, with $r_v = (r_1, \ldots, r_N)$, and $r_l = (r_{l_1}, \ldots, r_{l_L})$, where the arguments and subscript τ have been dropped for brevity. It is then the case that the matrix M has the structure shown in Figure 5.6, where $M_{\mathrm{xy}} = \frac{\partial r_{\mathrm{x}}}{\partial z_{\mathrm{y}}}$. The calculations required to solve (5.14) can thereafter be divided as

$$\Theta \Delta z_T = \theta \tag{5.15a}$$

$$\Gamma_j \Delta z_{l_j} = \gamma_j - \Lambda_j^\top \Delta z_T, \qquad j = 1, \ldots, L, \tag{5.15b}$$

$$M_{v_i} \Delta z_{v_i} = -r_{v_i} - M_{v_i l(i)} \Delta z_{l(i)} - M_{v_i T} \Delta z_{T(i)}, \qquad i = 1, \ldots, N, \tag{5.15c}$$

where

$$\Theta = \left(M_T - \sum_{i=1}^{N} M_{T v_i} M_{v_i}^{-1} M_{v_i T} - \sum_{j=1}^{L} \Lambda_j \Gamma_j^{-1} \Lambda_j^\top \right) \tag{5.16a}$$

$$\theta = -r_T + \sum_{i=1}^{N} M_{T v_i} M_{v_i}^{-1} r_{v_i} - \sum_{j=1}^{L} \Lambda_j \Gamma_j^{-1} \gamma_j, \tag{5.16b}$$

$$\Gamma_j = M_{l_j} - M_{l_j v(j)} M_{v(j)}^{-1} M_{l_j v(j)}^\top, \tag{5.16c}$$

$$\gamma_j = -r_{l_j} + M_{l_j v(j)} M_{v(j)}^{-1} r_{v(j)}, \tag{5.16d}$$

$$\Lambda_j = -M_{T v(j)} M_{v(j)}^{-1} M_{l_j v(j)}^\top. \tag{5.16e}$$

Algorithm 4 A PDIP algorithm for the fixed order problem. Here, C denotes intersection center, l_j the lane center for lane j, and v_i vehicle i. $l(i)$ is the center of vehicle i's lane, $v(j)$ is the vehicles on lane j.

procedure FixedOrderPDIP($\tau^{[0]}$)
  $C, l_j, v_i : z \leftarrow$ InitializeSolutionCandidates(), $k \leftarrow 0$
  while Exit criteria not met **do**
    $k \leftarrow k + 1$
    $\forall v_i$: Compute $\mathcal{D}^{[k]}_{v_i \to l(i)}$, $\mathcal{D}^{[k]}_{v_i \to C}$ send to $l(i)$ and C .
    $\forall l_j$: Compute $\mathcal{D}^{[k]}_{l_j \to C}$ send to C. .
    C : Solve (5.15a), send Δz_T to all l_j and v_i.
    $\forall l_j$: Solve (5.15b) send Δz_{l_j} to all $v(j)$. .
    $\forall v_i$: Solve (5.15c). .
    $\alpha^{[k]} \leftarrow$StepSizeSelection($z^{[k]}$,$\Delta z^{[k]}$,$\tau^{[k]}$)
    C, l_j, v_i : Perform update (5.12)
    $\tau^{[k+1]} \leftarrow \xi(\tau[k], z^{[k]})$
  end while
end procedure

Here, $l(i)$ denotes the lane of vehicle i and $v(j)$ denotes the vehicles that are on lane j. This enables a dynamic-programming like computational structure with an "upwards" and "downwards" pass: First, the components from each vehicle in $\gamma_j, \Gamma_j, \Lambda_j, \Theta, \theta$ are computed. Second, the components for each lane in Θ and θ are computed. Third, (5.15a) is solved for Δz_T, fourth, (5.15b) is solved for Δz_{l_j}, $\forall j$, and finally, (5.15b) is solved for Δz_{v_i}, $\forall i$.

5.2.2 Practical application

While most operations can be performed separately for each vehicle (calculation of $M_{Tv_i} M_{v_i}^{-1} M_{v_i T}$, $M_{Tv_i} M_{v_i}^{-1} r_{v_i}$, $M_{l(i)v_i} M_{v_i}^{-1} M_{l(i)v_i}^{\top}$, $M_{l(i)v_i} M_{v_i}^{-1} r_{v_i}$, $M_{Ti} M_{v_i}^{-1} M_{l(i)v_i}^{\top}$, solution of (5.15b)), some require information from all vehicles on each lane (calculation of $\Lambda_j \Gamma_j^{-1} \Lambda_j^{\top}$, $\Lambda_j \Gamma_j^{-1} \gamma_j$, solution of (5.15b)) and some information from all vehicles (solution of (5.15a)). In a practical setting, this can be realized by introducing lane-central computational units in addition to the central node of Algorithm 3. These lane-centers could, e.g., be an elected lead-vehicle.

Algorithm 4 summarizes a PDIP scheme where computation is split between vehicles, lane centers and the intersection center, and the corresponding data flows are shown in Figure 5.7. Here, the information sent from vehicle i to its lane center and to the central node is denoted $\mathcal{D}^{[k]}_{v_i \to l(i)}$, $\mathcal{D}^{[k]}_{v_i \to C}$ respectively, and the information sent from lane j's center to the

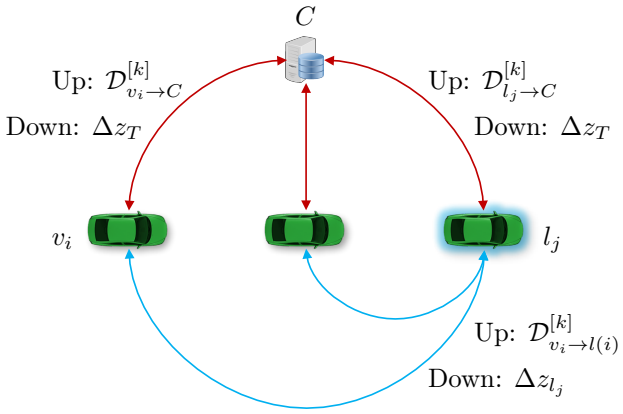


Figure 5.7: Illustration of the information exchange in Algorithm 4.

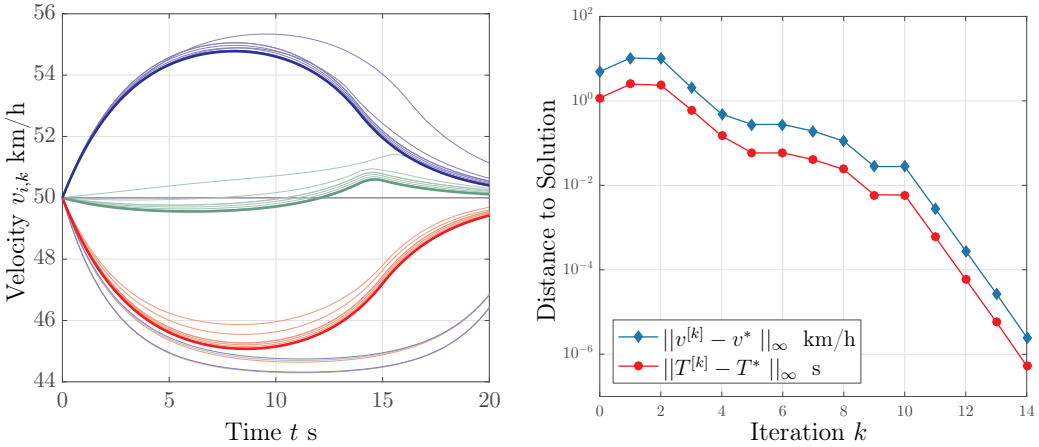
central node is $\mathcal{D}^{[k]}_{l_j \to C}$. Detailed descriptions of the content of $\mathcal{D}^{[k]}_{v_i \to l(i)}$, $\mathcal{D}^{[k]}_{v_i \to C}$ and $\mathcal{D}^{[k]}_{l_j \to C}$ are given in Paper F. Details on the StepSizeSelection and InitializeSolutionCandidates procedure are omitted for brevity, but we note that these involve computation split between v_i, l_j and C. Furthermore, the update on line 11 can be performed in parallel. We refer to Paper Ffor a detailed description of Algorithm (4).

Note that construction of $\mathcal{D}^{[k]}_{v_i \to l(i)}$, $\mathcal{D}^{[k]}_{v_i \to C}$,$\mathcal{D}^{[k]}_{l_j \to C}$ involves the factorization of the matrices M_i and Γ_j. For most problems, the size of M_i is significantly larger than Γ_j and Ψ, and its construction and factorization the most computational expensive parts of Algorithm 4. However, these operations are separable between the vehicles and can therefore be parallelized. Similarly, the factorization of Γ_j is parallelizable between the different lanes. Moreover, the factors computed on lines (5) and (6) can be re-used in the solution of (5.15c) and (5.15b) on lines (9) and (8).

5.2.3 Communication requirements

The communication patterns in Algorithm 4 are more complex than those of Algorithm 3 and not detailed in this summary. However, we note that $\mathcal{D}^{[k]}_{v_i \to L(i)}$ contains the symmetric matrix $M_{l(i)v_i} M_i^{-1} M_{l(i)v_i}^\top \in \mathbb{R}^{K \times K}$. This matrix contains the "second-order information" of the RECA couplings, which are enforced at each time instant k within the horizon K. Consequently, the amount of data shared *each iterate* between a vehicle and its lane center grows as K^2. Since long horizons K typically are desired, this can yield unrealistic communication requirements.

In Paper F, we discuss how to reduce the amount of communicated data while keeping K large. To this end, we proposed the following reformulation



(a) Velocity profiles from the iterates of Algorithm 4. Solution drawn in bold.

(b) Distance to solution. v collects $v_{i,k}$ $\forall i, k$, T collects T_i, $\forall i$

Figure 5.8: Application of Algorithm 4 on the scenario shown in Figure 5.4(a).

of the RECA constraints

$$p_{i,k} + \delta_{ij}/2 \leq B_{ij}(k, \theta_{ij}), \qquad k \in \mathcal{I}_K, \tag{5.17a}$$

$$B_{ij}(k, \theta_{ij}) + \delta_{ij}/2 \leq p_{j,k}, \qquad k \in \mathcal{I}_K, \tag{5.17b}$$

where $B_{ij}(k, \theta_{ij})$ is a function of k, parametrized with $\theta_{ij} \in \mathbb{R}^{n_p}$. Satisfaction of (5.17) implies conservative satisfaction of (4.10h), whereby the RECA constraints can be handled by selection of θ_{ij} This gives data exchange that grows as n_p^2 rather than K^2. Selecting n_p small consequently reduce the communication requirements.

5.2.4 Illustration of difference to Algorithm 3

To demonstrate the difference to the SQP-based Algorithm 3, results from the application of Algorithm 4 to the scenario of Figure 5.4(a) is shown in Figure 5.8. A comparison between Figure 5.8 and Figure 5.4 illustrates that the PDIP-based Algorithm 4 typically need more iterations to reach the solution than the SQP counterpart. While this necessitates more communication, it should be noted that the computational effort per iterate is significantly lower. However, as Figure 5.8(a) shows, a practically acceptable solution is reached rather fast. Following the arguments of Section 5.1.4, the PDIP algorithm could therefore be stopped early. In particular, as soon as $||\Delta T||_\infty < 0.1$, the "coordination" part of the problem can be considered solved. If desired, the vehicle problem (5.3) could thereafter be solved separately for each vehicle using T_i. An example of a scenario with several vehicles on each lane is given in Paper F.

5.3 A crossing order heuristic

The methods discussed so far applied to NLP (4.10), where we assumed that the intersection crossing order $\mathcal{S}$ was given. In this section, we introduce a heuristic for finding $\mathcal{S}$, which is discussed at length in Papers C and D.

5.3.1 Motivation

Finding the optimal solution to OCP (4.8) is in general intractable due to its combinatorial complexity in the number of vehicles N. When optimality requirements are relaxed, a number of heuristics could be used to obtain crossing orders $\mathcal{S}$, provided that some assumptions are satisfied (e.g. that all vehicles can stop before the intersection). An example is "First-Come-First-Served" (FCFS) policies, where the crossing order is based on e.g., proximity or predicted time of arrival to the intersection.

While common in the intersection coordination context (see e.g. [29, 33, 59, 65, 84]) heuristics that depend on a set of rules have weaknesses. In particular, they are inflexible to changes in the problem data, so that the same crossing order is returned irrespective of the objective functions and types of vehicles involved. This is a major drawback in scenarios with vehicle heterogeneity or when the coordination objective is changed. The latter could, e.g., be the case when priorities are changed based on the current traffic demand (e.g. favoring throughput over energy efficiency during rush hour traffic).

