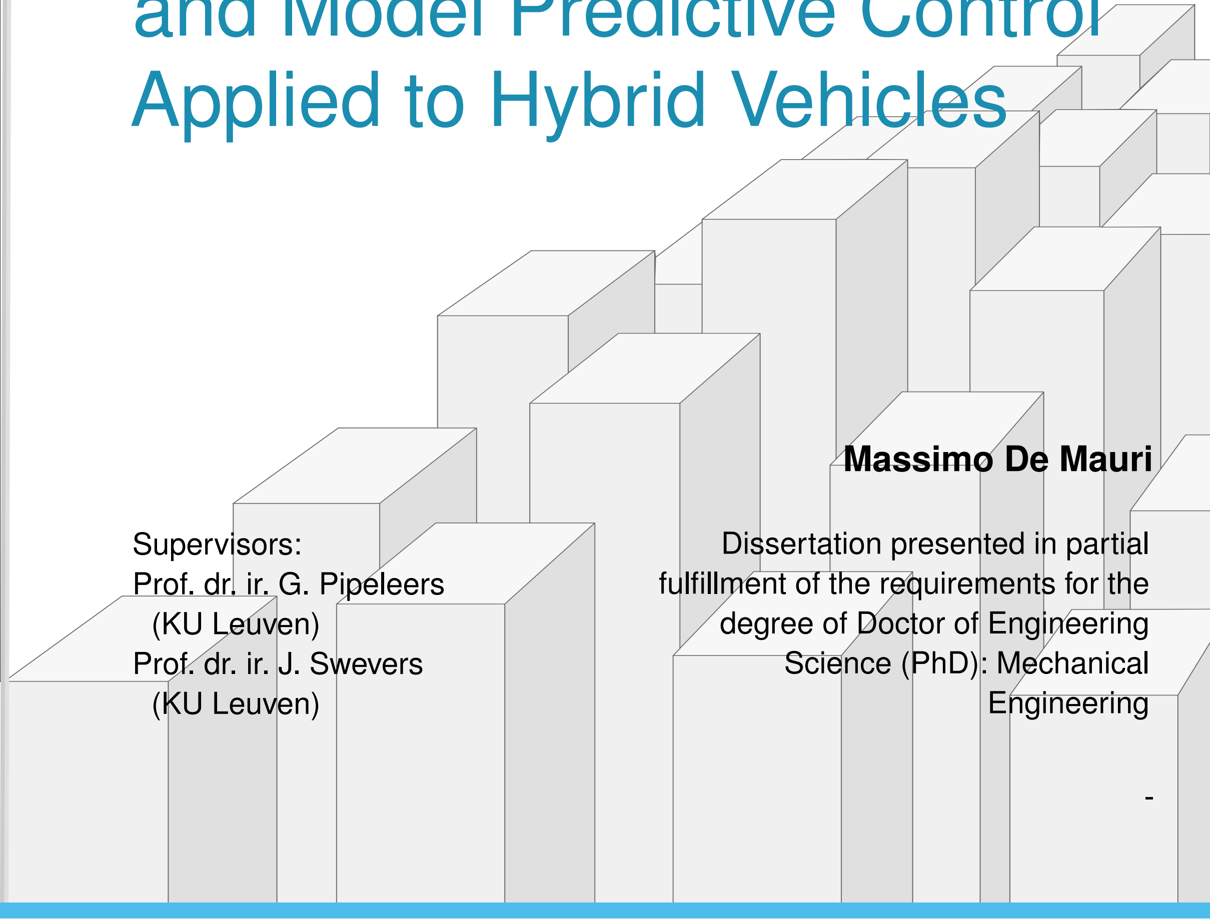
KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Mixed-Integer Optimization and Model Predictive Control Applied to Hybrid Vehicles



Massimo De Mauri

Supervisors:
Prof. dr. ir. G. Pipeleers
(KU Leuven)
Prof. dr. ir. J. Swevers
(KU Leuven)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Mechanical Engineering

-

Mixed-Integer Optimization and Model Predictive Control Applied to Hybrid Vehicles

Massimo DE MAURI

Examination committee:
Prof. dr. ir. D. Vandermeulen, chair
(KU Leuven)
Prof. dr. ir. G. Pipeleers, supervisor
(KU Leuven)
Prof. dr. ir. J. Swevers, supervisor
(KU Leuven)
Prof. dr. ir. P. Patrinos (KU Leuven)
Dr. J. Gillis (KU Leuven)
Prof. dr. M. Diehl
(Albert-Ludwigs-Universität Freiburg)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Mechanical Engineering

-

© 2020 KU Leuven – Faculty of Engineering Science
Uitgegeven in eigen beheer, Massimo De Mauri, Celestijnenlaan 300, B-3001 Leuven (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm, electronic or any other means without written permission from the publisher.

Acknowledgments

First, I would like to thank my supervisors, Prof. Goele Pipeleers and Prof. Jan Swevers, for supporting my personal growth and for their encouragements in most difficult days of my PhD journey. Another special thanks goes to Dr. Joris Gillis whose outstanding academic work and patient technical support made the present dissertation possible.

Secondly, I would like to thank Prof. Diehl, Dr. Gillis and Prof. Patrinos for their precious feedbacks and suggestions for the betterment of the present thesis. It was a pleasure and a honor to have you in my PhD examination committee. Prof. Vandermeulen thank you for accepting the chairing role in my defense ceremony.

Furthermore, I would like to convey my gratitude to all the members of the MECO research team, past and present, for giving life to a lively and stimulating work environment. In particular, I would like to thank Dr. Gijs Hilhorst for walking me through my first steps as PhD student and teaching assistant, Dr. Armin Steinhauser for our deep and lengthy conversations, and (Dr. soon enough) Taranjitsing Sing for our academic collaboration and for letting me win a ping-pong match from time to time.

Next, I thank my closest friends in Belgium, especially Arnaud, Caterina, Catherine, Gaia, Lorenzo, Marco, Naser and Nereo, for countering the effects of my D-vitamin deficiency. To the fellow members of PaP Bruxelles, Andrea, Caterina, Fabio, Francesca, Germano, Giorgio, Lorenzo, Piera, Pietro, Umberto, Valeria, thanks for believing and striving.

Un grazie ai miei amici di Roma (voi siete troppi per fare una lista di nomi) per essere parte fondamentale della mia storia e farmi sentire a casa ogni volta che ho la possibilità di tornare a Roma, nonostante gli anni passati all'estero.

Alla mia famiglia più stretta, Mamma, Papà, Ilaria, Enrico, Sandra e Steffen, grazie per essere il pilastro su cui la mia, talvolta caotica, vita resta in equilibrio.

Siete la migliore famiglia che si possa avere (senza dubbio la migliore che io abbia avuto). Grazie agli zii, le zie, cugini e nipoti per aver creato intorno a me una piccola società basata sul rispetto e l'amore reciproco.

Grazie ad Agatha che è apparsa nella mia vita a sorpresa come un acca che non si sa piazzare ed è presto divenuta la parte migliore della mia quotidianità. Compagna di viaggio, migliore amica e mia amata.

Dedicato al piccolo Alessandro, qualuque sarà la strada che sceglierai mi avrai al tuo fianco, in bocca al lupo.

Brussels, 12th of November 2020,
Massimo De Mauri.

Abstract

In this era of ubiquitous automation, all fields of optimal control are subject to widespread scientific and industrial interest. This includes a field like Mixed-Integer Optimal Control (MI-OCP) which, despite its ability to represent many interesting problems, has struggled to find application due to its high numerical complexity. Nevertheless, thanks to the general increase of available computational power and the development of efficient solution algorithms, MI-OCP has become a viable technique in an ever-increasing number of practical scenarios. The present PhD dissertation contributes to this field by presenting three distinct algorithms, each addressing a specific issue related to MI-OCP. Additionally, special attention is paid to hybrid electric vehicle control: an application that can largely benefit from robust and efficient MI-OCP schemes.

The most performant existing solvers for Mixed-Integer Convex Optimization implement some sort of outer approximation technique. Consequently, the first contribution presented is named Proximal Outer Approximation (POA). POA is an algorithm designed to reduce the computational cost of solving highly non-linear problem instances. POA has the same structure as the original Outer Approximation algorithm but, thanks to additional adaptive proximity-promoting cost terms, it is able to provide noteworthy computational savings, especially on the most difficult instances.

Designing a real-time Mixed-Integer Model Predictive Control (MI-MPC) scheme is often challenging from the computational complexity perspective. However, any MPC procedure consists mainly in solving a sequence of problems where each problem is quite similar to the next. The second algorithm presented in this dissertation, namely the Mixed-Integer Real Time Optimal Control (MIRT-OC) technique, exploits such similarity to reduce the complexity of MI-MPC. In particular, MIRT-OC uses part of the information collected during one MPC iteration to speed up the next. The collected numerical evidence clearly shows the greater effectiveness of MIRT-OC with respect to the commonly used approach.

The complexity of Mixed-Integer Non-Linear Optimal Control may greatly increase if systems described by non-convex functions are considered. The last contribution presented consists of an algorithm named Disjunctive Outer Approximation for Optimal Control (DOA-OC). DOA-OC is applicable to any mixed-integer optimal control problem where constraints and objective terms, while being non-convex, depend in a convex manner on the continuous variables of the problem. The proposed algorithm extends the OA framework to the aforementioned problem class and, by the same token, represents a much more convenient alternative to the extremely complex non-convex optimization schemes.

Additionally, the conclusion of this dissertation presents a number of suggestions on possible interesting future developments for the presented work.

Beknopte samenvatting

In dit tijdperk van alomtegenwoordige automatiseringis er brede wetenschappelijke en industriële interesse voor optimale controle. Dit omvat ook het domein van "Mixed-Integer"Optimale Controle (MI-OCP) dat, ondanks de brede waaier van toepassingsmogelijkheden, moeite heeft om toepassing te vinden vanwege de hoge numerieke complexiteit. Niettemin is MI-OCP, dankzij de algemene toename van de beschikbare rekenkracht en de ontwikkeling van efficiënte oplossingsalgoritmen, een levensvatbare techniek geworden in een steeds toenemend aantal praktische scenario's. Dit doctoraatsproefschrift draagt bij aan dit vakgebied met drie verschillende algoritmen, die elk een specifiek probleem met betrekking tot MI-OCP aanpakken. Daarnaast wordt speciale aandacht besteed aan de besturing van hybride elektrische voertuigen: een toepassing die in belangrijke mate kan profiteren van robuuste en efficiënte MI-OCP-schema's.

De best presterende bestaande oplossingsmethoden voor Mixed-Integer Convexe Optimalisatie implementeren een soort externe benaderingstechniek. Bijgevolg wordt de eerste de eerste bijdrage in dit proefschrif Proximal Outer approximation (POA) genoemd. POA is een algoritme dat is ontworpen om de rekenkosten voor het oplossen van zeer niet-lineaire probleemgevallen te verlagen. POA heeft dezelfde structuur als het oorspronkelijke Outer approximation-algoritme, maar die dankzij aanvullende adaptieve kostfuncties die de nabijheid van een oplossing bevorderen, aanzienlijke besparingen in rekentijd kunnen realiseren, vooral indien men te maken heeft met complexe problemen.

Het ontwerpen van een real-time Mixed-Integer model-gebaseerd voorspellend regelschema (MI-MPC)-schema is vaak een uitdaging vanuit het perspectief van computationele complexiteit. Elke MPC-procedure bestaat echter hoofdzakelijk uit het oplossen van een reeks problemen waarbij elk probleem vrij gelijkaardig is aan het volgende. Het tweede algoritme dat in dit proefschrift wordt gepresenteerd, namelijk de Mixed-Integer Real Time Optimal Control (MIRT-OC) -techniek, maakt gebruik van deze gelijkenis om de complexiteit van

MI-MPC te verminderen. In het bijzonder gebruikt MIRT-OC een deel van de informatie die tijdens de ene MPC-iteratie wordt verzameld om de volgende te versnellen. Aan de hand van numerieke experimenten wordt de grotere effectiviteit van MIRT-OC ten opzichte van de meest gebruikte aanpak aangetoond.

De complexiteit van niet-lineaire optimale controle kan aanzienlijk toenemen voor systemen die worden beschreven door niet-convexe functies. De laatste bijdrage van dit proefschrift bestaat uit een algoritme genaamd Disjunctive Outer approximation for Optimal Control (DOA-OC). DOA-OC is toepasbaar op elk mixed-integer optimaal controleprobleem waarbij beperkingen en doelfuncties, hoewel ze niet-convex zijn, op een convexe manier afhangen van de continue variabelen van het probleem. Het voorgestelde algoritme breidt het OA-raamwerk uit naar de bovengenoemde probleemklasse en vormt op dezelfde manier een veel handiger alternatief voor complexer niet-convexe optimalisatieschema's.

Tenslotte presenteert het besluit van dit proefschrift een aantal suggesties over mogelijke interessante toekomstige ontwikkelingen voor het gepresenteerde werk.

Abbreviations

B&B : Branch and Bound

DOA-OC : Disjunctive Outer Approximation for Optimal Control

EM : Electric Motor

GB : Gear Box

HEV : Hybrid Electric Vehicle

ICE : Internal Combustion Engine

LP-NLP B&B : Linear Programming/Non-Linear Programming Branch and Bound

MIOCP : Mixed-Integer Optimal Control Problem

MICP : Mixed-Integer Convex Problem

MILP : Mixed-Integer Linear Problem

MINCP : Mixed-Integer Non-Convex Problem

MINLP : Mixed-Integer Non-Linear Problem

MIP : Mixed-Integer Problem

MIQP : Mixed-Integer (convex) Quadratic Problem

MIRT-OC : Mixed-Integer Real-Time Optimal Control

MPC : Model Predictive Control

NLP : Non-Linear Problem

OA : Outer Approximation

OC : Optimal Control

OCP : Optimal Control Problem

PHEV : Parallel Hybrid Electric Vehicle

POA : Proximal Outer Approximation

Contents

Abstract iii

Beknopte samenvatting v

Abbreviations vii

Contents ix

List of Figures xiii

List of Tables xv

1 Introduction 1

1.1 Motivation 2

1.2 Challenges 3

1.3 Contributions 4

1.4 Overview 5

2 Preliminaries and Basic Methodologies 7

2.1 Optimal Control 7

2.2 Direct Methods For Optimal Control 9

2.2.1 Direct Multiple Shooting 10

2.3 Mixed-Integer Optimal Control 12
2.4 Mixed-Integer Model Predictive Control 15
  2.4.1 Mixed-Integer MPC in offline conditions 15
  2.4.2 Mixed-Integer MPC in online conditions 17
2.5 Mixed-Integer Optimization 19
  2.5.1 Nomenclature 20
  2.5.2 Feasibility and Optimality in Mixed-Integer Optimization 20
  2.5.3 Relaxations and Restrictions 21
2.6 Methods for Mixed-Integer Convex Optimization 22
  2.6.1 The Branch&Bound algorithm 23
  2.6.2 The Outer Approximation Algorithm 27
2.7 Mixed-Integer Non-Convex Optimization 31
2.8 Summary and Final Comments 31

3 Mixed-Integer Convex Optimization: Proximal-Outer Approximation 33
3.1 Motivation 34
3.2 Problem Definition 35
3.3 Preliminaries: The Non-Linear Feasibility Pump 35
3.4 Proximal Outer Approximation 37
  3.4.1 Structure of POA 37
  3.4.2 POA compared to OA 41
  3.4.3 Convergence and Exactness of POA 42
  3.4.4 Setting the parameters 45
3.5 Numerical Evaluation 47
  3.5.1 Analysis Via Shifted Geometric Mean 48
  3.5.2 Analysis Via Performance Profiles 50
3.6 Summary and Final Comments 51

3.7 Appendix: Table of Results 53

4 Mixed-Integer Convex Model Predictive Control: The MIRT-OC Algorithm 59

4.1 Problem Definition 60

4.2 LP/NLP Branch&Bound 62

4.3 The Algorithmic Structure of MIRT-OC 65

4.3.1 Linearization-Shift 66

4.3.2 Tree-Shift 68

4.4 Obtaining Lower-Bounds for The Shifted Nodes 69

4.4.1 Branch&Bound Sub-Problems 70

4.4.2 Computing a New Feasible Dual-Point 71

4.4.3 Computing a New Lower-Bound 73

4.5 Numerical Validation 74

4.5.1 The Case of Study 74

4.5.2 The resulting model 81

4.5.3 Numerical Results 82

4.6 Summary and Final Comments 85

4.7 Appendix A: Getting Rid Of Discrete States 86

4.8 Appendix B: Derivation of $\mathcal{O}_{\text{contraction}}$ 87

4.9 Appendix C: Extensions and Different Applications 88

4.10 Appendix D: Variants for the Shifting Procedure 90

5 Tackling Non-Convexity: Disjunctive Outer Approximation for Optimal Control 91

5.1 Problem Definition 92

5.1.1 From Mixed-Integer Semi-Convex to Disjunctive-Convex 94

5.2 Preliminaries 95

5.2.1 Big-M reformulation 95
5.2.2 Linear Bound-Propagation and Upper Bounds for Linear Expressions 96
5.3 Linear Relaxations for Disjunctive-Convex Constraints 97
5.3.1 Relaxing the Continuous Part of the Constraint 98
5.3.2 Relaxing the Discrete Part of the Constraint 98
5.4 Disjunctive Outer Approximation for Optimal Control 99
5.4.1 The Algorithmic Structure 100
5.4.2 Convergence Properties 103
5.5 Numerical Validation 103
5.5.1 The Case of Study 103
5.5.2 The Resulting Model 109
5.5.3 Numerical Results 110
5.6 Summary and Final Comments 115

6 Summary, Conclusions and Outlook 117

6.1 Summary 117
6.2 Conclusions 119
6.3 Outlook 120
6.3.1 Combining POA with MIRT-OC 120
6.3.2 Combining POA with DOA-OC 120
6.3.3 Combining DOA-OC with MIRT-OC 121

Bibliography 123

Curriculum Vitae 131

Pubilcations 133

List of Figures

2.1 Representation of a typical control solution found via DMS including pre-convergence discontinuous state trajectory. 11

2.2 Behaviour of B&B on a simple problem $\mathcal{P}$. 26

2.3 Schematic Representation of OA 30

3.1 Schematic Representation of Non-Linear Feasibility Pump . . . 37

3.2 Schematic Representation of POA 40

3.3 Performance Profiles plot obtained considering all instances. . . . 51

3.4 Performance Profiles plot obtained considering less trivial instances. 51

4.1 Parallel Hybrid Electric Vehicle Topology 74

4.2 the Federal Test Procedure 75 75

4.3 Engine:Fitted Model VS Original Data (Efficiency) 77

4.4 Electric Motor: Fitted Model VS Original Data (Efficiency) . . 79

4.5 Sample solutions with 15 seconds of prediction 84

5.1 Schematic representation of Disjunctive OA for Optimal Control 102

5.2 Parallel Hybrid Electric Vehicle Topology (Rep.) 103

5.3 Engine: Fitted Model VS Original Data (Instant. Fuel Consumption) 105

5.4 Electric Motor: Fitted Model VS Original Data (Efficiency) . . 107

5.5 Result for 6[s] of Prediction Horizon 111
5.6 Time-per-Iteration for Disjunctive OA with Linearizations Shift at the Variation of the Prediction Horizon 112
5.7 Disjunctive OA: Obtained Discrete Control Actions at the . . 113
5.8 Comparison Disjunctive OA versus SCIP 114

List of Tables

1.1 Presented Algorithms and Fields of Application 5

3.1 Absolute performances of the analyzed algorithms (full dataset). 48
3.2 Performance comparison versus classical OA (full dataset). . . . 49
3.3 Performance comparison versus classical OA (time-demanding instances). 49
3.4 Performance comparison versus classical OA (iteration-intensive instances). 49

4.1 Comparison between MIRT-OC and the classical MI-MPC approach for different scenarios/prediction lengths 83

5.1 Disjunctive OA: Overall Objective at the Variation of the Prediction Horizon 113

Chapter 1

Introduction

The ability of predicting and manipulating the behaviour of physical systems in a rigorous manner paved the way that led from the simply mechanical machines of the first and second industrial revolution to the contemporary autonomous mechatronic systems. Such rigorous approach to automation takes the name of "Control Theory". The twentieth century has witnessed a great proliferation of control theory applications as automation became almost ubiquitous. Starting from the 1980s, when desktop-sized digital computers started to become of common use, a new branch of control theory arose to popularity: Optimal Control. The aim of optimal control is to manipulate the behaviour of a system in order to maximize some measure of desirability (e.g. the amount of energy saved or minimal operation time). To date, Optimal Control finds good success in many control and planning applications.

The subject matter of the present PhD dissertation is Mixed-Integer Optimal Control: a branch of Optimal Control focused on the solution of control problems where some of the variables are constrained to take value within a finite number of possible assignments while the others are allowed to span continuous sets (commonly, intervals of the real numbers). In particular, this thesis will address the issue of developing new efficient strategies capable of reducing the effort required for the solution of specific classes of mixed-integer non-linear optimal control problems.

1.1 Motivation

Mixed-Integer Optimal Control is an important tool for taking the best possible decisions over time in all those applications where the outcome depends on some continuous signals as well as on the selection of one out of a finite number of possible actions. Common examples of optimal control applications are the autonomous selection of gears for a vehicle or the dynamic assignment of finite resources in a production line.

In the past, the high complexity of Mixed-Integer Optimal Control relegated its usage to systems with very slow dynamics and to digital simulations. In spite of that, over time, thanks to the great increment in available computational power, Mixed-Integer Optimal Control slowly started to be regarded as a viable approach for controlling of faster systems in real time.

However, more powerful computers alone are not sufficient to solve the challenges posed by Mixed-Integer Optimal Control. As a matter of fact, although digital computers have nowadays almost exhausted their margins of improvement, the efficient solution of medium-sized mixed-integer non-linear optimal problems is still quite difficult. As a consequence, it is essential to improve the performances of the algorithms involved in mixed-integer optimal control in order to further enlarge the set of its possible applications.

Hybrid Electric Vehicles as Case Study

In this thesis the control for Hybrid Electric Vehicles will be used as main source of numerical experiments to be used to asses and demonstrate the performances of the developed algorithm. Nevertheless, none of the methods presented in this dissertation is limited to this specific application.

The selection of hybrid electric vehicles control as main case study is not coincidental. In the recent years, many Hybrid Electric Vehicles (HEV) have entered the automotive market with a fairly good success. This is due to the fact that HEVs have the potential of achieving a much higher average fuel-efficiency and environmental sustainability compared to traditional vehicles. The main advantages of HEVs over traditional vehicles lies in their capability of recovering energy during deceleration (regenerative braking) together with some additional degrees of freedom in the selection of the operating point of the combustion engine. As a consequence of the additional freedom, automatic control plays a key role in HEVs design as such freedom must be carefully exploited in order maximize fuel economy and minimize pollutant emissions.

From a more technical point of view, the optimal control of hybrid cars makes for an interesting case study. In fact, in such application, difficult mixed-integer non-linear models have to be considered under the requirement of real-time-compatible solution times, and therefore, the efficiency of the employed solution methods is particularly relevant.

1.2 Challenges

Mixed-Integer Optimization is an NP-hard problem. This means that the time required for the solution of a mixed-integer optimization problem is an exponential function of the problem size. As a consequence, in Mixed-Integer Optimal Control the length of the time interval that can be considered might be severely limited. However, not all mixed-integer control problems are equally demanding: if, on the one hand, large linear and quadratic mixed-integer optimal control problems can nowadays be solved with relative ease, on the other hand, moderately sized mixed-integer non-linear problems still require long times to be solved due to their higher computational complexity. Consequently, when considering mixed-integer non-linear optimal control problems, it is in general quite challenging to obtain real-time compatible performances without taking shortcuts at the expense of the quality of the resulting solutions.

The difficulty of Mixed-Integer Non-Linear Optimal Control arises from the fact that, in this field, the high computational cost of Mixed-Integer Optimization meets the complexity of Non-Linear Programming. In fact, for directly solving a mixed-integer problem it is normally necessary to solve an exponential number of continuous subproblems having the same non-linearity characteristics of the original problem, and non-linear continuous problems are by themselves already quite demanding to solve. One of the consequences of this is that the most performant approaches in the field work by partially decoupling the two complicating aspects of the considered problems. This results in the use of multi-layered and quite involved algorithms.