A heuristic that to some extent takes the problem data into account is therefore desirable. To this end, we proposed to find the crossing order by solving a Mixed Integer Quadratic Program (MIQP), where the objective function and constraints are constructed using the objective functions and constraints of the involved vehicles.

5.3.2 An MIQP-based heuristic

The MIQP-based heuristic is derived from the decomposition introduced in Section 5.1, where the coordination is due to a timeslot allocation problem

$$
\begin{aligned}
\min_{T} \quad & \sum_{i=1}^{N} V_i(T_i) && && (5.18a)\\
\text{s.t.} \quad & g_i(T_i) \leq 0, && i \in \mathcal{N}, && (5.18b)\\
& \left(t_{r,i}^{\text{out}} \leq t_{r,j}^{\text{in}}\right) \vee \left(t_{r,j}^{\text{out}} \leq t_{r,i}^{\text{in}}\right), && (i,j,r) \in \mathcal{C}_S, && (5.18c)\\
& t_j^{\text{out}} + \delta t_{j,i}, \leq t_i^{\text{out}}, && (i,j) \in \mathcal{C}_R, && (5.18d)
\end{aligned}
$$

where $V_i(T_i)$ is defined by the vehicle problems (5.3) and $g_i(T_i)$ is defined as described in Section 5.1.1. While (5.18) could be used as a heuristic for $\mathcal{S}$, it is a non-convex MINLP, and is associated with the same difficulties as the original Problem (4.8). Instead, we have proposed to obtain the crossing order through the following MIQP approximation of (5.18)

$$\min_{T} \quad \sum_{i=1}^{N} \frac{1}{2} T_i^\top \tilde{H}_i T_i + \left(\nabla V_i - \nabla^2 V_i T_i^{[0]}\right)^\top T_i \tag{5.19a}$$

$$\text{s.t.} \quad g_i(T_i^{[0]}) + \nabla g_i^\top (T_i - T_i^{[0]}) \leq 0, \qquad i \in \mathcal{N}, \tag{5.19b}$$

$$\left(t_{r,i}^{\text{out}} \leq t_{r,j}^{\text{in}}\right) \vee \left(t_{r,j}^{\text{out}} \leq t_{r,i}^{\text{in}}\right), \qquad (i,j,r) \in \mathcal{C}_S, \tag{5.19c}$$

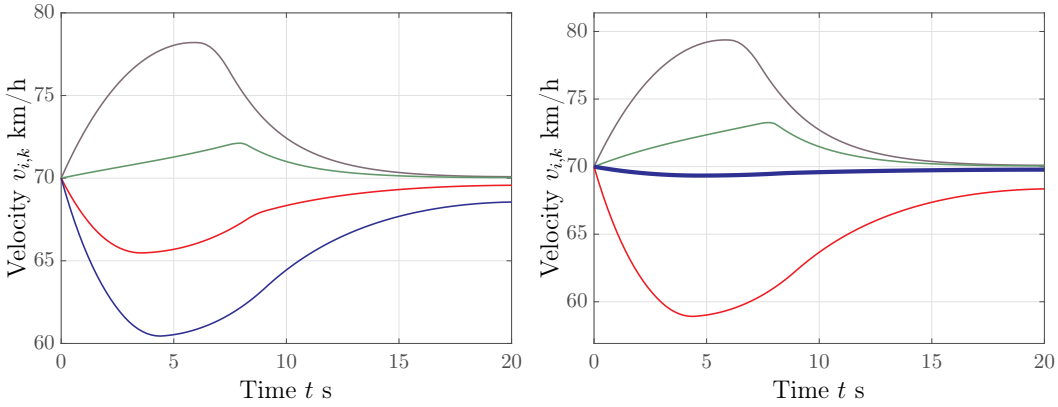
$$t_j^{\text{out}} + \delta t_{j,i} \leq t_i^{\text{in}}, \qquad (i,j) \in \mathcal{C}_R, \tag{5.19d}$$

where the derivatives in (5.19a),(5.19b), are evaluated at $T_i^{[0]}$ and $\tilde{H}_i$ is a positive definite approximation of $\nabla^2 V_i$. The MIQP is thus similar to the SQP sub-problem (5.11), and is based first and second order expansions of (5.18b) and (5.18a) respectively.

While (5.19) still experiences combinatorial growth of the solution space in the number of vehicles, the availability of efficient MIQP solvers (e.g. CPLEX) allows larger and practically relevant problems to be treated in real time. However, since several NLPs need to be solved for each vehicle to evaluate $V_i(T^{[0]_i})$, $g_i(T_i^{[0]})$, construction of the MIQP is expensive. For this reason, we have presented variations of MIQP (5.18) in Papers C and D where the constraints $g_i(T_i)$ and the solution of the corresponding NLPs are removed. With this simplification, the MIQP can be constructed by solving one NLP similar to (5.3) per vehicle.

5.3.3 Performance illustration

To demonstrate the ability of the proposed heuristic to incorporate the problem data when the crossing order is selected, we consider scenario with one vehicle on each lane. We apply the two-stage procedure of Section 4.4.1, with the crossing order selected by the MIQP heuristic, on two variations with the same initial configuration, one with only light vehicles and one with three light and one heavy vehicle. The resulting optimal velocity profiles are given in Figure 5.9, where Figure 5.9(a) shows the homogeneous case and Figure 5.9(b) the heterogeneous. In the former, the crossing order found by the MIQP is $(1, 2, 3, 4)$, (corresponding to black, green, red, blue in Figure 5.9(a)), and in the latter the crossing order is $(1, 2, 4, 3)$. That is, to avoid slowing down the heavier vehicle, the MIQP solution flipped the internal order of vehicles 3 and 4. Additional examples are given in



(a) Homogeneous (b) Heterogeneous.

Figure 5.9: Velocity profiles resulting from the application of Algorithm 5 on two variations of the same 4-vehicle scenario. In (a), the results are shown when all vehicles are passenger cars, and in (b) with one heavy vehicle (bold blue), and three passenger cars.

Algorithm 5 Approximate solution of (4.8)

1: **procedure** TwoStageApproximation
2: C : sends initiation command
3: $\forall v_i$: Constructs $\mathcal{D}_i$, sends to C
4: C : Constructs and solves (5.19) to get $\mathcal{S}$
5: C : Solve (4.10) with $\mathcal{S}$ using Algorithm 3 or Algorithm 4
6: **end procedure**

Paper C and the application to scenarios with continuously oncoming vehicle is considered in Paper D.

5.3.4 Approximate solution of OCP (4.8)

A practical implementation of the two stage-procedure of Section 4.4.1 using the MIQP-based heuristic is summarized in Algorithm 5. Here, $\hat{\mathcal{D}}_i = \{T_i^{[0]}, \nabla V_i, \tilde{H}_i, g_i, \nabla g_i\}$ is the data required by the central node from each vehicle to construct (5.19). This can be constructed in parallel on board each vehicle, and the results sent to a central network node where the MIQP is solved.

5.4 Summary

In this chapter we discussed computational methods for the two-stage solution strategy discussed in the previous chapter. We introduced an OC-based heuristic to find the crossing order, which relies on the solution of an

MIQP. We also introduced two algorithms for the fixed-order problem which both relies on parallelization and distribution of most computation to the vehicles. The first, based on SQP, is applicable to problems without rear-end constraints whereas the second, based on an Interior Point method, applies to general problems. We illustrated their difference through an example and discussed their communication characteristics. We also presented data from a practical application of the SQP-based algorithm to a real scenario where wireless communication links were used. The results illustrated that while benefits are obtained through parallelized computation, the necessary communication requires a substantial amount of time. In particular, severe degradation of the communication links results in unreasonably long solve-times for the algorithm. Due to the lower communication demand per iterate and fewer total iterations required, the SQP-based method has a practical advantage in cases where it is applicable. It is therefore relevant to investigate extensions of this method to scenarios with more than one vehicle on each lane.

In the next chapter, we discuss the application of the two-stage solution strategy to model predictive control, and present results from both simulation-based and experimental evaluations.

Chapter 6

Closed Loop Control via MPC

Intersection scenarios are highly dynamic, with continuously arriving and leaving vehicles. In a practical setting, the available measurements are typically uncertain, the motion model is often not exact and various external disturbances could be present. In this chapter, we therefore discuss introduction of feedback through the use of OCP (4.8) in a Model Predictive Controller (MPC)[1]. We first investigate the performance of MPC-based coordination through simulation in Section 6.1 and thereafter discuss an experimental validation of a simple scenario in Section 6.2. Finally, the use of nonlinear motion models and economic objective function is discussed in Section 6.3, where we also present results from the application of an approximation method from the literature.

6.1 Performance

In this section we summarize the simulation results detailed in Paper D, where the two-stage strategy of Section 4.4.1 is applied in a receding horizon fashion using the MIQP heuristic of Section 5.3 and fixed-order problem (4.10). We apply the controller to scenarios with continuous traffic and compare the performance with that of current regulatory mechanisms and two alternative coordination controllers. The purpose is to investigate (i) the benefit of automated coordination, and (ii) the impact of increased complexity in the automated coordination controllers.

Since the objective is to study the performance characteristics of the proposed method, the numerous practical aspects are not considered. In particular, the fixed-order problems (4.10) are solved with a fully centralized interior-point algorithm, but we note that the same solution would be obtained in a partly centralized setting using Algorithm 4.

[1] A brief introduction to MPC is given in Section 2.4.2.

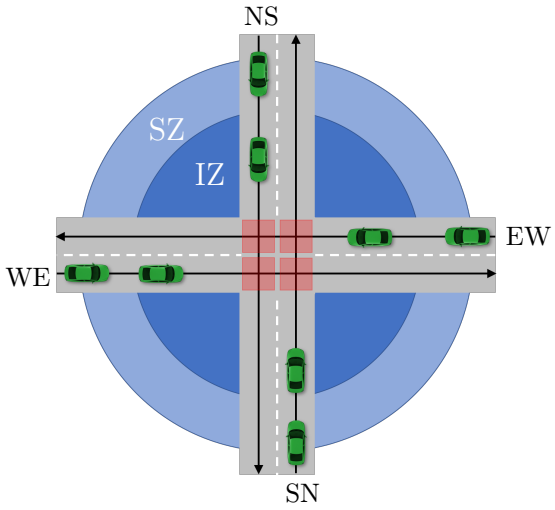


Figure 6.1: Scenario for the performance evaluation.

6.1.1 Scenario description

The evaluation is performed on the two-road intersection shown in Figure 6.1, with one lane in each direction (East-West (EW), West-East (WE), North-South (NS) and South-North (SN)). The vehicles are generated at the beginning of the *Scenario Zone* (SZ) and the coordination is performed on vehicles in the *Intersection Zone* (IZ). These begin at 350 and 200 m before the intersection, respectively. We investigate the performance for arrival rates ranging from 1000 to 2500 vehicles per lane per hour (4000-10000 per hour total), for a period of 15 minutes. The scenario is heterogeneous and consists of 10% Trucks and 90% Passenger Cars, where the type is drawn randomly on generation.

6.1.2 Evaluated controllers

MIQP/FO The *Mixed-Integer Quadratic Programming/Fixed order* controller is the two-stage scheme discussed in Section 4.4.1. At each t_k, it

1. computes the crossing order $\mathcal{S}$ using the MIQP-heuristic of Section 5.3,
2. solves the fixed order problem (4.10) under $\mathcal{S}$ for the control commands,

for all vehicles in the IZ.

FCFS/FO The *First-Come-First-Served/Fixed order* controller operates similarly to the MIQP/FO, but decides the crossing order $\mathcal{S}$ using a First-Come-First-Served heuristic.

Sequential The *Sequential* controller is of the type discussed in Section 4.4.2. The vehicles decide sequentially what action to take, and enforce collision avoidance only with previous vehicles in a decision order. The latter is constructed using a FCFS heuristic.

Traffic Light The *Traffic Light* controller is similar to standard traffic light. The vehicles on the two roads are allowed to cross the intersection in an alternating fashion, corresponding to red/green phases. The period-time is set to 20 s.

Overpass The *Overpass* "controller" corresponds to a physical separation of the two roads. All vehicles drive at constant speed.

We thus consider two representatives of current coordination mechanisms (Traffic Light, Overpass) and three automated controllers (MIQP/FO, FCFS/FO, Sequential). The latter differ in complexity, ranging from the Sequential (least complex) to the MIQP/FO (most complex), and share fundamental features with the *Sequential* and *Simultaneous* OC-based Algorithms in the literature (c.f. Section 1.3.2). The Traffic Light and Overpass controllers are similarly two extremes on a spectrum ranging from conservative (Traffic Lights) to risky (Overpass).