Therefore, in order to obtain better performances in the context of Mixed Integer Optimal Control, it is important to work at all levels of the involved solution procedures considering the fact that even small implementation details, such as the ordering function used in tree searches or the specific parameter setup of each of the involved solvers, might make a large difference. Moreover, the performances of any general mixed-integer optimization algorithm are ultimately bound by the underlying complexity of the field. Therefore, it is arguably beneficial to exploit as much as possible the specific structure of the particular class of problems of interest, as in our case, problems coming from optimal control applications.

1.3 Contributions

In this dissertation three different algorithms will be presented and evaluated with the help of a series of numerical experiments.

The first algorithm takes the name of Proximal Outer Approximation (POA). POA results from the merging and harmonization of the characteristics of two different approaches to mixed-integer convex optimization: Outer Approximation (OA) and Feasibility Pump (FP). The algorithm seeks to find an adaptive balance between the more feasibility-focused FP-like iterations and the more optimality-focused OA-like iterations. The collected empirical evidence suggests that, on the analyzed benchmark, Proximal Outer Approximation is often capable of yielding faster and more robust convergence with respect to the classical OA algorithm.

Next to be presented is the Mixed-Integer Real-Time Optimal Control (MIRT-OC) algorithm, a mixed-integer convex Model Predictive Control (MPC) approach that, inspired from the success of single-tree optimization schemes for Mixed-Integer Convex Optimal Control, reduces the whole MPC process to a single tree search. The algorithm, rather than dealing with each MPC iteration separately, makes use of two specialized routines in order to adapt the optimization data and the search-tree collected during one MPC iteration into optimization data and a partially explored search-tree for the subsequent iteration. Such strategy allows for maximal reuse of the collected information and, as confirmed by the performed numerical experiments, provides sizeable computational savings.

Finally, the thesis presents the Disjunctive Outer Approximation for Optimal Control (DOA-OC) scheme, a specialized variation of Outer Approximation capable of efficiently solving a certain class of mixed-integer non-convex optimal control problems. The algorithm exploits the peculiar structure of the optimization problems that arise from optimal control applications where the constraints result convex only on the continuous variables of the system. The distinctive feature of DOA-OC is its ability of efficiently generating linear relaxations for the non-convex constraints of the considered problems. The result is the possibility of avoiding the use of extremely expensive mixed-integer non-convex solution schemes or fiddly reformulations techniques, and thus, of obtaining vast reductions in computational complexity.

One schematic guide to the particular field of application of each method can be found in Table 1.1

Algorithm	**Field of Application**
POA	General Mixed-Integer Convex Optimization
MIRT-OC	Mixed-Integer Convex MPC
DOA-OC	Mixed-Integer Semi-Convex Optimal Control / MPC Disjunctive-Convex Optimal Control / MPC

Table 1.1: Presented Algorithms and Fields of Application

1.4 Overview

The remainder of this thesis is organized in five chapters besides the current. Chapter 2 consists of an introduction to the concepts which are the base of the work to be presented, Mixed-Integer Optimization and Mixed-Integer Optimal Control, together with the most common methodologies related to them. Chapters 3, 4 and 5, constitute the core of this dissertation: they consist of the presentation of, respectively, the Proximal Outer Approximation, the Mixed-Integer Real-Time Optimal Control and the Disjunctive Outer Approximation for Optimal Control algorithm. Finally, Chapter 6 concludes the dissertation: it summarizes the presented work and reports the final comments of the author as well as some suggestions for the further development of the discussed algorithms.

Chapter 2

Preliminaries and Basic Methodologies

This chapter serves the purpose of contextualizing the current dissertation. It consists of a sequence of brief discussions regarding some of the concepts that are at the base of the work to be presented. Initially, the reader will be introduced to the concept of optimal control and to one of the most common approaches to it: “Direct Multiple Shooting”. Then, the focus will shift towards the class of optimal control problems generally labeled as “Mixed-Integer”. In particular, towards the reasons for the relevance of such class as well as the challenges posed by the problems belonging to it. After that, a particularly successful approach to optimal control named “Model Predictive Control” will be presented. Finally, the two main existing algorithms for Mixed-Integer Convex Optimization will be explained, and a few words will be spent on Mixed-Integer Non-Convex Optimization.

2.1 Optimal Control

First of all, let us briefly and informally introduce the concept of dynamical systems. A dynamical system is an entity capable of changing its state over time according to its own characteristics and/or the characteristics of the environment it is immersed in. For example, consider a car traveling at a certain speed on the highway. If no torque is applied to the wheels, the speed of the car begins slowly decreasing by the effect of friction. Defining the state of the car in terms of its speed with respect to road surface, it is possible to observe the car continuously varying its state. Now, the state of a dynamical system gets defined according to the task at hand and needs not to be complete nor absolute. For instance, in our previous example, order to monitor the speed of the car it is possible to use different measurements, like the rotational speed of the wheels, and it is not necessary to include into our set of state variables the absolute position of the car nor the temperature of the engine.

A control task consists in figuring out a way to interact with a dynamical system in order to tailor its behavior towards the accomplishment of some goal. Going back to the previous example, suppose that the task at hand is to have the car reach and keep some desired speed, then, it is possible to manipulate the evolution of the state of the car by applying a variable torque to its wheels. In this case, the torque applied to the wheels is the control variable of the system and the torque profile over time is the selected control action (control signal). A complete introduction to the theory behind the field of classical control can be found in [80], a more practice-oriented introduction can be found in [68].

Optimal control is the branch of applied mathematics concerned with the development of methods aimed at finding a suitable set of control signals that, if applied to the system, guide its evolution is such a manner to maximize some measure of performance. For instance, optimal control may help us to find the torque profile allowing a car to remain within a certain interval of speeds with the lowest fuel consumption possible. Optimal control is an important aspect of many fields where the manner in which a task is executed is as important as the mere execution of the task is. For example, in aerospace applications, where the amount of fuel a vehicle can carry is limited, fuel economy is critical to the feasibility of many operations. An other example is industrial production: being able to produce a good in the least amount of time or with the least amount of energy can make the difference between a profitable production and non-profitable one. An introduction to the field of optimal control can be found in [31].

There exist several ways of formalizing optimal control problems, each tailored towards a specific type of dynamical system or task. However, in this dissertation we will focus on the following particular form:

$$\begin{aligned}
(\tilde{x}^*, \tilde{u}^*) := \quad & \underset{\tilde{x},\tilde{u}}{\operatorname{argmin}} \quad \int_{t=0}^{T} \mathbf{s}(\tilde{x}(t), \tilde{u}(t), t)\,\mathrm{d}t + \mathbf{f}(\tilde{x}(T)) \\
& \text{s.t.} \quad \tilde{x}(0) = \bar{x}_0 \\
& \qquad\;\; \begin{bmatrix} \dot{\tilde{x}}(t) = \mathbf{d}(\tilde{x}(t), \tilde{u}(t), t) \\ \mathbf{p}(\tilde{x}(t), \tilde{u}(t), t) \leq 0 \end{bmatrix}_{t \in [0,T]} \\
& \qquad\;\; \mathbf{e}(\tilde{x}(T)) \leq 0
\end{aligned} \tag{$\mathcal{P}_{\mathrm{OC}}$}$$

where $\tilde{x}$ and $\tilde{u}$ are array-valued functions defined over the interval $[0, T]$ representing, respectively, the state of the system and the applied controls.

At this point, it is useful to lay-down some terminology to be used in this dissertation:

- $\mathbf{s}(\tilde{x}(t), \tilde{u}(t), t)$: “path cost”.
- $\mathbf{f}(\tilde{x}(T))$: “end cost” or “tail cost”.
- $\bar{x}_0$: “initial state”.
- $\mathbf{e}(\tilde{x}(T)) \leq 0$: “end constraint” or “tail constraint”.
- $\dot{\tilde{x}}(t) = \mathbf{d}(\tilde{x}(t), \tilde{u}(t), t)$: “dynamic equation”.
- $\mathbf{p}(\tilde{x}(t), \tilde{u}(t), t) \leq 0$: “path constraints”

In general, problems like $\mathcal{P}_{\mathrm{OC}}$ need to be reformulated or approximated in order to be solved. The next subsection is devoted to the presentation of the approximation technique of choice for this thesis: Direct Multiple Shooting.

2.2 Direct Methods For Optimal Control

In the early years of optimal control as a main-stream topic of research (1950-1980), solutions were mostly obtained by means of indirect optimization methods[1] or via Dynamic Programming[2] ([3],[13]). However, on the one hand, indirect optimization methods are difficult to deploy and very sensitive to the characteristics of the specific problem instance to be solved. On the other hand, the usefulness of Dynamic Programming is limited to problems with low state dimensionality due to Bellman’s “curse of dimensionality”. As a consequence, starting from the 80s, a new class of approaches has risen to prominence: direct optimization methods.

Direct optimization methods derive tractable approximations to optimal control problems by considering only piece-wise defined control actions which can be determined using a finite number of parameters. Thanks to this choice,

---

[1] Indirect optimization methods use optimality conditions (conditions that any optimal solution satisfies) in order to transform the optimal control at hand into a boundary-value-problem (a differential equation coupled with a set of constraints on the value of the solution at the borders of its domain of definition) which is then solved analytically or numerically.

[2] Dynamic Programming uses the Bellman’s Principle of Optimality to recursively split an optimal control problem into a number of easier subproblems. Then, the solutions of the subproblems are used to compute an approximate optimal control action.

the original infinite-dimensional optimal control problem is translated into a finite-dimensional numerical optimization problem that can be readily solved using any appropriate out-of-the-shelf solver. The reason for the success of such class of methods is two-fold: firstly, their "translation procedure" from optimal control to optimization is not very sensitive to the characteristics of a problem instance and can be easily automated; secondly, they benefit from the abundance of results of a well established and extensively developed field as numerical optimization.

Direct optimization methods differ one from another on the way the evolution of the state of the system is handled. Each of the existing methods has its own advantages and flaws. However, for the sake of simplicity, in this dissertation only Direct Multiple Shooting will be considered[3].

2.2.1 Direct Multiple Shooting

Direct Multiple Shooting (DMS) was firstly presented by Bock and Plitt in 1984 ([15]). To date, DMS is possibly the most used approach to optimal control.

Before discussing the reasons for the success of DMS it is useful to briefly dive into its functioning. In DMS, the control action is assumed to be defined in a piece-wise fashion. For simplicity, assume the control action to be piecewise-constant. As a consequence, the first step to be taken in order to apply DMS to a control problem is to define a suitable partition of the time interval $[0, T]$. To this end, consider N sub-intervals $[t_k, t_{k+1}]$ such that:

$$0 = t_0 < t_1 < \cdots < T_{N-1} < T_N = T \tag{2.1}$$

Then, for each $k \in \{0, \cdots, N-1\}$ DMS proceeds as follows:

1. A variable u_k is defined to represent the value of $\tilde{u}(t)$ over the sub-interval $[t_k, t_{k+1}]$.
2. A variable x_k is defined to represent the value $\tilde{x}(t_k)$.
3. A function $\mathbf{d}_k(x_k, u_k)$ is defined as $\mathbf{x}_{x_k,u_k}(t_{k+1})$ where $\mathbf{x}_{x_k,u_k}$ is the parametric solution of the following initial-value problem:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{d}(\mathbf{x}(t), u_k, t) \\ \mathbf{x}(t_k) = x_k \end{cases} \tag{2.2}$$

---

[3] Nevertheless, all the presented results can be easily extended to all those other methods that generate block-sparse optimization problems as, for example, Direct Collocation

4. A function $\mathbf{s}_k(x_k, u_k)$ is defined as follows:

$$\mathbf{s}_k(x_k, u_k) := \int_{t_k}^{t_{k+1}} \mathbf{s}(\mathbf{x}_{x_k,u_k}(t), u_k, t)\,\mathrm{d}t \tag{2.3}$$

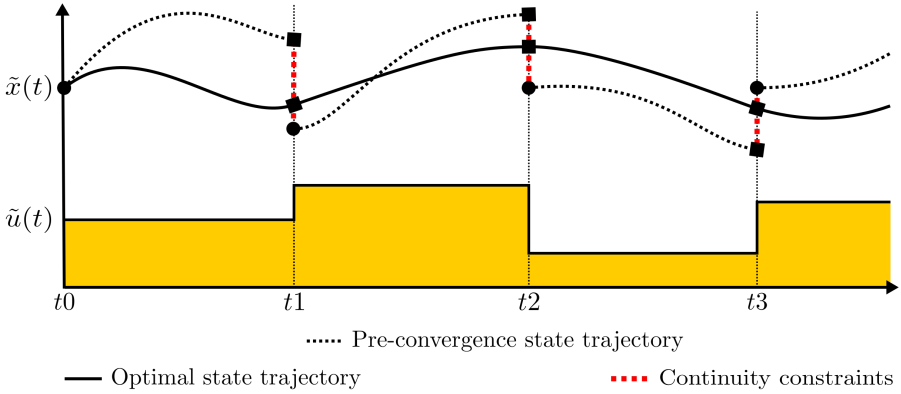
5. A function $\mathbf{p}_k(x_k, u_k)$ is defined as follows:

$$\mathbf{p}_k(x_k, u_k) := \mathbf{p}(x_k, u_k, t_k) \tag{2.4}$$

Finally, a last state variable x_N is created and the following optimization problem defined:

$$\begin{aligned}(x^*, u^*) := \quad & \underset{x,u}{\operatorname{argmin}} && \sum_{k=0}^{N-1} \mathbf{s}_k(x_k, u_k) + \mathbf{f}(x_N) \\ & \text{s.t.} && x_0 = \bar{x}_0 \\ &&& \begin{bmatrix} x_{k+1} = \mathbf{d}_k(x_k, u_k) \\ \mathbf{p}_k(x_k, u_k) \leq 0 \end{bmatrix}_{k=0}^{N-1} \\ &&& \mathbf{e}(x_N) \leq 0 \end{aligned} \tag{$\mathcal{P}_{\text{DMS}}$}$$

Figure 2.1: Representation of a typical control solution found via DMS including pre-convergence discontinuous state trajectory.



Now, $\mathcal{P}_{\text{DMS}}$ is an approximation of $\mathcal{P}_{\text{OC}}$. This is mainly due to the fact that the control action is limited to be piece-wise constant, but also because the path constraints are imposed only at the beginning of each sub-interval. Moreover, the vast majority of practical implementations of DMS use numerical integration to approximate the value of $\mathbf{s}_i(x_i, u_i)$ and $\mathbf{d}_i(x_i, u_i)$. Nevertheless, the accuracy of the DMS approximation can be arbitrarily increased by refining the defined partition of $[0, T]$ and by using error-controlled numerical integration schemes.

The optimization problems generated by DMS are fairly large. The more so the finer the used time partition is. However, they have a specific block-sparse structure which can be used by specialized solvers to significantly reduce the computational burden of the solution process. Additionally, the structure of $\mathcal{P}_{\text{DMS}}$ allows solvers to take advantage of a good initialization for the state trajectory (warm-starting). Another important feature of DMS is the fact that the continuity of the state trajectory is imposed only via a set of state continuity constraints: $x_{k+1} = \mathbf{d}_k(x_k, u_k)$. This prevents the non-linearity due to the dynamics of the system to accumulate along the whole time interval making easier to deal with strongly non-linear dynamical systems. One drawback of such approach is that many solvers (especially the most efficient ones) do not ensure the satisfaction of the problems constraints up until convergence. Consequently, if a solver is interrupted well before convergence, for example because of some time limit, there is the possibility of obtaining a plan based on a discontinuous state trajectory (which is clearly unrealistic). But in light of the high performances attainable with DMS, this issue interests only extremely fast applications.

In conclusion, the advantages of DMS can be summarized as follows:

- It is easily generalized and automated.
- It can use error-controlled numerical integration schemes to boost accuracy.
- Its structure allows for effective warm-starting.
- There exist efficient solvers specifically tailored to the kind of optimization problems it generates.
- It handles strongly non-linear dynamical systems with more ease than other methods.

2.3 Mixed-Integer Optimal Control

In order to define a Mixed-Integer Optimal Control Problem (MI-OCP) it is necessary to introduce two new vector valued functions: the "integer states" $\tilde{y}(t)$ and the "integer controls" $\tilde{v}(t)$. The new functions are allowed to take value only in a bounded subset, respectively: D_y and D_v, of the integer numbers, $\mathbb{Z}$. Inserting these new components in the formulation of $\mathcal{P}_{\mathrm{OC}}$ results in:

$$
\begin{aligned}
(\tilde{x}^*, \tilde{y}^*, \tilde{u}^*, \tilde{v}^*) := \quad & \underset{\tilde{x},\tilde{y},\tilde{u},\tilde{v}}{\operatorname{argmin}} \int\limits_{t=0}^{T} \mathbf{s}(\tilde{x}(t), \tilde{y}(t), \tilde{u}(t), \tilde{v}(t), t)\, \mathrm{d}t + \mathbf{f}(\tilde{x}(T), \tilde{y}(T)) \\
\text{s.t.} \quad & \tilde{x}(0) = \bar{x}_0 \\
& \tilde{y}(0) = \bar{y}_0 \\
& \left.\begin{array}{l}
\dot{\tilde{x}}(t) = \mathbf{d}(\tilde{x}(t), \tilde{y}(t), \tilde{u}(t), \tilde{v}(t), t) \\
\tilde{y}(t^+) = \mathbf{tr}(\tilde{x}(t), \tilde{y}(t), \tilde{u}(t), \tilde{v}(t), t) \\
\mathbf{p}(\tilde{x}(t), \tilde{y}(t), \tilde{u}(t), \tilde{v}(t), t) \leq 0 \\
(\tilde{y}(t), \tilde{v}(t)) \in D_y \times D_v \subset \mathbb{Z}^{n_y + n_v}
\end{array}\right|_{t \in [0,T]} \\
& \mathbf{e}(\tilde{x}(T), \tilde{y}(T)) \leq 0
\end{aligned}
\tag{$\mathcal{P}'_{\mathrm{OC}}$}
$$

where the symbol t^+ represents the time an infinitesimal instant after t.

At this point, applying DMS, with some adaptations[4], to $\mathcal{P}'_{\mathrm{OC}}$ leads to:

$$
\begin{aligned}
(x^*, y^*, u^*, v^*) := \quad & \underset{x,y,u,v}{\operatorname{argmin}} \sum_{k=0}^{N-1} \mathbf{s}_k(x_k, y_k, u_k, v_k) + \mathbf{f}(x_N, y_N) \\
\text{s.t.} \quad & x_0 = \bar{x}_0 \\
& y_0 = \bar{y}_0 \\
& \left[\begin{array}{l}
x_{k+1} = \mathbf{d}_k(x_k, y_k, u_k, v_k) \\
y_{k+1} = \mathbf{tr}_k(x_k, y_k, u_k, v_k) \\
\mathbf{p}_k(x_k, y_k, u_k, v_k) \leq 0 \\
(y_k, v_k) \in D_y \times D_v \subset \mathbb{Z}^{n_y + n_v}
\end{array}\right]_{k=0}^{N-1} \\
& \mathbf{e}(x_N, y_N) \leq 0
\end{aligned}
\tag{$\mathcal{P}'_{\mathrm{DMS}}$}
$$

Problems like $\mathcal{P}'_{\mathrm{DMS}}$ are generally referred to as Mixed-Integer Problems (MIPs). Mixed-integer optimization is a really powerful instrument to deal with

---

[4] Discrete states have to be treated carefully. In facts, unhandled state switches occurring in between DMS time discretization grid points may harm the accuracy of the DMS method. One solution to this is to use integration schemes able detect state switches and act accordingly. However, such schemes introduce discontinuity in the constraint set of the optimization problem obtained by DMS, possibly hindering the solution process. A less general but easier option is to consider only discrete states whose transition function $\mathbf{tr}(\dots)$ does not depend on the continuous states of the system. In this way, state switches are bound to occur on DMS time discretization grid points, and the integration issue is avoided altogether.

applications where the assigning a non-integer value to some decision variables makes little sense. However, in particular, $\mathcal{P}'_{\mathrm{DMS}}$ represents a control problem, and consequently, it falls in the field of Mixed-Integer Optimal Control (MIOC).

MIOC is useful in many applications, here are a few examples:

- Dynamic Resource Assignment: If, in an automated warehouse, the aim is to optimally plan the assignment of a number of robots to each of the necessary transportation tasks then, as new orders with different priority levels come in, the plan has to be extended and updated in a dynamic and adaptive manner.
- Hybrid Dynamic Control: Consider a running legged robot. During the run, the robot continuously switches between being and not being in contact with the ground. Clearly, the robot acquires a profoundly different dynamic behaviour depending on the phase it is in. Such discontinuity in the dynamics can be handled via the definition of an appropriate binary state.
- Policy optimization: A famous example concerns the regulation of fishing in a sea region ([73]). Assuming that the population growth of the fish follows a specific dynamical equation (e.g. the Lotka-Volterra's equation) Mixed-Integer Optimal Control could be used in order to establish the periods of the year in which to forbid fishing in order to stabilize the fish population around a healthy level while also preserving the productivity of fishing.
- Control of Hybrid Systems: The drive-train of many land vehicles features stepped gearboxes. This kind of transmissions limit the usable engine-to-wheels gear-ratios to a predetermined set of choices. Therefore, the car dynamics is best modeled using discrete variables.

When dealing with Mixed-Integer Optimal Control, the greatest challenge to overcome concerns the computational complexity of the involved optimization methods. In fact, the complexity of Mixed-Integer Optimization grows exponentially with the problem size. In terms of optimal control, this limits either the size of the time interval that can be considered at once or the maximum switching frequency for the integer states/controls.

Nevertheless, despite its limitations, MIOC can still generate high quality controlling strategies in many applications. Hence, MIOC constitutes an interesting and fairly wide-spread field of research.

2.4 Mixed-Integer Model Predictive Control

As said, Mixed-Integer Optimal Control is hard. Still, billions of people around the world are capable of properly managing the stepped-gearbox of a car without the help of a super-computer. The reason for this resides in the fact that humans do not normally plan the exact sequence of gears they will select during a whole trip beforehand. Rather, drivers change gears reactively (e.g. whenever the engine gives signs of lower efficiency) or slightly proactively (e.g. whenever a curve or a change of speed is foreseen). This approach has three main advantages: firstly, it prevents brains from over-heating, secondly, it enables the driver to take into account changes in the control problem that cannot be predicted a priori (e.g. due to the behavior of other drivers), and, thirdly, does not impose any limit to the length of the time interval over which the control problem is defined (it could even be infinite).

Such slightly proactive approach to optimal control is present also in the literature with the name of Model Predictive Control (MPC). Since the 1980s, MPC is commonly used to tackle complex optimal control problems where the methods from classical control theory would fail (e.g. problems with strong non-linearities or restrictive constraints). MPC is a (potentially online) scheme that incrementally builds a control strategy for a long-horizon optimal control problem by, at each time step, extracting a control action from the solution of a short-horizon subproblem. Generally, MPC does not provide optimal solutions but it is possible to run it for as long as needed with substantially constant memory and time requirements across the iterations. Such feature qualifies MPC as an interesting tool for finding close-to-optimal solutions to mixed-integer optimal control problems. A complete presentation of MPC, its characteristics and its applications is well beyond the scope of this introduction. Therefore, in this section only the aspects of MPC that are the most interesting from the perspective of the Mixed-Integer Optimal Control approach presented so far will be discussed. A complete introduction to MPC can be found in [72].