6.1.3 Results

All controllers employ the double-integrator prediction model (3.5) with bounded acceleration, together with the objective

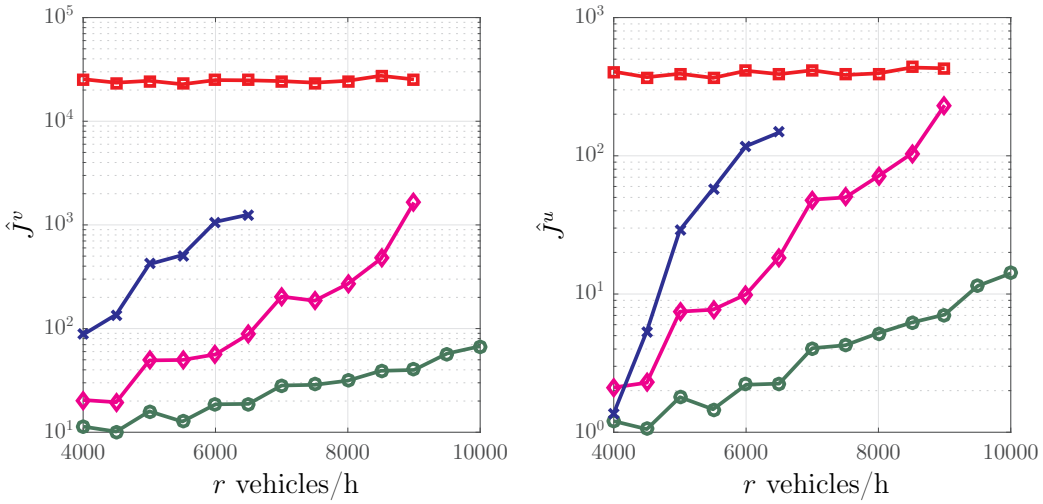
$$J_i(w_i) = Q_i^f(v_{i,N} - v_i^{\mathrm{r}})^2 + \sum_{k=0}^{K-1} Q_i(v_{i,k} - v_i^{\mathrm{r}})^2 + R_i u_{i,k}^2, \tag{6.1}$$

where v_i^r is a reference speed. We sample time $\Delta t = 0.2$, horizon $K = 100$, $Q_i = R_i = 1$ and $v_i^{\mathrm{r}} = 70$ km/h, but scale the objective with the vehicle mass m_i (20000 kg for trucks, 1700 kg for Cars).

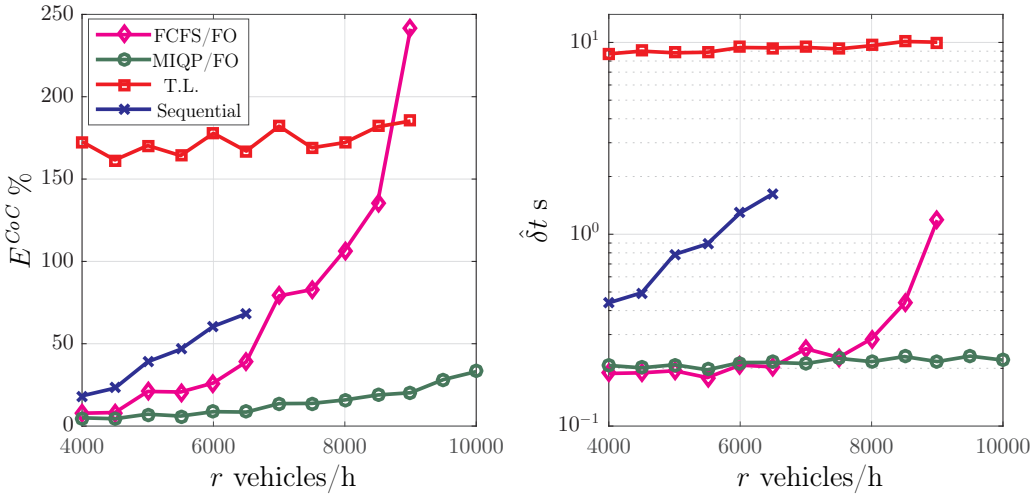
The performance in terms of the two components of (6.1) is computed as

$$\hat{J}^v = \frac{1}{|\mathcal{N}^{\mathrm{c}}|} \sum_{i \in \mathcal{N}^{\mathrm{c}}} \sum_{k=k_i^{\mathrm{e}}}^{k_i^{\mathrm{d}}} m_i Q_i (v_{i,k} - v^{\mathrm{r}})^2, \tag{6.2}$$

$$\hat{J}^u = \frac{1}{|\mathcal{N}^{\mathrm{c}}|} \sum_{i \in \mathcal{N}^{\mathrm{c}}} \sum_{k=k_i^{\mathrm{e}}}^{k_i^{\mathrm{d}}} m_i R_i u_{i,k}^2. \tag{6.3}$$



(a) Components of the quadratic objective



(b) Energy (% increase over the Overpass solution) and Travel-Time delay

Figure 6.2

Here, $\mathcal{N}^{\mathrm{c}}$ collects the indices of vehicles that have crossed the intersection during the simulation, where k_i^{e} and k_i^{d} are the time instant of entry and departure of the SZ. Moreover, we assess the effect on average energy consumption ("Cost of Coordination", $\hat{E}^{\mathrm{CoC}}$) and average travel time delay, $\hat{\delta t}$, which both are defined in relation to the Overpass solution. Here, $\hat{E}^{\mathrm{CoC}}$ is computed by the energy needed by an EV (using model (3.2)) to follow the trajectories generated by the controllers.

The results are shown in Figure 6.2. The first thing to note is the lack of data-points for the Traffic Light, and FCFS/FO controller for arrival rates $r > 9000$ vehicle/h, and the lack of data-points for the Sequential scheme for $r > 6500$ vehicles/h. In these cases, the simulation was terminated due to

congestion (the Sequential scheme) or performance degradation (FCFS/FO).

As can be seen, all automated controllers significantly outperforms the Traffic-Light fo low to medium traffic intensities. This is mainly a consequence of the different controllers' ability to coordinate the vehicles through the intersection without forcing them to stop. We highlight the performance of the MIQP/FO controller in particular. As can be seen, very high traffic intensities ($r = 10000$) can be handled without "paying" more than 40% energy than simply driving at the initial velocity v^{e} or incurring more than 0.1 s delay on average.

Moreover, note that the performance of the different automated controllers improves with added complexity: it is significantly better in all performance metrics to jointly optimize the trajectories under a First-Come-First Serve policy (the FCFS/FO) than to decide both the order and the trajectories sequentially. Both are in turn worse than the optimization of both the crossing order and the trajectories (the MIQP/FO). The latter is true for all performance metrics except the travel-time delay, where the FCFS/FO and MIQP/FO perform virtually the same for $r \leq 8000$ vehicles/h, with close to zero average delay.

An intuitive understanding of the performance differences can be obtained by examining Figure 6.3, which shows the development of the average, maximum and minimum velocity in the scenario. As the figure shows, the variations in velocity decrease with increasing controller complexity, where we in particular note the presence of stationary vehicles in the Traffic Light case. Since all velocity changes ultimately lead to increased energy consumption, this gives the results shown in Figure 6.2(b).

An animation showing the performance of the MIQP/FO controller can be found at [91].

6.2 A bi-level MPC for simplified problems

In this section we introduce an MPC where feedback is split over two levels, detailed in Paper A. The controller is derived from the decomposition of Section 5.1 and applicable to problems without RECA constraints. We recall that such problems can be split into the timeslot allocation problem

$$\min_{T} \quad \sum_{i=1}^{N} V_i(T_i, x_{i,k}) \tag{6.4a}$$

$$\text{s.t.} \quad g_i(T_i, x_{i,k}) \leq 0, \qquad i \in \mathcal{N}, \tag{6.4b}$$

$$\left(t_{r,i}^{\text{out}} \leq t_{r,j}^{\text{in}}\right) \vee \left(t_{r,j}^{\text{out}} \leq t_{r,i}^{\text{in}}\right), \qquad (i,j,r) \in \mathcal{C}_S, \tag{6.4c}$$

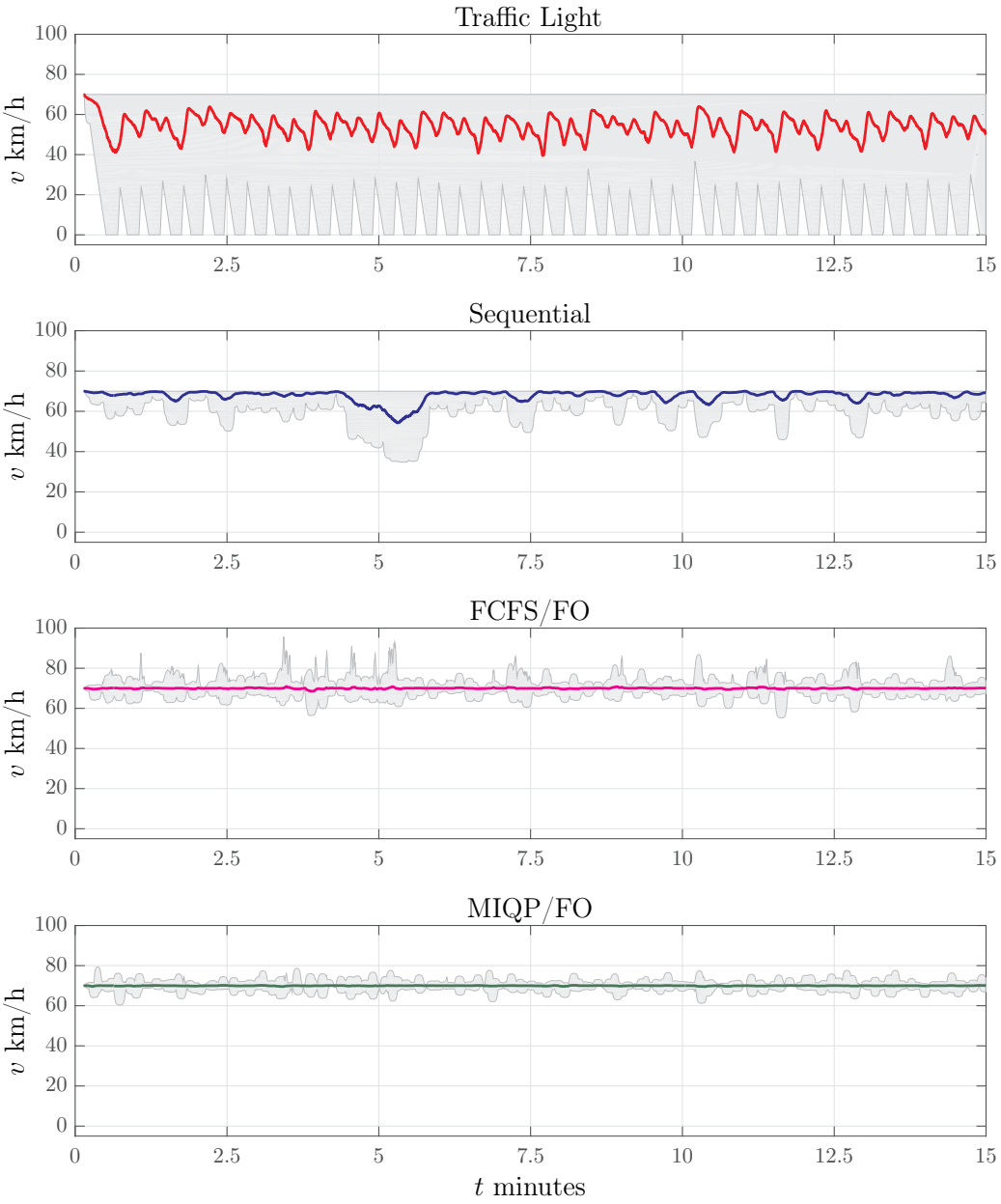


Figure 6.3: Average velocity (colored lines) and velocity intervals (gray surface) in a scenario with $r = 6000$ vehicles/hour.

and the vehicle problems

$$V_i(T^*_{i,k}, x_{i,k}) = \min_{\tilde{w}_{i,k}} \quad J_i(\tilde{w}_{i,k}) \tag{6.5a}$$

$$\text{s.t.} \quad \tilde{x}_{i,k} = x_{i,k}, \tag{6.5b}$$

$$\tilde{x}_{i,k+l+1} = F_i(\tilde{x}_{i,k+l}, \tilde{u}_{i,k+l}, \Delta t), \qquad l \in \mathcal{I}_{K-1}, \tag{6.5c}$$

$$h_i(\tilde{x}_{i,k+n}, \tilde{u}_{i,k+l}) \leq 0, \qquad l \in \mathcal{I}_{K-1}, \tag{6.5d}$$

$$p_i(t^{\text{in}}_{r,i}, \tilde{w}_{i,k}) = p^{\text{in}}_{r,i}, \qquad r \in \mathcal{R}_i, \tag{6.5e}$$

$$p_i(t^{\text{out}}_{r,i}, \tilde{w}_{i,k}) = p^{\text{out}}_{r,i}, \qquad r \in \mathcal{R}_i. \tag{6.5f}$$

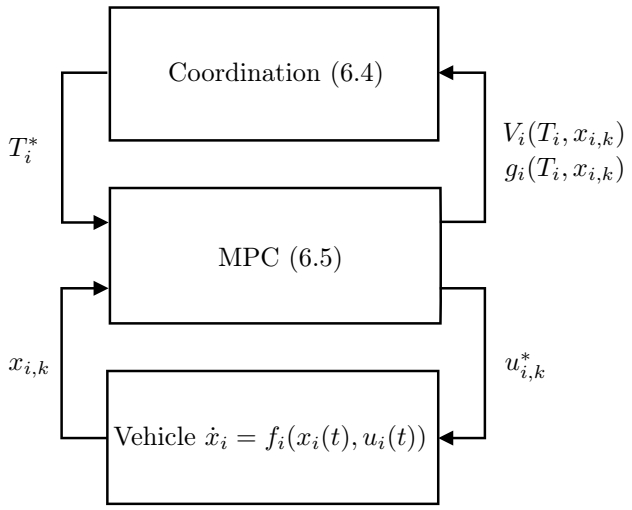


Figure 6.4: Illustration of the bi-level controller for intersection coordination.