2.4.1 Mixed-Integer MPC in offline conditions

Assume perfect a priori knowledge of the system to control and its environment, i.e. there are no unmodeled or unpredictable effects that would impose changes to the control problem during the optimization. Then, assume that we are interested in computing a sufficiently good (not necessary the globally optimal) feasible point for the following problem obtained via DMS:

$$
\begin{aligned}
(x^*, y^*, u^*, v^*) := \quad & \underset{x,y,u,v}{\text{argmin}} && \sum_{k=0}^{N-1} \mathbf{s}_k(x_k, y_k, u_k, v_k) \\
& \text{s.t.} && x_0 = \bar{x}_0 \\
& && y_0 = \bar{y}_0 \\
& && \begin{bmatrix} x_{k+1} = \mathbf{d}_k(x_k, y_k, u_k, v_k) \\ y_{k+1} = \mathbf{tr}_k(x_k, y_k, u_k, v_k) \\ \mathbf{p}_k(x_k, y_k, u_k, v_k) \le 0 \\ (y_k, v_k) \in D_y \times D_v \subset \mathbb{Z}^{n_y + n_v} \end{bmatrix}_{k=0}^{N-1}
\end{aligned}
\tag{$\mathcal{P}''_{\text{DMS}}$}
$$

In order to tackle $\mathcal{P}''_{\text{DMS}}$ in a MPC fashion, a sequence of shorter subproblems are defined:

$$
\begin{aligned}
(x^{(j)}, y^{(j)}, u^{(j)}, v^{(j)}) := \quad & \underset{x,y,u,v}{\text{argmin}} && \sum_{k=I_j}^{E_j-1} \mathbf{s}_k(x_k, u_k) + \mathbf{f}_j(x_{E_j}, y_{E_j}) \\
& \text{s.t.} && x_{I_j} = \bar{x}_{I_j} \\
& && y_{I_j} = \bar{y}_{I_j} \\
& && \begin{bmatrix} x_{k+1} = \mathbf{d}_k(x_k, y_k, u_k, v_k) \\ y_{k+1} = \mathbf{tr}_k(x_k, y_k, u_k, v_k) \\ \mathbf{p}_k(x_k, y_k, u_k, v_k) \le 0 \\ (y_k, v_k) \in D_y \times D_v \subset \mathbb{Z}^{n_y + n_v} \end{bmatrix}_{k=I_j}^{E_j-1} \\
& && \mathbf{e}_j(x_{E_j}, y_{E_j}) \le 0
\end{aligned}
\tag{$\mathcal{P}^{(j)}_{\text{MPC}}$}
$$

where:

- $H_j := E_j - I_j > 0$ is the length of the (discretized) time window considered by the j-th subproblem (horizon length).
- $I_0 \le I_1 \le \cdots \le I_M$, i.e. the subproblems are numbered in ascending order with respect to their initial time.
- $\forall j \in \{0, \cdots, M\}$: $I_{j+1} \le E_j$, i.e. the time windows of two subsequent subproblems intersect.
- $I_0 = 0$ and $E_M = T$, i.e. the union of all the considered time windows covers the whole interval $[0, T]$.
- $\bar{x}_{I_j}$ is an initial state derived from the solution of the $(j-1)$-th subproblem.

- $\mathbf{f}_j(x_{E_j}, y_{E_j})$ and $\mathbf{e}_j(x_{E_j}, y_{E_j}) \leq 0$ are, respectively, a tail cost and a tail constraint that were not present in $\mathcal{P}''_{\mathrm{DMS}}$. This type of costs/constraints are often added to MPC subproblems for specific reasons that will be discussed later in this section.

The MPC procedure starts by defining $\bar{x}_0 := x_0$. Then, in each iteration, the problem $\mathcal{P}^{(j)}_{\mathrm{MPC}}$ is solved obtaining the point $(x^{(j)}, y^{(j)}, u^{(j)}, v^{(j)})$, and the values:

$$\{x^{(j)}_{I_j}, \cdots, x^{(j)}_{I_{j+1}}\},\ \{y^{(j)}_{I_j}, \cdots, y^{(j)}_{I_{j+1}}\},\ \{u^{(j)}_{I_j}, \cdots, u^{(j)}_{I_{j+1}-1}\},\ \{v^{(j)}_{I_j}, \cdots, v^{(j)}_{I_{j+1}-1}\}$$

are stored. In this way, if each of the subproblems results feasible, the procedure is able to iteratively build a state trajectory and a control action for $\mathcal{P}''_{\mathrm{DMS}}$.

However, even when applied to a feasible problem $\mathcal{P}''_{\mathrm{DMS}}$, MPC does not intrinsically ensure the feasibility of each subproblem and therefore, if not carefully deployed, the scheme might not be able to produce meaningful solutions. Moreover, it is not straightforward to predict in advance how far from the global optimum an eventually obtained solution would lie.

In order to ensure the feasibility of each iteration, the additional tail cost $\mathbf{f}_j(x_{E_j}, y_{E_j})$ and/or the tail constraint $\mathbf{e}_j(x_{E_j}, y_{E_j})$ can be designed, respectively, to discourage or to prevent the solution of $\mathcal{P}^{(j)}_{\mathrm{MPC}}$ from ending up in a state that would make $\mathcal{P}^{(j+1)}_{\mathrm{MPC}}$ infeasible. Further, if $\mathbf{f}_j(x_{E_j}, y_{E_j})$ defines a look-ahead cost, i.e. it approximates the optimal objective value for the residual part of the control task (as in Approximate Dynamic Programming [13]), it can be used to deduce bounds on the sub-optimality of the eventual MPC solution.

Remark: Originally, MPC was designed for online applications and using it offline might seem unnecessary. But actually, for difficult control applications, MPC can allow for the solution of problems that are unsolvable otherwise. For example, while the complexity of solving $\mathcal{P}''_{\mathrm{DMS}}$ all at once is exponential in N, the complexity of solving M MPC iterations is linear in M.

2.4.2 Mixed-Integer MPC in online conditions

In control applications, it is often the case that the mathematical model for a system does not allow for perfect predictions of the model behavior. Moreover, the environment the system is immersed in might present some degree of unpredictability. In those cases, it is necessary to constantly monitor for possible prediction errors in the state of the system/environment and, consequently, adapt the future control actions just before executing them.

In online conditions, each MPC iteration runs while the control action obtained by the last iteration is being executed. More explicitly, assume that in iteration $j-1$ a control action $\left(u^{(j-1)}, v^{(j-1)}\right)$ was found. Then, the MPC controller would apply to the plant the initial portion of such control action:

$$\mathcal{C}^{(j-1)} := \left[\left(u^{(j-1)}_{I_{j-1}}, v^{(j-1)}_{I_{j-1}}\right), \left(u^{(j-1)}_{I_{j-1}+1}, v^{(j-1)}_{I_{j-1}+1}\right), \cdots, \left(u^{(j-1)}_{I_j-1}, v^{(j-1)}_{I_j-1}\right)\right]$$

while solving the j-th MPC subproblem.

Assuming for simplicity that the disturbances in the system/environment are random, the "actual" discretized dynamic of the system could be represented as follows:

$$\hat{x}_{k+1} := \mathbf{d}_k(\hat{x}_k, u_k) + \Delta\hat{x}_{k+1} \tag{2.5}$$

where $\Delta\hat{x}_{k+1}$ is a random variable of some form. Further, assume that it is possible to measure/estimate the state of the system but that such measurement/estimation is subject to some random error as well. Then, at the beginning of each MPC iteration j, the following quantity will be available to the controller:

$$\xi_{I_{j-1}} := \hat{x}_{I_{j-1}} + \Delta\xi_{j-1} \tag{2.6}$$

for some random variable $\Delta\xi_{j-1}$. As a consequence, in order to take into account this information, the MPC subproblems need to be redefined:

$$\begin{aligned}
(x^{(j)}, y^{(j)}, u^{(j)}, v^{(j)}) := \quad & \underset{x,y,u,v}{\operatorname{argmin}} \quad \sum_{k=I_j}^{E_j-1} \mathbf{s}_k(x_k, u_k) + \mathbf{f}_j(x_{E_j}, y_{E_j}) \\
& \text{s.t.} \quad x_{I_j} = \mathbf{D}_{j-1,j}(\hat{\xi}_{I_{j-1}}) \\
& \qquad\; y_{I_j} = \bar{y}_{I_j} \\
& \qquad\; \begin{bmatrix} x_{k+1} = \mathbf{d}_k(x_k, y_k, u_k, v_k) \\ y_{k+1} = \mathbf{tr}_k(x_k, y_k, u_k, v_k) \\ \mathbf{p}_k(x_k, y_k, u_k, v_k) \le 0 \\ (y_k, v_k) \in D_y \times D_v \subset \mathbb{Z}^{n_y+n_v} \end{bmatrix}_{k=I_j}^{E_j-1} \\
& \qquad\; \mathbf{e}_j(x_{E_j}, y_{E_j}) \le 0
\end{aligned} \tag{$\hat{\mathcal{P}}^{(j)}_{\text{MPC}}$}$$

where $\mathbf{D}_{j-1,j}(\hat{\xi}_{I_{j-1}})$ is a prediction of the state at time I_j obtained assuming measured state $\xi_{I_{j-1}}$ to be accurate and considering the future application of the control action $\mathcal{C}^{(j-1)}$.

$\hat{\mathcal{P}}^{(j)}_{\text{MPC}}$ is identical to $\mathcal{P}^{(j)}_{\text{MPC}}$ except for the definition of the initial point for the state. Including in the formulation of the MPC subproblems information obtained at run-time constitutes a correction mechanism (or feedback) that prevents the controller internal representation of the state of the system from "drifting" away from the actual state. This characteristic allows online MPC to be robust with respect to mild model errors or external disturbances.

Online-MPC has few limitations with respect to its offline counterpart. The most important one regards the allowed time per iteration. In fact, the maximum time that can be spent in solving $\hat{\mathcal{P}}^{(j)}_{\text{MPC}}$ is restricted by the size of the interval of time between t_{I_j} and $t_{I_{j+1}}$. This implies that there might be not enough time to identify an optimal point for $\hat{\mathcal{P}}^{(j)}_{\text{MPC}}$. Nevertheless, it is often safe to assume that during each iteration at least one feasible point for the current MPC subproblem is found.

Remark: When using online-MPC one can easily assume that the MPC controller could run forever. This gives rise to a new concern: stability. To give an informal definition of stability we can say that a system is stable if its state remains within a bounded region at all time. Stability is important to avoid malfunctioning in the controller or critical failures of the system due "unreasonably" large values of the state (e.g. the temperature of the core of a nuclear plant cannot be allowed to raise above a given threshold). One common way to ensure the stability in MPC is to design the terminal costs/constraints in such a way that the optimal solution of each MPC subproblem would always drive the state closer to a certain desired region according to some measure of distance (e.g. a Lyapunov Function).

2.5 Mixed-Integer Optimization

While MPC alleviates the complexity issue of Mixed-Integer Optimal Control, it does not solve it all together. In fact, on the one hand, in MPC the quality of the obtained control action increases with the length of the control horizon of the subproblems, on the other hand, the computational cost of each MPC iteration depends exponentially on the size of the considered subproblem. Consequently, it is still important to be able to rely on efficient approaches to Mixed-Integer Optimization. For this reason, this thesis presents also results in the field of pure Mixed-Integer Optimization.

In the present and the next section, the optimal control nature of the problems to be solved will be momentarily set aside in order to explain some basic concepts regarding optimization and Mixed-Integer Optimization. These concepts will be helpful for understanding of the work presented in this dissertation.

Let us start by defining a general form for a mixed-integer optimization problem:

$$\begin{aligned} (x^*, y^*) = \quad & \underset{x,y}{\operatorname{argmin}} \quad c(x, y) \\ & \text{s.t.} \qquad g(x, y) \leq 0 \\ & \qquad\qquad y \in \mathbb{Z}^{n_y} \end{aligned} \tag{$\mathcal{P}_{\mathrm{MI}}$}$$

where:

- c is either a linear function or a quadratic function.
- g is a continuously differentiable function.
- The constraints $g(x, y) \leq 0$ imply the following box constraints on y:

$$l_y \leq y \leq u_y \text{ where } \|l_y\|_\infty, \|u_y\|_\infty < \infty$$

2.5.1 Nomenclature

Problems like $\mathcal{P}$ are generally referred to as Mixed-Integer Problems (MIPs). Still, depending on the particular characteristics of the problem at hand, it is useful to define a further categorization:

MILP: If c and g are linear, $\mathcal{P}$ is said to be Mixed-Integer Linear.

MIQP: If only g is linear, $\mathcal{P}$ is said to be Mixed-Integer Quadratic.

MINLP: If g is non-linear, $\mathcal{P}$ is said to be Mixed-Integer Non-Linear. [5]

MICP: If c and g are convex, $\mathcal{P}$ is said to be Mixed-Integer Convex.