Here, $\tilde{w}_{i,k} = (\tilde{x}_{i,k}, \tilde{u}_{i,k}, \ldots, \tilde{x}_{i,k+K-1}, u_{i,k+K-1}, \tilde{x}_{i,k+K})$ collects the predicted state and control at time t_k and $g_i(T_i, x_{i,k})$ is defined as in Section 5.1.1, with the dependence on $x_{i,k}$ made explicit. Problem (6.4) gives the predicted optimal and non-overlapping timeslots at t_k, $T_k^* = (T_{1,k}, \ldots, T_{N,k}^*)$, given $V_i(T_i, x_{i,k})$ and $g_i(T_i, x_{i,k})$ $\forall i \in \mathcal{N}$, whereas Problem (6.5) gives the predicted optimal control command at t_k, $u_{i,k}^*$ given $T_{i,k}^*$ and vehicle state $x_{i,k}$.

The optimal timeslot $T_{i,k}^*$ is thereby a function of the state of all vehicles $\bar{x}_k = (x_{1,k}, \ldots, x_{N,k})$ through $V_i(T_i, x_{i,k})$, $g_i(T_i, g_{i,k})$ and the solution of (6.4b). Therefore, the feedback law of vehicle i is a function of $\bar{x}$ on the form

$$u_{i,k}^*(\bar{x}_k) = \tilde{u}_{i,k}(x_{i,k}, T_{i,k}^*(\bar{x}_k)), \tag{6.6}$$

where $\tilde{u}_{i,k}$ is the first element of the predicted input sequence returned by (6.5). That is, feedback is due both to the state of the vehicle itself and the states of all other vehicles through $T_{i,k}^*$. The controller thus has the bi-level structure illustrated in Figure 6.4, where $T_{i,k}^*$ is the result of the *intersection*-level control loop and $u_{i,k}^*$ through the *vehicle level* control loop.

6.2.1 Persistent feasibility and stability

To be useful in practice, conditions for persistent feasibility and stability must be established for the bi-level MPC. To this end, we introduce the *uncoordinated vehicle MPC*, based on the NLP

$$\min_{\tilde{w}_{i,k}} \quad J_i(\tilde{w}_{i,k}) \tag{6.7a}$$

$$\text{s.t.} \quad \tilde{x}_{i,k} = x_{i,k}, \tag{6.7b}$$

$$\tilde{x}_{i,k+l+1} = F_i(\tilde{x}_{i,k+l}, \tilde{u}_{i,k+l}, \Delta t), \qquad l \in \mathcal{I}_{K-1}, \tag{6.7c}$$

$$h_i(\tilde{x}_{i,k+l}, \tilde{u}_{i,k+l}) \leq 0. \qquad l \in \mathcal{I}_{K-1}, \tag{6.7d}$$

The uncoordinated MPC thus gives the optimal control command for vehicle i when no interactions with other vehicles are considered.

While the uncoordinated vehicle MPC can be designed to be both persistently feasible and stabilizing, the intersection level control loop must be accounted for in the bi-level controller. A thorough analysis is performed in Paper A, where the following results are established:

Proposition 6.1. *Persistent feasibility. If the uncoordinated vehicle MPC is persistently feasible and a feasible solution exists to* (6.4) *and* (6.5) *for the initial state $\bar{x}_0$, the bi-level controller* (6.6) *is persistently feasible.*

Theorem 6.1. *Stability. If the uncoordinated vehicle MPC stabilizes a vehicle, and* (6.5) *is well posed for all feasible $(T_i, x_{i,k})$, the bi-level controller is stable and allows all vehicles to cross the intersection.*

Corollary 6.1. *Proposition 6.1 and Theorem 6.1 hold when*

- *T is computed once and is thereafter fixed*
- *T is updated by solving* (6.4) *for a fixed crossing order (c.f* (5.2)*)*
- *The crossing order $\mathcal{S}$ is updated through a heuristic, and T is found by solving* (6.4) *for a fixed crossing order, provided that*

$$\sum_{i=1}^{N} V_i(T_i^*(x_{i,k+1}), x_{i,k+1}) < \sum_{i=1}^{N} V_i(T_i^*(x_{i,k}), x_{i,k}). \tag{6.8}$$

Corollary 6.1 has two important consequences: First, it enables the use of, e.g., the MIQP-based heuristic of Section 5.3, provided its output is monitored. Second, it enables separate update frequencies in the two control-loops, whereby T can be computed less frequently than the vehicle commands. Given that solving the timeslot problem (6.4) over the wireless channel might require more time than that commonly used for vehicle control, the latter is relevant for practical applications. Finally, Corollary 6.1 enables event-triggered re-computation of T, where (6.4) is solved when a triggering criteria is satisfied (e.g. detection of a significantly large perturbation or the entry of a new vehicle to the intersection area).

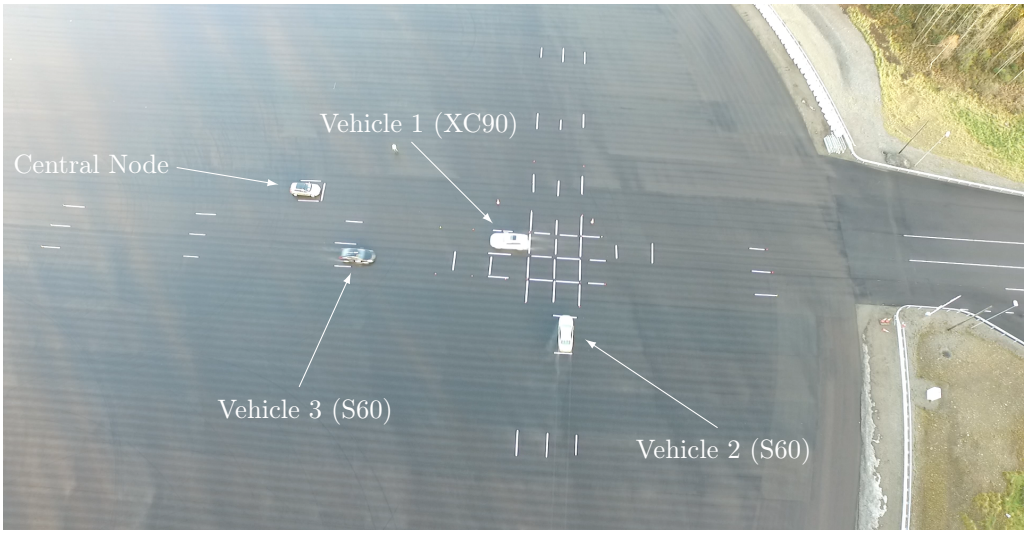


Figure 6.5: Aerial photo of an experiment. Video available at [86].

6.2.2 Experimental validation

To assess the practical performance of the bi-level controller we performed experiments with real vehicles. These were executed on the AstaZero Test-Track outside Gothenburg, Sweden, and involved the three Volvo vehicles shown in Figure 6.5. All vehicles were equipped with real-time computers, vehicle-to-vehicle communication equipment, RTK-GPS receivers and Inertial Measurement Units (IMU), and used Extended Kalman Filters (EKF) to fuse the information. The set-up is illustrated in Figure 6.6.

The intersection-level control loop was closed by solving (6.4) for a fixed crossing order, using the SQP-algorithm of Section 5.1. At each update of the timeslots T, Algorithm 3 was executed over the wireless communication links. The information required to build the SQP sub-problem (5.11) at each iterate, $\mathcal{D}_i^{[k]}$, was computed on-board the vehicles and sent to a central network node, consisting of a centrally placed laptop with a communication access point. We employed the simple double integrator (DI) (3.5) with acceleration bounds as a prediction model and the objective (6.1) for different values of v_i^r, Q_i and R_i. This rendered the MPC problem (6.5) a QP, which was solved on-board the vehicles using the QP-solver HPMPC [87].

Finally, the intersection level control-loop was closed with a sampling time of 3 seconds due to hardware limitations (see the discussion in Paper B), and the vehicle level control loops were closed with $\Delta t = 0.1$ s.

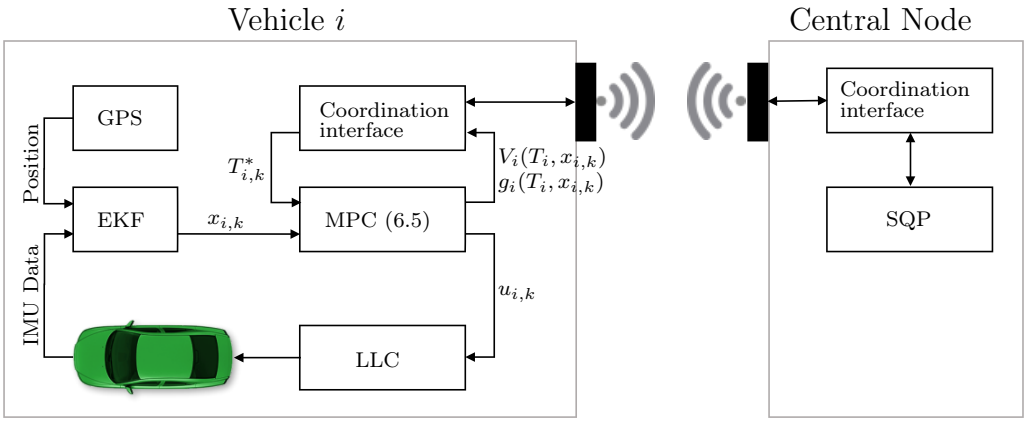
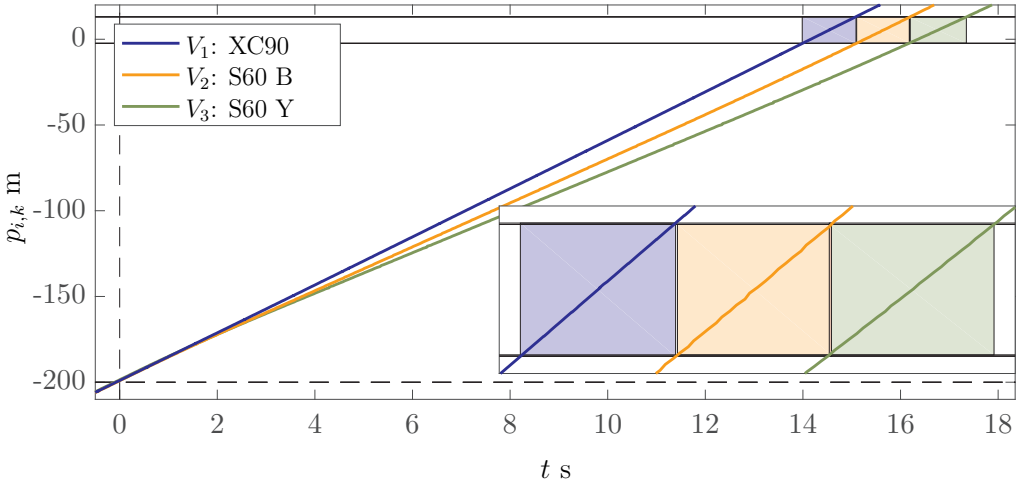


Figure 6.6: Schematic illustration of the test-setup. The Low-Level Controller (LLC) translates $u_{i,k}$ to appropriate powertrain commands .

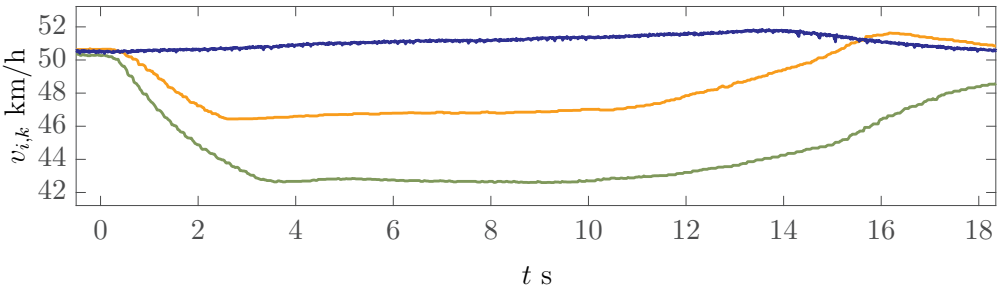
Results

In total, over 80 successful tests were performed, where we investigated different initial conditions and parameter settings. Data from one instance is shown in Figures 6.7(a)-6.7(d), from a scenario where all vehicles were initialized 200 m before the intersection, traveling at $v_i^{\mathrm{r}} = 50$ km/h, so that a three-way collision would occur if no action was taken. As can be seen in Figure 6.7(a), the trajectory for each vehicle was adjusted so that only one vehicle was inside the intersection at a given time. The corresponding control commands from the intersection and vehicle level controllers are shown in Figure 6.7(c) and Figure 6.7(d). Note the small changes made to T as the vehicles approach the intersection. These are made in response to perturbations arising from e.g. prediction model inaccuracies and measurement noise.

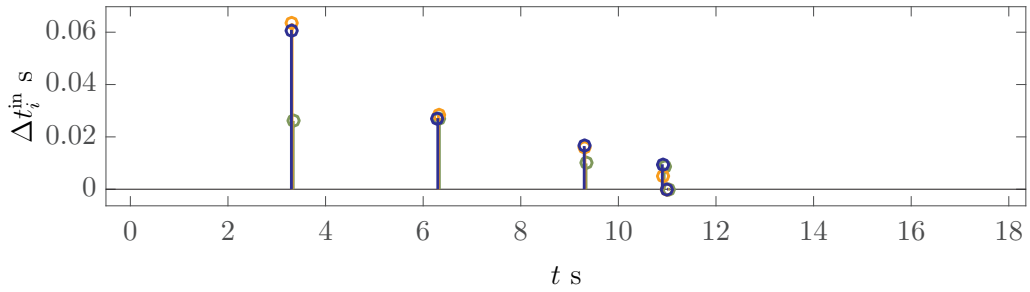
To investigate the ability to handle large perturbations, we performed tests where the brake-pedal was pressed in the first vehicle, causing a temporary suspension of the automated control of that vehicle. Data from such an experiment is presented in Figure 6.8, where the gray slabs mark the time during which the brake pedal was pressed. The immediate effect of the perturbation can be seen in Figure 6.8(a), where the vehicle level controller of the first vehicle tries to compensate the braking action by increasing the commanded acceleration. Even though the vehicle eventually catches up to the control command at $t \approx 6$ s, it does so with significant effort. The reaction of the intersection-level control loop is seen at $t \approx 7$ seconds, where the timeslots of all vehicles are postponed around 0.4 s. Note the impact this has on the lower level controllers: when the new timeslots arrive at the vehicles, the control command of the first vehicle is reduced, while that of the second vehicle is increased. This demonstrates the ability of the bi-level MPC to distribute the effort optimally between the vehicles.



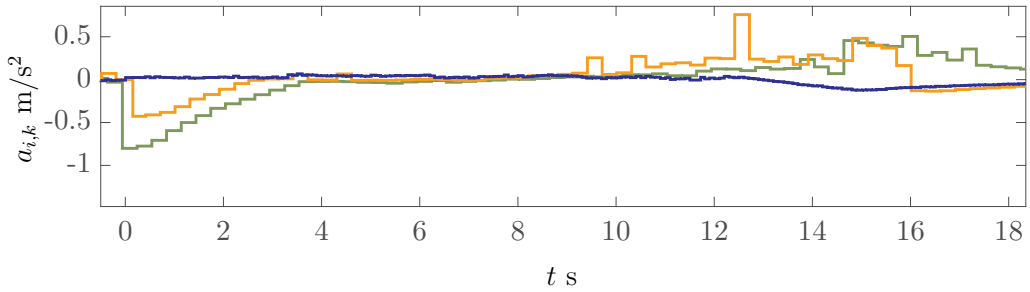
(a) Position trajectories. The two horizontal lines represents p_i^{in} and p_i^{out}, and the colored boxes shows the time at which the vehicles are inside the intersection.



(b) Velocity

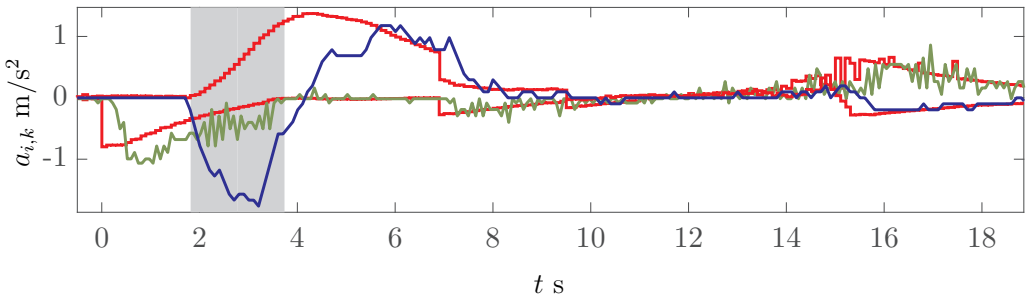


(c) Changes in timeslots between executions of the intersection level control loop. Not to be confused with the search direction in the SQP scheme.

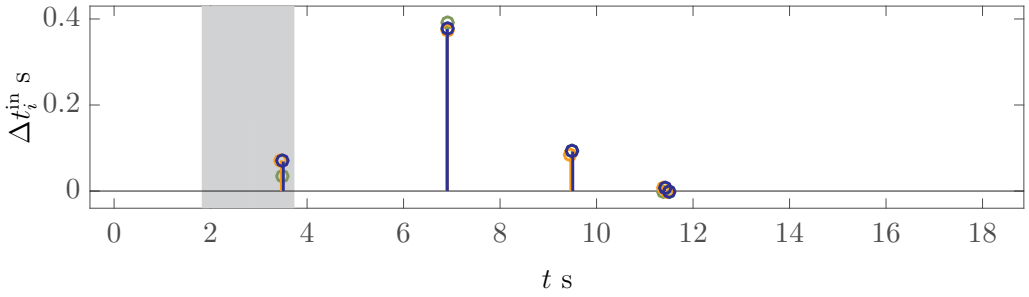


(d) Control commands in the vehicle level control loop. The noisy behavior between $t = 12$ and $t = 17$ is due to problems with the positioning system in two of the vehicles. The interested reader can find a longer discussion of the issue in Paper A

Figure 6.7: Data from an experimentally validated scenario.



(a) Commanded and acceleration in red and green/blue respectively. The delay in the intersection-level's response is discussed further in Paper B.



(b) Changes in timeslots between updates in the intersection level control loop

Figure 6.8: Data from an experimental scenario. The driver of the vehicle corresponding to the blue trajectory pressed the brake pedal during the time marked by the gray slab, temporarily suspending automation. Only data from two vehicles is shown for clarity.

Robustness aspects

To avoid side-collisions inside the intersection it was necessary the vehicles were required to only occupy the intersection within their given timeslots. This was enforced by the vehicle controllers (6.5) through constraints (6.5e),(6.5f). Interestingly, the vehicles showed a remarkable ability to satisfy these constraints. As shown in Figure 6.9, the observed constraint violations were consistently small.

The consistency indicates that small violations of constraints that mandate a vehicle to be at a specific position at a specific time is a property of the closed-loop system. As a consequence, robust coordination could be achieved through constraint tightening. In particular, by replacing constraints (6.5e),(6.5f) with

$$p_i(t_{r,i}^{\text{in}}, \tilde{w}_{i,k}) = p_{r,i}^{\text{in}} - \Delta, \qquad r \in \mathcal{R}_i, \tag{6.9a}$$

$$p_i(t_{r,i}^{\text{out}}, \tilde{w}_{i,k}) = p_{r,i}^{\text{out}} + \Delta, \qquad r \in \mathcal{R}_i, \tag{6.9b}$$

constraint violations $\epsilon < \Delta$ will not cause the actual vehicles to violate the SICA constraints (6.5e),(6.5f). That is, by artificially making the intersection "larger" the problem can be avoided.

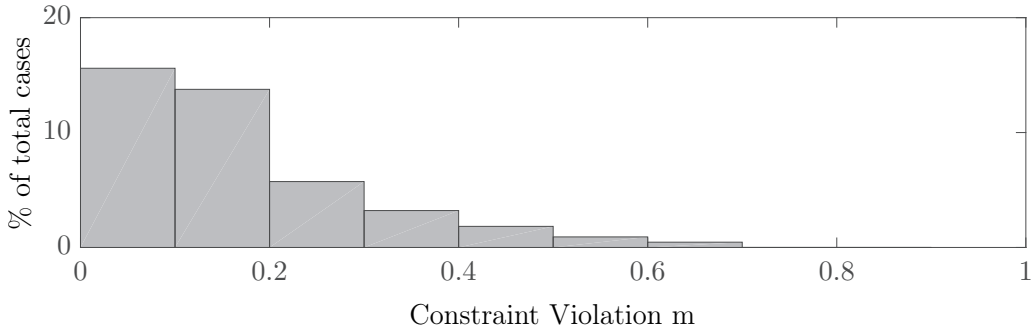


Figure 6.9: Constraint violations in 450 closed-loop evaluations of (6.5e),(6.5f)

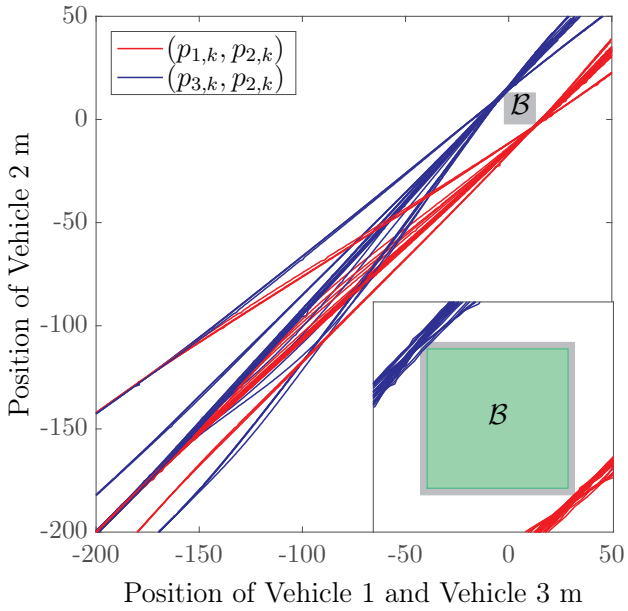


Figure 6.10: Configuration space trajectories. The gray rectangle $\mathcal{B} = \left([p_1^{\text{in}}, p_1^{\text{out}}] \times [p_2^{\text{in}}, p_2^{\text{out}}]\right) \cup \left([p_3^{\text{in}}, p_3^{\text{out}}] \times [p_2^{\text{in}}, p_2^{\text{out}}]\right)$ corresponds to the intersection. The green square illustrates an intersection where the width is 0.7 m smaller in all dimensions.

An illustration of this approach is given in Figure 6.10, which shows the configuration-space trajectories from 34 experiments together with a large and small representation of the intersection (c.f. Figure 3.4(b)). While the former was that used in the experiments, the figure shows it as the result of (6.9), with $\Delta = 0.7$ m. The "real" intersection in this case corresponds to the green area in the figure. As can be seen, none of the trajectories are inside at any time, whereby the SICA constraints are satisfied by the closed loop system.