2.5.2 Feasibility and Optimality in Mixed-Integer Optimization

In the formulation of problem $\mathcal{P}$, two kinds of constraints are present: the explicit constraints: $g(x, y) \leq 0$, and the integer requirements $y \in \mathbb{Z}^{n_y}$. Consequently, in the remainder of this thesis, given a couple (x, y), we will distinguish between three concepts of feasibility:

---

[5] In case of a problem having a non-linear objective function, the problem can be brought to adhere to the structure of $\mathcal{P}$ via an epigraph reformulation of its objective.

Constraint-feasibility: $(\tilde{x}, \tilde{y})$ is "constraint-feasible" for $\mathcal{P}$ if $g(\tilde{x}, \tilde{y}) \leq 0$.

Integer-feasibility: $(\tilde{x}, \tilde{y})$ is "integer-feasible" for $\mathcal{P}$ if $\tilde{y} \in \mathbb{Z}^{n_y}$

Feasibility: $(\tilde{x}, \tilde{y})$ is "feasible" for $\mathcal{P}$ if it is simultaneously constraint- and integer-feasible for $\mathcal{P}$.

Moreover, we will define the feasible set of $\mathcal{P}$ as the set of all the couples (x, y) such that: (x, y) is feasible for $\mathcal{P}$. Further, with a slight abuse of terminology, we will say that a discrete assignment $\bar{y}$ is feasible for $\mathcal{P}$ if there exists $\bar{x}$ such that $(\bar{x}, \bar{y})$ is feasible for $\mathcal{P}$.

Finally, in optimization it is usual to distinguish between local and global optima. However, in Mixed-Integer Optimization, only the concept of global optimality has an unambiguous and universally accepted definition: a global optimum (or globally optimal point) for $\mathcal{P}$ is a feasible point for $\mathcal{P}$, (x^*, y^*), such that:

$$c(x^*, y^*) \leq c(x, y), \;\; \forall (x, y) \text{ feasible for } \mathcal{P}$$

The Cover Property: Consider a set of restrictions $\{\hat{\mathcal{P}}_i\}$ of $\mathcal{P}$ such that the union of their feasible sets coincide with the feasible set of $\mathcal{P}$. Collecting a globally optimal point for each of the $\hat{\mathcal{P}}$'s and then taking the best among the collected points we obtain a globally optimal point for $\mathcal{P}$.

2.5.3 Relaxations and Restrictions

Denote with (x^*, y^*) a globally optimal point for $\mathcal{P}$.

Relaxation: Any optimization problem $\tilde{\mathcal{P}}$ that can be obtained from $\mathcal{P}$ by expanding its feasible set is called a relaxation of $\mathcal{P}$. If a relaxation of $\mathcal{P}$, $\tilde{\mathcal{P}}$, has a global optimum in $(\tilde{x}^*, \tilde{y}^*)$ then:

$$c(\tilde{x}^*, \tilde{y}^*) \leq c(x^*, y^*) \tag{2.7}$$

Restriction: Any optimization problem $\bar{\mathcal{P}}$ that can be obtained from $\mathcal{P}$ by reducing its feasible set is called a restriction of $\mathcal{P}$. If a restriction of $\mathcal{P}$, $\bar{\mathcal{P}}$, has a global optimum in $(\bar{x}^*, \bar{y}^*)$ then:

$$c(\bar{x}^*, \bar{y}^*) \geq c(x^*, y^*) \tag{2.8}$$

Moreover, if a problem $\mathcal{P}$ is infeasible then so are all its restrictions.

Note: The notions of relaxation and restriction define a partial ordering in the set of optimization problems:

Reflexivity: $\mathcal{P}$ is simultaneously a relaxation and a restriction of itself.

Anti-symmetry: If $\mathcal{P}_1$ is a relaxation of $\mathcal{P}_2$ then $\mathcal{P}_2$ is a restriction of $\mathcal{P}_1$, and vice versa

Transitivity: If $\mathcal{P}_1$ is a relaxation of $\mathcal{P}_2$ and $\mathcal{P}_2$ is a relaxation of $\mathcal{P}_3$ then $\mathcal{P}_1$ is a relaxation of $\mathcal{P}_3$.

Notable Examples:

- The continuous relaxation of $\mathcal{P}$ can be obtained by dropping the integer requirements: $y \in \mathbb{Z}^{n_y}$.
- A linearly-constrained relaxation of $\mathcal{P}$ can be obtained by substituting the constraint $g(x, y) \leq 0$ with a linear constraint: $L_x x + L_y y + l \leq 0$, where, for all possible couples (x, y), $L_x x + L_y y + l \leq g(x, y)$. [6]
- A discrete-domain restriction for $\mathcal{P}$ can be obtained by adding to its formulation a number of constraints of the form : $y_k \in [\underline{y}'_k, \overline{y}'_k]$.

2.6 Methods for Mixed-Integer Convex Optimization

The demanding complexity level of Mixed-Integer Optimization calls for the continuing development of more efficient algorithms. As a consequence, there exists a high number of different solution approaches for it. Nevertheless, each one of these approaches is actually just a specific implementation of one out of few theoretical methodologies. This section presents the two main methodologies for Mixed-Integer Convex Optimization to date: Branch&Bound and Outer Approximation.

Despite the fact that in theory all MIPs are equally difficult to solve, in practical terms, the characteristics of each subclass have a large impact on how rapidly and cheaply a problem instance can be solved. In this regard, one important class of MIPs is constituted by those problem whose constraint sets and objective function are convex. Such characteristic entails two important properties:

---

[6] Often, if $c(x, y)$ is linear, we will refer to the linearly-constrained relaxation of $\mathcal{P}$ with the term "linear relaxation".

1. The continuous relaxation of a MICP is a continuous convex problem.
2. For any MICP $\mathcal{P}$, substituting $g(x, y)$ with any number of first-order Taylor approximations for it computed around an arbitrary set of points: $\{(\bar{x}_i, \bar{y}_i)\}_{i \in I}$, leads to the definition of a relaxation of $\mathcal{P}$.

Since linearly-constrained problems are easier to solve than non-linearly-constrained ones, and convex problems are easier to solve than non-convex ones, what the two properties above mean is that it is "easy" to compute lower bounds for the optimal objective value of a MICP. Notably, the first property is vital for the efficiency of Branch&Bound while the second is at the base of Outer Approximation.

2.6.1 The Branch&Bound algorithm

The simplest way of solving a mixed-integer convex problem $\mathcal{P}$ consists in collecting all the possible value assignments for the discrete variables in the problem formulation, and solving all the continuous problems that can be obtained by fixing the discrete variables of $\mathcal{P}$ to a single assignment. Then, the best of the obtained solutions would be a globally optimal point for $\mathcal{P}$. However, such brute-force approach is enormously cumbersome for obvious reasons.

Unfortunately, it is not possible to tell a priori if a specific assignment would generate a good, a bad or no solution at all. Therefore, a complete search on the set of all the discrete assignments is unavoidable. Anyhow, the search can be performed in a "informed" or "smart" way. In particular, it is possible to avoid the explicit enumeration of all the assignments by grouping them in meaningful subsets for which general bounds on the quality of the solutions obtainable from each group can be deduced.

In this direction goes the Branch&Bound (B&B) algorithm. B&B was described for the first time by Land and Doing in 1960 in the context of discrete programming ([58]). However, the name "Branch and Bound" appears for the first time in a paper by Little et. al. addressing the "Traveling Salesman Problem" in 1963 ([61]). From the sixties onward, B&B has become the most prominent method for the solution of many NP-hard problems.

The B&B algorithm explores the whole set of possible discrete assignments for $\mathcal{P}$ but avoids explicit enumeration of entire portions of the set whenever such portions are proven to contain only suboptimal or infeasible assignments. More specifically, B&B searches for a globally optimal point for $\mathcal{P}$ exploring an implicitly defined tree where the nodes represent optimization problems. The root node of the tree represents the continuous relaxation of $\mathcal{P}$, each non-root node of the tree is constructed from its parent by adding a domain restriction on one of the discrete variables of $\mathcal{P}$ and the leaf nodes represent all the possible problems that can be obtained from $\mathcal{P}$ by fixing the discrete variables to a specific assignment. This structure helps B&B to limit the time spent exploring useless portions of the assignment space.

During the search, B&B maintains a queue of "active" subproblems. The active subproblems represent the portion of the assignment space still to explore. At the beginning, the queue contains only the continuous relaxation of $\mathcal{P}$ (root). Then, at each iteration, B&B removes one subproblem from the queue and, depending on its feasibility and optimal objective value, decides whether or not to "branch" on it to obtain a number of "smaller" new subproblems. Finally, the obtained subproblems, if any, are inserted in the queue and a new iteration starts.

In the most classical setting, the branching procedure applied to the subproblem $\hat{\mathcal{P}}$ starts by selecting one of the discrete variables of $\mathcal{P}$, y_k, whose value in the optimal solution of $\hat{\mathcal{P}}$, $\hat{y}_k$, is not integer. Then, two copies of $\hat{\mathcal{P}}$ are produced and to each of them a discrete-domain restriction is added: in one copy, the value of y_k is constrained to take value below the integer that is obtained by rounding-down $\hat{y}_k$, while, in the other copy, the value of y_k is constrained to take value above the integer that is obtained by rounding-up $\hat{y}_k$. A simplified pseudo-code for the standard branching procedure can be found in algorithm 1.

Algorithm 1: B&B: Branch

```
Input  : A B&B subproblem $\hat{\mathcal{P}}$
Output : Children subproblems $\hat{\mathcal{P}}_1$ and $\hat{\mathcal{P}}_2$
Select a discrete variable $d_i$ whose value $\hat{d}_i$ at the optimal solution of $\hat{\mathcal{P}}$ is not integer;
Create two children subproblems $\hat{\mathcal{P}}_1$ and $\hat{\mathcal{P}}_2$ by adding to the formulation of $\hat{\mathcal{P}}$ the constraints $y_k \leq \text{floor}(\hat{y}_k)$ and $y_k \geq \text{ceil}(\hat{y}_k)$, respectively;
```

While exploring the tree, B&B keeps in memory the best feasible point encountered (incumbent solution). B&B stops whenever the queue gets empty or a user-defined time limit is exceeded. If the algorithm terminates having emptied the queue and, in the meanwhile, an incumbent solution was found, then such solution is a globally-optimal point for $\mathcal{P}$. Contrarily, if when the queue gets empty no incumbent solution exists, then $\mathcal{P}$ is guaranteed to be infeasible. Further, if B&B terminates because of the time limit, the incumbent solution, if any, is a feasible point for $\mathcal{P}$ whose objective value can be guaranteed to lie within an estimable distance from the globally optimal objective value.

Denoting with $\mathcal{O}_{\text{obj.}}(\hat{\mathcal{P}})$ and with $\mathcal{O}_{\text{sol.}}(\hat{\mathcal{P}})$, respectively, the optimal objective value and the obtained optimal point for problem $\hat{\mathcal{P}}$, a pseudo-code for B&B can be found in Algorithm 2.