Note that a real application might require large margins for other reasons than inaccurate constraint-enforcement (e.g human psychology), in which case the small constraint violations observed would be of little significance.

Figure 6.11: Photo from an experimental run. The white vehicle applies the friction-brake immediately after leaving the intersection (marked with white lines).

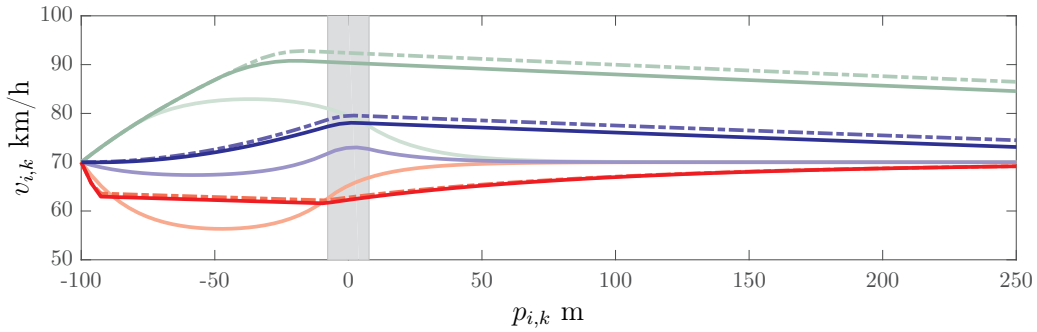
6.3 Use of economic objectives

The experiments made an undesired effect of the DI model and objective apparent. In cases where a vehicle crossed the intersection with $v_{i,k} > v^r$ (e.g, the blue trajectory in Figure 6.7(b)), it was commanded to reduce its speed immediately after $p_{i,k} > p_i^{\text{out}}$, due to the $(v_{i,k} - v_i^{\text{r}})^2$ term in (6.1). As illustrated in Figure 6.11 this sometimes resulted in use of the friction brakes. However, since energy thereby is wasted, this is not a desirable behavior. For this reason, we proposed the following objective function in Paper E

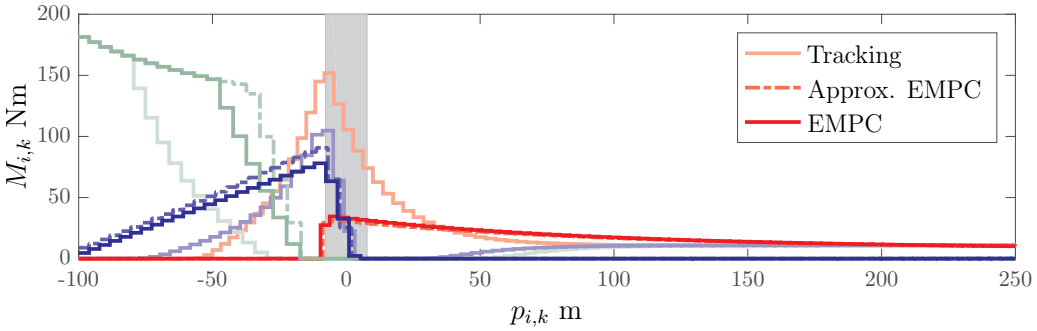
$$\ell_{i,k}^{\text{Ev}}(v_{i,k}, u_{i,k}) = \underbrace{\int_{t_k}^{t_{k+1}} \frac{M_{i,k}\omega_i(t)}{\eta_i(M_{i,k}, \omega_i(t))} \mathrm{d}t}_{\ell_{i,k}^{\text{E}}(x_{i,k}, u_{i,k})} + w_i^{\text{v}} \underbrace{\int_{t_k}^{t_{k+1}} -\mathrm{v}_i(t)\mathrm{d}t}_{\ell_{i,k}^{\text{v}}(x_{i,k}, u_{i,k})}, \tag{6.10}$$

together with the nonlinear EV motion model of Section 3.2, where $\eta_i : \mathbb{R}^2 \mapsto [0, 1]$ is the efficiency map of the motor and $w_i^{\text{v}} > 0$ is a scalar parameter. We recall that for the EV model, the inputs are the motor torque $M_{i,k}$ and the friction-brake force $F_{i,k}^{\text{b}}$, and $\omega_i(t)$ is the electric motor speed. Minimization of (6.10) thereby includes energy minimization ($\ell_{i,k}^{\text{E}}(x_{i,k}, u_{i,k})$) and travel-time minimization ($\ell_{i,k}^{\text{v}}(x_{i,k}, u_{i,k})$), in a trade-off determined by w_i^{v}. Due to resistive forces, all speed-reductions implies that energy must be inserted to the system at a later time. Since (6.10) cannot be decreased by the friction brakes, these will therefore only be employed to enforce feasibility.

However, use of indefinite, *Economic* objectives such (6.10) can make the NLPs involved harder to solve, and prevents application of the standard stability proof for MPC. Due to this, we employed the method of [92] in



(a) Position-Velocity



(b) Position-Torque

Figure 6.12: Position-Velocity and Position-Torque plots from a scenario with Economic, approximate Economic and Tracking objectives. The the gray slab is the intersection.

Paper E, and selected a convex quadratic objective

$$\ell_{i,k}^{\tilde{Q}}(v_{i,k}, u_{i,k}) = \begin{bmatrix} v_{i,k} - v_i^{\mathrm{ss}} \\ u_{i,k} - u_i^{\mathrm{ss}} \end{bmatrix}^\top \tilde{Q} \begin{bmatrix} v_{i,k} - v_i^{\mathrm{ss}} \\ u_{i,k} - u_i^{\mathrm{ss}} \end{bmatrix} + \nabla_{(v,u)} \ell_{i,k}^{\mathrm{Ev}}(v_i^{\mathrm{ss}}, u_i^{\mathrm{ss}}) \begin{bmatrix} v_{i,k} - v_i^{\mathrm{ss}} \\ u_{i,k} - u_i^{\mathrm{ss}} \end{bmatrix} \tag{6.11}$$

such that the resulting MPC control law is first-order equivalent near v_i^{ss} to that resulting from (6.10). Here, $\tilde{Q}$ is obtained by solving the Semi-Definite Program detailed in Paper E and v_i^{ss}, u_i^{ss} are the solutions to the steady state problem

$$\min_{v_i^{\mathrm{ss}}, u_i^{\mathrm{ss}}} \ \ell^{\mathrm{Ev}}(x_i^{\mathrm{ss}}, u_i^{\mathrm{ss}}) \quad \text{s.t. } v_i^{\mathrm{ss}} - F_{i,v}(v_i^{\mathrm{ss}}, u_i^{\mathrm{ss}}) = 0, \ h(x_i^{\mathrm{ss}}, u_i^{\mathrm{ss}}) \leq 0, \tag{6.12}$$

where $F_{i,v}$ is the velocity component of F_i.

Example results from use of objectives (6.10) and (6.11) is shown in Figure 6.12. The results obtained with the "tracking" objective

$$\ell_{i,k}^{Q}(x_{i,k}, u_{i,k}) = \int_{t_k}^{t_{k+1}} Q_i(\mathrm{v}_i(t) - v_i^{\mathrm{ss}})^2 + R_i(\mathrm{a}_i^{\mathrm{lon}}(t))^2 \mathrm{d}t, \tag{6.13}$$

i.e., the EV "version" of objective (6.1) are also shown for comparison. For (6.10), (6.11), w_i^{v} selected such that $v_i^{\mathrm{ss}} = 70$ km/h, and for (6.13) $Q_i = R_i = 1$. The approximating quality of the MPC using (6.11) can be seen in the speed profiles and control commands, which are highly similar to their counterparts using (6.10). Figure 6.12 also show that the economic MPC formulation avoids the issue observed in the experiments. Instead of braking, the vehicles with $v_{i,k} > v_i^{\mathrm{ss}}$ for $p_{i,k} > p_i^{\mathrm{out}}$ cease to supply energy to the motor and let the resistive forces reduce their speed. Note also the qualitative difference to the results obtained with (6.13).

For the intersection scenario, the method of [92] thus works well. This means that almost economic performance can be obtained at no extra expense compared to standard tracking objectives.

Finally, note the scenario shown is rather "hard": 3 vehicles start at 70 km/h at $\sim$ 100 m from the intersection. This forces all vehicles to deviate significantly from v_i^{ss}, thus worsening the approximation between the controllers based on (6.10) and (6.11). In "easier" scenarios, the difference is less pronounced.

6.4 Summary

In this chapter we discussed the application of the two-stage strategy of Chapter 4 to receding horizon control. We demonstrated the efficiency of the approach when the MIQP-based crossing order heuristic of Section 5.3 was used through simulation. The results indicate that the proposed approach outperforms both traffic-lights and two other automated coordination algorithms that share fundamental characteristics with the Simultaneous and Sequential/Parallel approaches discussed in Chapter 1.3.2. We demonstrated the practical applicability of the MPC through experiments and discussed robust collision-avoidance via constraint-tightening. The results show that remarkably good control performance can be achieved even with simple kinematic prediction model, large perturbations, poor positioning and communication deficiencies. Motivated by some undesired effects observed during the experiments, we also discussed the use of non-linear motion models and economic objectives. We showed that an automatic tuning procedure from the literature can be applied to the economic coordination MPC with good results. As a consequence, almost economic performance can be obtained by solving problems not harder than those addressed during the experiments.

Chapter 7

Summary of Appended Papers

7.1 Paper A

R. Hult, M. Zanon, S. Gros and P. Falcone, "Optimal Coordination of Automated Vehicles at Intersections: Theory and Experiment", to appear in *IEEE Transactions on Control Systems Technology*

In this paper, we introduce a Model Predictive Controller for coordination of automated vehicles at intersections, and present experimental results. The MPC is based on a primal decomposition of an optimal control formulation of the problem and consists of two levels. At the upper level, non-overlapping intersection occupancy timeslots are computed by solving a Nonlinear Program. At the lower level, the vehicle control commands are computed such that each vehicle only is inside the intersection during its prescribed timeslot. We derive conditions for persistent feasibility and stability of the controller, and discuss application of constraint-tightening to handle positioning uncertainties. We present experimental results, where the controller is applied to three real vehicles using an optimization algorithm where most computations are performed on-board the vehicles. The results demonstrate the efficacy of our approach, and show that the proposed bi-level MPC performs well under adverse conditions including both positioning errors, inefficient implementation and communication deficiencies.

7.2 Paper B

R. Hult, M. Zanon, G. Frison, S. Gros and P. Falcone, "Experimental Validation of a Semi-Distributed SQP Method for Optimal Coordination of Automated Vehicles at Intersections", submitted to *Optimal Control Applications and Methods*

In this paper, we detail a primal decomposition-based Sequential Quadratic Programming (SQP) algorithm for the optimal coordination at intersections under fixed crossing orders We discuss results from an experimental validation of the SQP algorithm, and highlight the algorithm's performance in terms of computational time and communication requirements. Additionally, we introduce a Real-Time Iteration like reformulation of the MPC from Paper A, and present results from its validation in experiment. The first part of this paper contains the same material as the conference paper [44].

We remark that the experimental results reported in both Paper A and B were collected during the same experimental campaign, due to which their content partly overlap. The papers should be seen as companions with focus on different parts of the same problem, taking a closed-loop control perspective in Paper A, and an optimization perspective in Paper B.

7.3 Paper C

R. Hult, M. Zanon, S. Gros and P. Falcone, "An MIQP-based Heuristic for Optimal Coordination of Vehicles at Intersection", presented at the *Conference on Decision and Control*, 2018

In this paper, we detail a heuristic for selecting the order in which the vehicles cross the intersection. The heuristic is derived from an optimal control formulation of the problem, and constructed in two steps. First, collision avoidance between vehicles on the same lane is ignored and the primal decomposition of Paper B is used to derive an approximate mixed-integer nonlinear program (MINLP) representation of the problem. Second, tools from sensitivity analysis are used to derive a mixed integer quadratic program (MIQP) approximation of the MINLP. We propose to use the MIQP heuristic in a two-stage, approximate solution scheme for the problem, where 1) the MIQP is solved for the crossing order and 2) a nonlinear program is solved for the vehicle state and control trajectories. We demonstrate the efficacy of our approach through simulation and compare it with the natural first-come-first-served heuristic.

7.4 Paper D

R. Hult, M. Zanon, S. Gros, H. Wymeersch and P. Falcone, "Optimization-based Coordination of Connected, Automated Vehicles at Intersection", submitted to *Vehicle System Dynamics*

In this paper, we study the closed-loop performance of an MPC for intersection scenarios with continuously oncoming traffic, and derive conditions for persistent feasibility in this context. The controller is based on repeated execution of the two-stage approximation scheme proposed in Paper C. We compare the controller to a physical separation of the roads and to traffic lights to assess the difference to current coordination mechanisms. We also compare the performance to that of two other optimal control-based coordination schemes to assess the impact of increasing controller complexity. These share fundamental properties with many other algorithms discussed in the literature and are 1) a scheme based on sequential decision-making and 2) a scheme based on a first-come-first-served crossing order heuristic The results show that the proposed MPC outperforms traffic lights and the alternative coordination controllers, in particular for high traffic intensities.

7.5 Paper E

R. Hult, M. Zanon, S. Gros and P. Falcone, "Energy-Optimal Coordination of Automated Vehicles at Intersections", presented at *European Control Conference*, 2018

In this paper, we examine the use of non-linear motion models and indefinite, economic, objective functions for MPC-based intersection coordination. In particular, we employ longitudinal dynamics of an electric vehicle, including air-drag and non-linear motor constraints, and consider an objective function that directly combines energy consumption with travel-time delay. We discuss the application of an approximation method from the literature, with which a quadratic cost can be tuned so that the resulting control-law is first-order equivalent to that obtained with the economic objective. We demonstrate the performance of both the economic and two varieties of the approximate economic MPC, and compare the results to that obtained with a standard MPC formulation that use a quadratic cost with diagonal weighting matrices. The results indicate that the approximate economic MPCs performs remarkably well on the intersection problem, and that almost economic performance can be obtained at the computational expense of a tracking MPC.

7.6 Paper F

R. Hult, M. Zanon, S. Gros and P. Falcone, "An Interior Point Algorithm for Optimal Coordination of Automated Vehicles at Intersections", to be submitted.

In this paper we discuss a primal-dual interior point algorithm for coordination problems with a fixed crossing order. The primal-dual search direction is obtained via distributed solution of the KKT system, where most computations are performed individually for each vehicle, a lesser part using information from all vehicles on each lane and a minor part using information from all vehicles. The step-size is selected using a distributed procedure with the same computational structure. This algorithm takes the same steps as a centralized solution, has the ability to treat rear-end collision avoidance constraints and avoids the non-smooth optimization of the primal-decomposition-based algorithm of Paper B. In a practical setting, most computations are performed on-board the vehicles, and the algorithm relies heavily on vehicle-to-vehicle communication. The communication demand is analyzed and an approximate reformulation is proposed to reduce the amount of information communicated. The approximation uses a lower-dimensional representation of the rear-end collision avoidance constraints and it is demonstrated that while the communication demand is reduced significantly, only minor sub-optimality is incurred.

Chapter 8

Conclusion

This thesis has contributed to the solution of the coordination problem for connected and automated vehicles at intersections. We formulated the coordination problem using tools from optimal control and optimization. Since the optimization formulation of the coordination problem is of large size, and the application is physically distributed by nature, we developed distributed solution algorithms. To handle the combinatorial nature of the problem, we derived optimization-based heuristics, applicable to realistic problem sizes. We applied the optimization algorithms to closed-loop control through MPC, examined their viability through experiments and assessed their impact on the traffic system through simulation. A number of questions remain open, which are briefly summarized below.

Robustness to perception uncertainties: The most critical measurement for the considered coordination problem is the geodesic distance to the intersection. This is typically constructed by combining maps with an absolute positioning system (e.g. GNSS). Unfortunately, such positioning systems are notoriously sensitive to disturbances (e.g. interference or lack of satellite coverage), and expensive when high accuracy and update frequency is required. Coordination algorithms that are robust to positioning uncertainties must therefore be developed to promote real-world application. While we proposed a simple constraint-tightening procedure in Paper A, more elaborate schemes should be investigated.

Coordination in mixed traffic: The work in this thesis is based on the assumption that all vehicles are cooperative, which restricts its application to a distant future. The OC formulation should therefore be extended to account for automated non-cooperative vehicles. One possibility is to include such vehicles in the OC formulation, but use driver models to generate the control commands. Moreover, vulnerable road users must also be included if

the algorithms are applied in urban environments. While arguably difficult to handle, pedestrian models have been proposed to be used as prediction tools [93]. The possibility to use such predictions in the OC formulation should be considered.

Communication-aware algorithms: The algorithms proposed in this thesis rely on wireless communications. In reality, the capacity of the communication channel is limited and will restrict the amount of data sent, the number of simultaneous transmissions and the frequency of transmissions. On top of this, packets may be lost, thus inducing delays or intermittency. Due to well-known effects of such communication impairments on the behavior of a closed-loop system, it is important to extend the proposed algorithms to operate safely with partial or delayed information. One possibility is to schedule the communication to account for the vehicle limitations, along the lines of [94, 95]. Another possibility which we explored in [96], is to adopt event-triggered coordination, where only the vehicles that are in risk of collision are considered.

Multi-intersection coordination: Several intersections are often positioned in connection to each other, and the coordination strategy used in one intersection could have detrimental effects on the performance adjacent ones. The extension of the OC-formulation to a network of intersections should therefore be considered. While it is essentially impossible to solve such problems to optimality, reasonably good heuristics could possibly be found. A possible strategy is to resolve the "combinatorial aspects" using heuristics, and thereafter solve a continuous OCP (similar to the two stage-procedure proposed in this thesis). The continuous OCP will be highly structured, and solution schemes similar to those discussed in Papers B and Fcould be employed.

Human Factors During the experiments reported in Papers A and B, it became apparent that automated intersection coordination with small margins can induce significant passenger distress. It therefore needs to be determined how to derive and include margins such that psychological well-being of the passengers is ensured. Moreover, while we considered a comfort promoting formulation in [83] that penalized large accelerations, further research is required to validate its efficiency and possibly derive extensions.

Investigation of alternative formulations: The coordination problem involves the optimization of the trajectories of dynamical systems. It *is*

therefore arguably an OCP, but it might not be most efficiently treated as such. A thorough investigation is necessary to determine the benefits and drawbacks of the different methodologies that have been suggested for the problem.

Moreover, this thesis has exclusively considered temporal formulation of the OC/MPC problem. An alternative is to use spatial formulations (as in [41, 42, 67]), where the dynamics are defined in terms of the position on the path rather than time. Since the two formulations have different benefits and drawbacks, a proper investigation should be conducted to determine which is most suitable for the coordination problem.

8.1 Outlook and Reflections

While the literature on the intersection problem is fairly recent, the interest from the research community seems to be increasing. In 2018 alone, more than 30 papers were published on the topic, and the problem is becoming a popular object of study within connected driving. The growing literature clearly demonstrates that it is not that hard to devise coordination schemes which outperform traffic lights or stop-signs, and that tools from a number of fields can be used.

However, demonstrating that the current traffic-system is inefficient has little practical relevance when full penetration of automated vehicles is assumed. Even though we seem close to the break-through of automated driving, it will likely take decades before all vehicles will be fully automated. Senior officials in the industry have even questioned whether this ever will be the case, given the difficulty they have had dealing with mildly adverse conditions such as snow or rain. It is therefore reasonable to shift focus towards algorithms that can operate under partial technology penetration. Unfortunately, some results indicate that the performance of automated coordination degrades rapidly when the fraction of non-cooperative vehicles increases [29]. These findings are intuitive, as a safe coordination algorithm for partial penetration scenarios must be conservative to robustly account for the uncertainty in the behavior of non-cooperative actors.

Even if cooperative AD becomes ubiquitous, pedestrians and other vulnerable road users remain a challenge. Since safety mandates a precautionary approach, the coordination algorithms need to hedge against all possible choices of the actors around, e.g., a pedestrian crossing. As a consequence, it could prove very difficult to deploy automated schemes in dense urban areas. It therefore seems reasonable that intersection coordination algorithms primarily will be applicable to pedestrian-free zones.

On top of this, certification of coordination algorithms is a daunting

task. Proving the safety of a coordination algorithm under the wide range of adverse conditions that the real world provides is significantly more difficult than making such complex systems work in idealized simulations or controlled experiments.

Finally, the quest towards decentralized or physically distributed strategies for computation and control could be somewhat misguided. Considering the current development of 5G, it is not unlikely that secure, high-bandwidth, low-latency communication could be the norm in a couple of decades. Similarly, given the development of "cloud"-based computation, it is not unreasonable to assume that an infrastructure of capable remote servers would be present. The easiest to certify, simplest and safest approach could therefore be to centralize more or less everything, and essentially control all vehicles remotely. All things considered, the traffic system envisioned at the *Futurama* fair of 1939 might just be what we end up with.

References

[1] National Highway Traffic Safety Administration (NHTSA), "National Motor Vehicle Crash Causation Survey," Sep. 2008.

[2] L. M. Clements and K. M. Kockelman, "Economic effects of automated vehicles," *Transportation Research Record*, vol. 2606, no. 1, pp. 106–114, 2017.

[3] Z. Wadud, D. MacKenzie, and P. Leiby, "Help or hindrance? the travel, energy and carbon impacts of highly automated vehicles," *Transportation Research Part A: Policy and Practice*, vol. 86, pp. 1 – 18, 2016.

[4] F. Duarte and C. Ratti, "The impact of autonomous vehicles on cities: A review," *Journal of Urban Technology*, vol. 25, pp. 1–16, 07 2018.

[5] R. Ferlis, "The dream of an automated highway," *Federal Highway Administration (FHA) Public Roads Magazine*, vol. 71, no. 1, 2007.

[6] K. Cardew, "The automatic steering of vehicles: an experimental system fitted to a DS19 Citroen car," *Road Research Laboratory Technical Report*, 1970.

[7] S. Tsugawa, T. Yatabe, T. Hirose, and S. Matsumoto, "An automobile with artificial intelligence," in *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, ser. IJCAI'79, 1979, pp. 893–895.

[8] E. D. Dickmanns *et al.*, "The seeing passenger car 'vamors-p'," in *Proceedings of the Intelligent Vehicles '94 Symposium*, Oct 1994, pp. 68–73.

[9] M. Maurer, R. Behringer, S. Furst, F. Thomanek, and E. D. Dickmanns, "A compact vision system for road vehicle guidance," in *Proceedings of 13th International Conference on Pattern Recognition*, vol. 3, Aug 1996, pp. 313–317 vol.3.

[10] A. Broggi, M. Bertozzi, A. Fascioli, C. Guarino, L. Bianco, and A. Piazzi, "The argo autonomous vehicles vision and control systems," *International Journal of Intelligent Control and Systems*, pp. 409–441, 1999.

[11] M. Bertozzi *et al.*, "Viac: An out of ordinary experiment," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, June 2011, pp. 175–180.

[12] M. E. Campbell, M. Egerstedt, J. P. How, and R. M. Murray, "Autonomous driving in urban environments: approaches, lessons and challenges." *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, vol. 368 1928, pp. 4649–72, 2010.

[13] Waymo, (https://waymo.com/ontheroad/), accessed Feb. 2019.

[14] T. Walker, "The self-driving car timeline - predictions from the top 11 global automakers," Emerj Online: (https://emerj.com/ai-adoption-timelines/self-driving-car-timeline-themselves-top-11-automakers/), feb 2019.

[15] S. Kim *et al.*, "The impact of cooperative perception on decision making and planning of autonomous vehicles," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 3, pp. 39–50, Fall 2015.

[16] S. E. Shladover *et al.*, "Automated vehicle control developments in the path program," *IEEE Transactions on Vehicular Technology*, vol. 40, no. 1, pp. 114–130, Feb 1991.

[17] T. Robinson and E. Coelingh, "Operating platoons on public motorway -: An introduction to the sartre platooning programme," in *Proceedings of the 17th ITS World Congress*, 03 2010.

[18] E. van Nunen, M. Kwakkernaat, J. Ploeg, and B. D. Netten, "Cooperative competition for future mobility," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1018–1025, Sep. 2012.

[19] C. Englund *et al.*, "The grand cooperative driving challenge 2016: boosting the introduction of cooperative automated vehicles," *IEEE Wireless Communications*, vol. 23, no. 4, pp. 146–152, August 2016.

[20] European Commission, "Traffic Safety Basic Facts - Junctions," Sep. 2018.

[21] National Highway Traffic Safety Administration (NHTSA), "Fatality Analysis Reporting System (FARS)," Online: (http://www.nthsa.gov/FARS), Sep. 2018.

[22] M. Li *et al.*, "Traffic energy and emission reductions at signalized intersections: a study of the benefits of advanced driver information," *International Journal of Intelligent Transportation Systems Research*, vol. 7, no. 1, pp. 49–58, 2009.

[23] US Treasury, "A new economic analysis of infrastructure investment," Online: (https://www.treasury.gov/press-center/news/Pages/03232012-infrastructure.aspx), Mar. 2012.

[24] R. Hult *et al.*, "Coordination of cooperative autonomous vehicles: Toward safer and more efficient road transportation," *IEEE Signal Processing Magazine*, vol. 33, no. 6, pp. 74–84, Nov 2016.

[25] A. Colombo and D. D. Vecchio, "Least restrictive supervisors for intersection collision avoidance: A scheduling approach," *IEEE Transactions on Automatic Control*, vol. 60, no. 6, pp. 1515–1527, June 2015.

[26] H. Wymeersch, G. R. de Campos, P. Falcone, L. Svensson, and E. G. Ström, "Challenges for cooperative its: Improving road safety through the integration of wireless communications, control, and positioning," in *2015 International Conference on Computing, Networking and Communications (ICNC)*, Feb 2015, pp. 573–578.

[27] R. Naumann, R. Rasche, and J. Tacken, "Managing autonomous vehicles at intersections," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 3, pp. 82–86, May 1998.

[28] 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*, July 2004, pp. 530–537.

[29] ——, "A Multiagent Approach to Autonomous Intersection Management," *Journal of Artificial Intelligence Research*, vol. 31, no. 1, pp. 591–656, Mar. 2008.

[30] L. Chen and C. Englund, "Cooperative intersection management : A survey," *IEEE transactions on intelligent transportation systems*, vol. 17, no. 2, pp. 570–586, 2016.

[31] J. Rios-Torres and A. A. Malikopoulos, "A survey on the coordination of connected and automated vehicles at intersections and merging at highway on-ramps," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, pp. 1066–1077, 2017.

[32] Q. Jin *et al.*, "Platoon-based multi-agent intersection management for connected vehicle," in *16th International IEEE Conference on Intelligent Transportation Systems*, Oct 2013, pp. 1462–1467.

[33] H. Kowshik, D. Caveney, and P. R. Kumar, "Provable systemwide safety in intelligent intersections," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 3, pp. 804–818, March 2011.

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

[35] E. M "Intersection control for automated vehicles with milp," *IFAC-PapersOnLine*, vol. 49, no. 3, pp. 37 – 42, 2016.

[36] D. Miculescu and S. Karaman, "Polling-systems-based control of high-performance provably-safe autonomous intersections," in *53rd IEEE Conference on Decision and Control*, Dec 2014, pp. 1417–1423.

[37] F. Yan, M. Dridi, and A. E. Moudni, "A scheduling approach for autonomous vehicle sequencing problem at multi-intersections," *International Journal of Operations Research*, vol. 8, no. 1, pp. 57–68, 2011.

[38] K. Zhang, D. Zhang, A. de La Fortelle, X. Wu, and J. Grégoire, "State-driven priority scheduling mechanisms for driverless vehicles approaching intersections," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2487–2500, Oct 2015.

[39] J. Grégoire and E. Frazzoli, "Hybrid centralized/distributed autonomous intersection control: Using a job scheduler as a planner and inheriting its efficiency guarantees," in *IEEE 55th Conference on Decision and Control*, Dec 2016, pp. 2549–2554.

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

[41] N. Murgovski, G. R. de Campos, and J. Sjöberg, "Convex modeling of conflict resolution at traffic intersections," in *IEEE Conference on Decision and Control*, Dec 2015, pp. 4708–4713.

[42] L. Riegger, M. Carlander, N. Lidander, N. Murgovski, and J. Sjöberg, "Centralized mpc for autonomous intersection crossing," in *IEEE 19th International Conference on Intelligent Transportation Systems*, Nov 2016, pp. 1372–1377.

[43] P. Tallapragada and J. Cortés, "Coordinated intersection traffic management," *IFAC-PapersOnLine*, vol. 48, no. 22, pp. 233 – 239, 2015.

[44] R. Hult, M. Zanon, S. Gros, and P. Falcone, "Primal decomposition of the optimal coordination of vehicles at traffic intersections," in *IEEE 55th Conference on Decision and Control*, Dec 2016, pp. 2567–2573.

[45] M. Zanon, S. Gros, H. Wymeersch, and P. Falcone, "An asynchronous algorithm for optimal vehicle coordination at traffic intersections," 2017, 20th IFAC World Congress.

[46] Y. Jiang, M. Zanon, R. Hult, and B. Houska, "Distributed algorithm for optimal vehicle coordination at traffic intersections," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 11 577 – 11 582, 2017.

[47] J. Shi *et al.*, "Distributed control algorithm for vehicle coordination at traffic intersections," in *European Control Conference*, June 2018, pp. 1166–1171.

[48] M. Kneissl, A. Molin, H. Esen, and S. Hirche, "A feasible mpc-based negotiation algorithm for automated intersection crossing *," in *European Control Conference*, 06 2018, pp. 1282–1288.

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

[50] B. Li, Y. Zhang, Y. Zhang, J. Ning, and Y. Ge, "Near-optimal online motion planning of connected and automated vehicles at a signal-free and lane-free intersection," in *IEEE Intelligent Vehicle Symposium*, 06 2018, pp. 1432–1437.

[51] C. Bali and A. Richards, "Merging vehicles at junctions using mixed-integer model predictive control," in *European Control Conference*, 06 2018, pp. 1740–1745.

[52] R. Hult, G. R. Campos, P. Falcone, and H. Wymeersch, "An approximate solution to the optimal coordination problem for autonomous vehicles at intersections," in *2015 American Control Conference (ACC)*, July 2015, pp. 763–768.

[53] A. I. M. Medina, N. van de Wouw, and H. Nijmeijer, "Cooperative intersection control based on virtual platooning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, pp. 1727–1740, 2018.

[54] N. Neuendorf and T. Bruns, "The vehicle platoon controller in the decentralised, autonomous intersection management of vehicles," in *Proceedings of the IEEE International Conference on Mechatronics, 2004. ICM '04.*, June 2004, pp. 375–380.

[55] M. Di Vaio, P. Falcone, R. Hult, A. Petrillo, and S. S. A. Salvi, "Design and experimental validation of a distributed interaction protocol for connected autonomous vehicles at road intersection," *submitted to IEEE Transactions on Vehicular Technology*, 2018.

[56] M. V. Middlesworth, K. M. Dresner, and P. Stone, "Replacing the stop sign: unmanaged intersection control for autonomous vehicles," in *AAMAS*, 2008.

[57] V. Milanes, J. Perez, E. Onieva, and C. Gonzalez, "Controller for urban intersections based on wireless communications and fuzzy logic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 1, pp. 243–248, March 2010.

[58] R. Azimi, G. Bhatia, R. Rajkumar, and P. Mudalige, "Intersection management using vehicular networks," in *SAE Technical Paper*. SAE International, 04 2012. [Online]. Available: https://doi.org/10.4271/2012-01-0292

[59] 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*, Oct 2013, pp. 1456–1461.

[60] G. R. de Campos, P. Falcone, H. Wymeersch, R. Hult, and J. Sjöberg, "Cooperative receding horizon conflict resolution at traffic intersections," in *53rd IEEE Conference on Decision and Control*, Dec 2014, pp. 2932–2937.

[61] X. Qian, J. Grégoire, A. de La Fortelle, and F. Moutarde, "Decentralized model predictive control for smooth coordination of automated vehicles at intersection," in *European Control Conference*, July 2015, pp. 3452–3458.

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

[63] F. Molinari and J. Raisch, "Automation of road intersections using consensus-based auction algorithms," in *2018 Annual American Control Conference (ACC)*, June 2018, pp. 5994–6001.

[64] L. Makarem and D. Gillet, "Model predictive coordination of autonomous vehicles crossing intersections," in *16th International IEEE Conference on Intelligent Transportation Systems*, Oct 2013, pp. 1799–1804.

[65] A. Katriniok, P. Kleibaum, and M. Josevski, "Distributed model predictive control for intersection automation using a parallelized optimization approach," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 5940 – 5946, 2017.

[66] T. Sprodowski and J. Pannek, "Stability of distributed MPC in an intersection scenario," *Journal of Physics: Conference Series*, vol. 659, p. 012049, nov 2015.

[67] A. Britzelmeier and M. Gerdts, "Non-linear model predictive control of connected, automatic cars in a road network using optimal control methods," *IFAC-PapersOnLine*, vol. 51, no. 2, pp. 168 – 173, 2018.

[68] 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, 2018.

[69] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear programming - theory and algorithms (2. ed.)*. Wiley, 1993.

[70] M. Patriksson *et al.*, *An Introduction to Continuous Optimization - Foundations and Fundamental Algorithms*. Studentliteratur, 11 2016.

[71] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

[72] A. V. Fiacco and Y. Ishizuka, "Sensitivity and stability analysis for nonlinear programming," *Annals of Operations Research*, vol. 27, no. 1, pp. 215–235, Dec 1990.

[73] J. V. Frasch, S. Sager, and M. Diehl, "A parallel quadratic programming method for dynamic optimization problems," *Mathematical Programming Computation*, vol. 7, no. 3, pp. 289–329, Sep 2015.

[74] D. S. Naidu and S. Naidu, *Optimal Control Systems*, R. C. Dorf, Ed. Boca Raton, FL, USA: CRC Press, Inc., 2002.

[75] L. Biegler, *Nonlinear Programming*. Society for Industrial and Applied Mathematics, 2010. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9780898719383

[76] J. Butcher, *Numerical Methods for Ordinary Differential Equations*. Wiley, 2005.

[77] H. Bock and K. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems*," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603 – 1608, 1984, 9th IFAC World Congress.

[78] J. Rawlings and D. Mayne, *Model Predictive Control: Theory and Design*. Nob Hill, 2009.

[79] M. Diehl, R. Amrit, and J. B. Rawlings, "A lyapunov function for economic optimizing model predictive control," *IEEE Transactions on Automatic Control*, vol. 56, no. 3, pp. 703–707, March 2011.

[80] in *Tire and Vehicle Dynamics*, third edition ed., H. B. Pacejka, Ed. Oxford: Butterworth-Heinemann, 2012.

[81] L. Guzzella and A. Sciarretta, *Vehicle Propulsion Systems: Introduction to Modeling and Optimization*, January 2007.

[82] International Energy Agency (IEA), "Global EV Outlook," 2018.

[83] R. Hult, M. Zanon, S. Gros, and P. Falcone, "Optimal coordination of automated vehicles at intersections with turns," in *European Control Conference*, 2019.

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

[85] A. Richards and J. How, "Mixed-integer programming for control," in *Proceedings of the 2005, American Control Conference, 2005.*, June 2005, pp. 2676–2683 vol. 4.

[86] R. Hult, M. Zanon, S. Gros, and P. Falcone, "Optimal coordination of three cars approaching an intersection," Online: `https://youtu.be/nYSXvnaNRK4`, 2017.

[87] G. Frison, H. H. B. Sørensen, B. Dammann, and J. B. Jørgensen, "High-performance small-scale solvers for linear model predictive control," in *European Control Conference*, June 2014, pp. 128–133.

[88] A. Domahidi and J. Jerez, "FORCES Professional," embotech GmbH (http://embotech.com/FORCES-Pro), Jul. 2014.

[89] Novatel, *An Introduction to GNSS*. Novatel, Inc., 2015. [Online]. Available: https://www.novatel.com/assets/Documents/Books/Intro-to-GNSS.pdf

[90] R. Hult *et al.*, "Design and experimental validation of a cooperative driving control architecture for the grand cooperative driving challenge 2016," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1290–1301, April 2018.

[91] R. Hult, M. Zanon, W. H. Gros, S., and P. Falcone, "Animated illustration of the MIQP/FO controller," Online `https://youtu.be/hUWQoaiqdAY`, 2019, accessed: 2019-02-26.

[92] M. Zanon, S. Gros, and M. Diehl, "A tracking mpc formulation that is locally equivalent to economic mpc," *Journal of Process Control*, vol. 45, pp. 30 – 42, 2016.

[93] I. Batkovic, M. Zanon, N. Lubbe, and P. Falcone, "A computationally efficient model for pedestrian motion prediction," in *European Control Conference*, June 2018, pp. 374–379.

[94] A. Colombo, M. Bahraini, and P. Falcone, "Measurement scheduling for control invariance in networked control systems," in *IEEE Conference on Decision and Control*, Dec 2018, pp. 3361–3366.

[95] M. Bahraini, M. Zanon, A. Colombo, and P. Falcone, "Receding-horizon robust online communication scheduling for constrained networked control systems," in *European Control Conference*, 2019.

[96] E. Steinmetz *et al.*, "Collision-aware communication for intersection management of automated vehicles," *IEEE Access*, vol. 6, pp. 77 359–77 371, 2018.