Algorithm 2: Branch&Bound

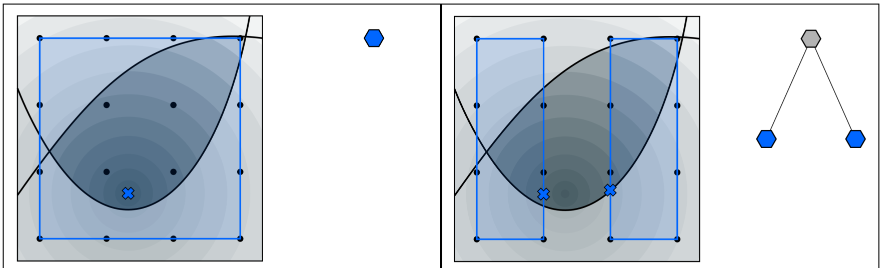
```
Input  : A Mixed-integer convex problem $\mathcal{P}$, a time limit $\bar{T}$
Output : A globally-optimal solution for $\mathcal{P}$
Insert into Queue the continuous relaxation of $\mathcal{P}$;
$UpB := \infty$ ;
while Queue is not empty and $time \leq \bar{T}$ do
    Extract the first problem $\hat{\mathcal{P}}$ from Queue;
    Solve $\hat{\mathcal{P}}$ until convergence or until proving $\mathcal{O}_{\text{obj.}}(\hat{\mathcal{P}}) \geq UpB$;
    if $\hat{\mathcal{P}}$ is feasible and $\mathcal{O}_{obj.}(\hat{\mathcal{P}}) < UpB$ then
        if $\hat{\mathcal{P}}$ is integer-feasible then
            $I := \mathcal{O}_{\text{sol.}}(\hat{\mathcal{P}}),\ UpB := \mathcal{O}_{\text{obj.}}(\hat{\mathcal{P}})$
        else
            $Children :=$ Branch($\hat{\mathcal{P}}$);
            for $\hat{\mathcal{P}}_j \in Children$ do
                Insert $\hat{\mathcal{P}}_j$ into Queue
            end
        end
    end
end
```

In B&B some of the subproblems do not generate children. This happens if the subproblem $\hat{\mathcal{P}}$ satisfies at least one of the following conditions:

- **Infeasibility:** $\hat{\mathcal{P}}$ is infeasible.
- **Sub-optimality:** The optimal objective value of $\hat{\mathcal{P}}$ is greater or equal than the objective value of the incumbent solution.
- **Integrality:** The solution of $\hat{\mathcal{P}}$ is integer-feasible.

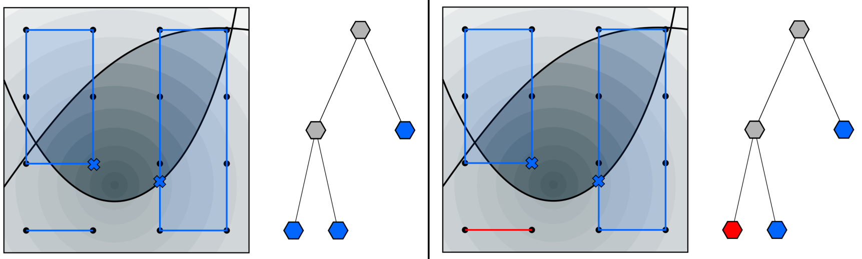
Such phenomenon is called pruning. Whenever a subproblem is pruned, the full sub-tree of its descendants is removed from the search. Pruning is a critical aspect for the efficiency of B&B as it reduces the search space while keeping in it at least one globally optimal point.

At the left of each image, the problem to be solved is represented: the black dots indicate the possible discrete assignments, the central darker region represents the region of constraint-feasibility, and the circular shades depict a quadratic objective function. At the right of each image, the status of the B&B tree is represented. The aim of the optimization is to find the constraint-feasible discrete assignment closest to the center of the circular shades.



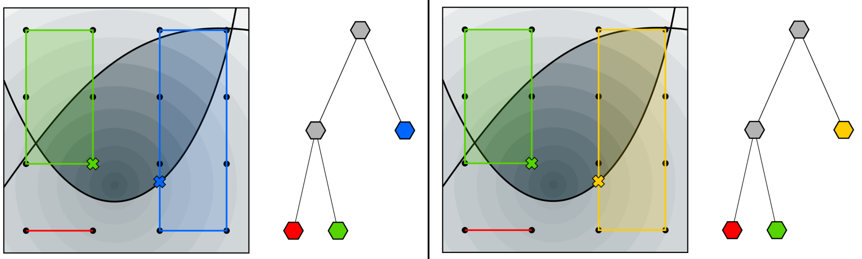
The first node to explore is the continuous relaxation of $\mathcal{P}$. The discrete variables can take any value within the blue square. The solution is marked with a blue cross.

Since the optimal solution for the root is not integer-feasible, two new subproblems are obtained via branching.



The solution of the leftmost subproblem is still not integer-feasible: a new round of branching is applied.

The leftmost subproblem is found infeasible: pruning is applied.



The solution of the highlighted node is integer-feasible: the node is pruned but the solution is kept as the incumbent solution.

The objective value of the last subproblem left open is higher than the objective value of the incumbent solution: the node is pruned for suboptimality.

⬢ = inactive ⬢ = active ⬢ = infeasible ⬢ = solution? ⬢ = suboptimal

Figure 2.2: Behaviour of B&B on a simple problem $\mathcal{P}$.

In order to understand the logic behind pruning it is important to note that all the descendants of a problem $\hat{\mathcal{P}}$ in the B&B tree are restrictions of it. Therefore, their optimal objective value cannot be lower than the one of $\hat{\mathcal{P}}$. Moreover, if $\hat{\mathcal{P}}$ is infeasible, so all its descendants are. As a consequence, if $\hat{\mathcal{P}}$ is found to be suboptimal or infeasible, it is possible to extend this conclusion to all its descendants, and accordingly, remove them from the search. Similarly, regarding the third rule of pruning, the fact the solution found for $\hat{\mathcal{P}}$ is already integer-feasible entails that no descendant of $\hat{\mathcal{P}}$ may lead to a better solution.

In practice, the efficiency of B&B depends heavily on the selection of index k in Algorithm 1 and on the ordering of *Queue* in Algorithm 2. Nevertheless, analyzing these aspects goes beyond the scope of this brief introduction. Thus, for a more in depth discussion on B&B, we refer the interested reader to [45].

B&B has proven to be a very efficient algorithm for the solution of MILPs and MIQPs, and it has established itself as the gold-standard in these contexts. However, in the mixed-integer non-linear setting (MINLPs), B&B does not shine for efficiency. In fact, in this setting, after each branching step, two expensive non-linear optimizations have to be performed. Despite all, B&B constitutes an important building stone for the majority of the mixed-integer non-linear solvers, although rarely as a stand-alone approach.

2.6.2 The Outer Approximation Algorithm

If $\mathcal{P}$ is a convex MINLP, there exist algorithms generally more efficient than B&B. The most prominent of these algorithms is called Outer Approximation (OA) and it was introduced by Duran and Grossmann in 1986 [33]. OA is a multi-step iterative procedure that generates an alternating sequence of integer-feasible and constraint-feasible guesses for (x, y) which are guaranteed to converge together to a globally optimal point for $\mathcal{P}$ ([39],[59],[45],[60]). After more than thirty years, the algorithmic ideas of the original OA method are still at the core of some of the most competitive MICP solution approaches. Supporting evidence for that can be found on recent survey from Kronqvist et al. [56]. The survey shows that, on the selected benchmark, the most robust and performant solvers to date are those based on some form of outer approximation, as for example SHOT ([57],[63]) and AOA ([52]).

Assume, without loss of generality[7], that the objective of $\mathcal{P}_{\mathrm{MI}}$ is linear and denote it with: $C_x^T x + C_y^T y$. OA starts by solving the continuous relaxation of $\mathcal{P}$. From the obtained solution, $(\tilde{x}_0, \tilde{y}_0)$, OA obtains an integer-feasible (but normally constraint-infeasible) guess $\bar{y}_0$ on the value of the discrete variables of the MICP via a rounding step. Then, the following quantities are defined:

- A lower bound on the optimal value for $\mathcal{P}$:

$$V^{\mathrm{LB}} := -\infty$$

- An upper bound on the optimal value for $\mathcal{P}$:

$$V^{\mathrm{UB}} := \infty$$

- An initial set of linear constraints:

$$\begin{array}{lcl} L_{x,0} & := & \frac{\partial g}{\partial x}(\tilde{x}_0, \tilde{y}_0) \\ L_{y,0} & := & \frac{\partial g}{\partial y}(\tilde{x}_0, \tilde{y}_0) \\ l_0 & := & g(\tilde{x}_0, \tilde{y}_0) - \dfrac{\partial g}{\partial x, y} g(\tilde{x}_0, \tilde{y}_0) \left[\begin{array}{c} \tilde{x}_0 \\ \tilde{y}_0 \end{array} \right] \end{array} \tag{2.9}$$

Then, at each iteration k, OA attempts at solving the following NLP:

$$\begin{array}{rcll} \tilde{x}_k & := & \underset{x}{\operatorname{argmin}} & C_x^T x + C_y^T \bar{y}_{k-1} \\ & & \text{s.t.} & g(x, \bar{y}_{k-1}) \leq 0 \\ & & & x \in \mathbb{R}^{n_x} \end{array} \tag{$\mathcal{P}_1$}$$

If the above problem is feasible, i.e. $\bar{y}_{k-1} \in Y$, then $(\tilde{x}_k, \bar{y}_{k-1})$ is a feasible point for $\mathcal{P}$ yielding an upper bound on the optimal objective-value for $\mathcal{P}$. Consequently, in case:

$$V^{\mathrm{UB}} > C_x^T \tilde{x}_k + C_y^T \bar{y}_{k-1}$$

the upper bound, V^{UB}, and the incumbent solution, I, can be updated as follows:

$$\begin{array}{ll} V^{\mathrm{UB}} & := C_x^T \tilde{x}_k + C_y^T \bar{y}_{k-1} \\ I & := (\tilde{x}_k, \bar{y}_{k-1}) \end{array}$$

However, frequently, $\mathcal{P}_1$ is infeasible. In that case, $\tilde{x}_k$ is extracted from the solution of an NLP of the following form:

$$\begin{array}{rcll} (\tilde{x}_k, \tilde{s}_k) & := & \underset{x,s}{\operatorname{argmin}} & w^T s \\ & & \text{s.t.} & g(x, \bar{y}_{k-1}) \leq s \\ & & & s \geq 0 \\ & & & x \in \mathbb{R}^{n_x}, \ s \in \mathbb{R}^{n_g} \end{array} \tag{$\mathcal{P}_2$}$$

---

[7] Any MIP can be easily reformulated so to have a linear objective. For instance, using an epigraph reformulation.

where, s is a vector of "slack" variables and w is a vector of non-negative weights. The idea behind $\mathcal{P}_2$ is to find a $\tilde{x}_k$ that, in some sense, minimizes the constraint-infeasibility of $(\tilde{x}_k, \bar{y}_{k-1})$.

Whether $\mathcal{P}_1$ resulted to be feasible or not, the constraint function g is linearized via a first order Taylor expansion around $(\tilde{x}_k, \bar{y}_{k-1})$ and the resulting linear approximations are added to the current set of linear approximations as follows:

$$\begin{gathered}L_{x,k} := \begin{bmatrix} L_{x,k-1} \\ \dfrac{\partial g}{\partial x}(\tilde{x}_k, \bar{y}_{k-1}) \end{bmatrix} \qquad L_{y,k} := \begin{bmatrix} L_{y,k-1} \\ \dfrac{\partial g}{\partial y}(\tilde{x}_k, \bar{y}_{k-1}) \end{bmatrix} \\ l_k := \begin{bmatrix} l_{k-1} \\ g(\tilde{x}_k, \bar{y}_{k-1}) - \dfrac{\partial g}{\partial x, y}(\tilde{x}_k, \bar{y}_{k-1}) \begin{bmatrix} \tilde{x}_k \\ \bar{y}_{k-1} \end{bmatrix} \end{bmatrix}\end{gathered} \tag{2.10}$$

Next, a new integer-feasible guess is found by solving the following MILP:

$$\begin{aligned} (\bar{x}_k, \bar{y}_k) := \quad & \underset{x,y}{\text{argmin}} \quad C_x^T x + C_y^T y \\ & \text{s.t.} \quad L_{x,k} x + L_{y,k} y + l_k \leq 0 \\ & \qquad\;\; x \in \mathbb{R}^{n_x},\ y \in \mathbb{Z}^{n_y} \end{aligned} \tag{$\mathcal{P}_3$}$$

Thanks to the convexity assumption **(A-2)**, at each iteration k, the linear function:

$$g(\tilde{x}_k, \bar{y}_{k-1}) + \frac{\partial g}{\partial x, y}(\tilde{x}_k, \bar{y}_{k-1}) \begin{bmatrix} x - \tilde{x}_k \\ y - \bar{y}_{k-1} \end{bmatrix}$$

under-estimates $g(x, y)$. Therefore, the feasible set defined from $L_{x,k}x + L_{y,k}y + l_k \leq 0$ contains the feasible set defined from $g(x, y) \leq 0$. Consequently, the optimal value of $\mathcal{P}_3$ yields a lower bound on the optimal value of $\mathcal{P}$. Then, V^{LB} can be updated as follows:

$$V^{\text{LB}} := \max\{C_x^T \bar{x}_k + C_y^T \bar{y}_k, V^{\text{LB}}\}$$

Now, if the following condition is met[8]:

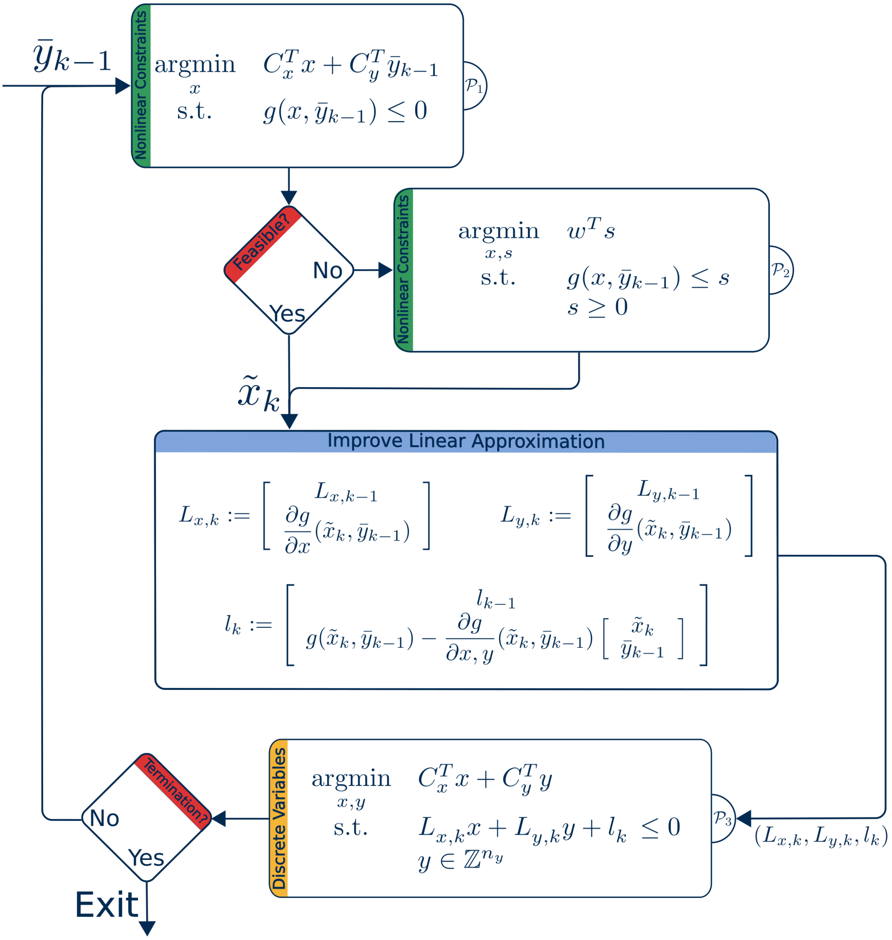
$$V^{\text{UB}} - V^{\text{LB}} \leq \epsilon_{\text{abs}} \quad \text{or} \quad \frac{V^{\text{UB}} - V^{\text{LB}}}{\delta + |V^{\text{UB}}|} \leq \epsilon_{\text{rel}} \ : \ \delta > 0$$

The algorithm stops as I is guaranteed to be a globally optimal point (up to the required tolerance) for $\mathcal{P}$. Otherwise, OA proceeds with the next iteration.

---

[8] According to the definition of the relative optimality gap used in Cplex ([90])

Figure 2.3: Schematic Representation of OA


$\bar{y}_{k-1}$
Nonlinear Constraints
$\operatorname{argmin}_x \quad C_x^T x + C_y^T \bar{y}_{k-1}$
$\text{s.t.} \quad g(x, \bar{y}_{k-1}) \leq 0$
$\mathcal{P}_1$
Feasible?
No
Yes
Nonlinear Constraints
$\operatorname{argmin}_{x,s} \quad w^T s$
$\text{s.t.} \quad g(x, \bar{y}_{k-1}) \leq s$
$s \geq 0$
$\mathcal{P}_2$
$\tilde{x}_k$
Improve Linear Approximation
$L_{x,k} := \begin{bmatrix} L_{x,k-1} \\ \frac{\partial g}{\partial x}(\tilde{x}_k, \bar{y}_{k-1}) \end{bmatrix}$
$L_{y,k} := \begin{bmatrix} L_{y,k-1} \\ \frac{\partial g}{\partial y}(\tilde{x}_k, \bar{y}_{k-1}) \end{bmatrix}$
$l_k := \begin{bmatrix} l_{k-1} \\ g(\tilde{x}_k, \bar{y}_{k-1}) - \frac{\partial g}{\partial x, y}(\tilde{x}_k, \bar{y}_{k-1}) \begin{bmatrix} \tilde{x}_k \\ \bar{y}_{k-1} \end{bmatrix} \end{bmatrix}$
Discrete Variables
$\operatorname{argmin}_{x,y} \quad C_x^T x + C_y^T y$
$\text{s.t.} \quad L_{x,k} x + L_{y,k} y + l_k \leq 0$
$y \in \mathbb{Z}^{n_y}$
$\mathcal{P}_3$
$(L_{x,k}, L_{y,k}, l_k)$
Termination?
No
Yes
Exit

2.7 Mixed-Integer Non-Convex Optimization

Mixed-integer problems that do not satisfy the convexity assumption pose a largely greater challenge than the convex ones. This is due to the fact that not being able to easily estimate the lowest objective value obtainable by selecting a specific discrete assignment makes informed searches in the assignment space of a problem very hard to achieve. Moreover, applying convex algorithms to non-convex mixed-integer problems does not only entail the possibility of missing the globally optimal solution, as in the case of continuous optimization, but also the possibility of not finding any feasible point at all.

Generally, solution methods with guarantees of feasibility and global optimality for mixed-integer non-convex problems are extremely computationally expensive. For example, one such method can be obtained by using the algorithm named Spatial Branch&Bound (SB&B). SB&B differs from B&B in the fact that it branches on the domain of the continuous variables of the problem as well as the discrete ones. This causes the search-tree of SB&B to grow exponentially many times larger than the one of B&B, for a given problem size. Additionally, the construction of even a single SB&B-subproblem may be quite cumbersome as it requires the identification of a convex relaxation of the non-convex constraints of the problem.

In conclusion, mixed-integer non-convex formulations are, in general, ill suited for optimal control. Consequently, this dissertation will consider only mixed-integer non-convex optimal control problems having a very specific structure.

2.8 Summary and Final Comments

In this chapter, the reader has been introduced to the concepts of Mixed-Integer Optimal Control, Model Predictive Control and Mixed-Integer Optimization. In the reminder of the present dissertation, these topics will often recur, and familiarity with the discussed methodologies will be of help for the understanding of the presented work.

Chapter 3

Mixed-Integer Convex Optimization: Proximal-Outer Approximation

In the last decades, several approaches for tackling Mixed-Integer Convex Problems (MICPs) have been developed. Some of such approaches, usually defined as exact methods, aim at finding a globally optimal solution for a given MICP at the expense of a long execution time, while others, generally defined as heuristics, aim at discovering suboptimal feasible points in the shortest time possible. Among the various proposed paradigms, Outer Approximation (OA) and Feasibility Pump (FP), respectively as exact method and as heuristic, deserve a special mention for the number of relevant publications and successful implementations related to them.

In this chapter we present an original exact method called Proximal Outer Approximation (POA). POA blends the fundamental ideas behind FP into the general OA scheme in order to attempt at yielding faster and more robust convergence with respect to OA while retaining the good performances of FP in terms of rapid retrieval of feasible solutions.

The remainder of this chapter is organized in six sections. The first section discusses the considerations that led to the development of POA. In the second section we introduce the class of problems that the proposed algorithm solves. The third section provides the reader with a brief introduction to FP. In the fourth section, our proposed algorithmic approach is described. Finally, in the last two sections, the supporting computational results are shown and some conclusions drawn.

3.1 Motivation

The practical solvability of a MICP instance often depends, among other factors, on how rapidly the used solution algorithm is capable of identifying feasible points for it. For instance, OA derives its convergence from the construction and the iterative refinement of a linear approximation of the non-linear constraint set of the MICP to solve. The refinements are obtained by adding to the current linear approximation new linearizations of the original constraint-set. Using feasible points it is possible to obtain stronger linearizations ([54]), therefore, better refinements and quicker convergence. Consequently, it is common in practice to deploy heuristic methods to rapidly retrieve feasible points in the early stages of MICPs optimizations ([37],[11],[24]). For example, in [9] the authors propose to prepend to the OA iterations some iterations of the widely used heuristics for MICPs named Feasibility Pump (FP) ([19],[37],[1]). Furthermore, in this setting, it is also important to deploy heuristics that are capable of rapidly retrieve close-to-optimal points for a given MICP ([79],[36]). In fact, a close-to-optimal feasible point for a problem provides a tight upper bound on the globally optimal objective-value of the problem and, in turn, such upper bound can be used to speed-up the convergence. In this direction goes the concept of objective-FP ([37],[1],[79]) from which the work to be presented in this chapter takes some inspiration.

Differently from the algorithm presented in [9], our solution scheme follows an integrated approach. The underlying idea is to incorporate some of the fundamental concepts of FP into the general OA scheme. Specifically, the NLP step of POA closely resembles the one of FP while its MILP step is an adaptive trade-off between the more feasibility focused MILP step of FP and the more optimality focused MILP step of OA. Such operation results into an optimization scheme that breaks through the distinction between main scheme and supporting heuristic yielding fast convergence properties while retaining good performances in terms of rapid retrieval of feasible solutions. It is important to note that the presented work does not stand isolated in the literature. For instance, in [49] the concept of inner-approximation for separable MICPs is introduced in order to ease the task of providing feasible integer guesses and, analogously, the algorithm presented in [55] selects integer guesses close to the center of the current linear outer approximation.

3.2 Problem Definition

Consider problem $\mathcal{P}_{\text{MI}}$:

$$\begin{aligned}(x^*, y^*) = \quad & \underset{x,y}{\text{argmin}} \quad && c(x,y) \\ & \text{s.t.} && g(x,y) \leq 0 \\ & && y \in \mathbb{Z}^{n_y}\end{aligned} \tag{$\mathcal{P}_{\text{MI}}$}$$

and assume, possibly performing an epigraph reformulation, its objective to be linear, i.e.:

$$c(x,y) := C_x^T x + C_y^T y$$

Further, let us assume that $\mathcal{P}_{\text{MI}}$ satisfies the following conditions:

- **(A-1)** The constraints set $g(x,y) \leq 0$ implies the box constraints: $l_y \leq y \leq u_y$ where $\|l_y\|_\infty, \|u_y\|_\infty < \infty$.
- **(A-2)** The function g is continuously differentiable and jointly convex in x and y.
- **(A-3)** A constraint qualification, such to guarantee the satisfaction of the first order KKT conditions, holds at the solution of every NLP obtained from $\mathcal{P}_{\text{MI}}$ by fixing the value of the variables y to any discrete assignment within the box constraints.

3.3 Preliminaries: The Non-Linear Feasibility Pump

Similarly to OA, the non-linear FP generates an alternating sequence of integer-feasible and constraint-feasible guesses for (x, y). Differently from OA, however, the FP scheme guarantees that the two sequences will converge to a feasible assignment for (x, y), disregarding optimality.

FP starts by computing $(\tilde{x}_0, \tilde{y}_0)$ as the solution of the continuous relaxation of $\mathcal{P}_{\text{MI}}$. Next, an initial linear approximation for the non-linear constraints set g is obtained as in (2.9). Then, at each iteration, a new integer-feasible guess for (x, y) is generated by solving the following MILP:

$$\begin{aligned}(\bar{x}_k, \bar{y}_k) := \quad & \underset{x,y}{\text{argmin}} \quad && \|y - \tilde{y}_{k-1}\|_1 \\ & \text{s.t.} && L_{x,k} x + L_{y,k} y + l_k \leq 0 \\ & && x \in \mathbb{R}^{n_x}, \ y \in \mathbb{Z}^{n_y}\end{aligned} \tag{$\mathcal{P}_4$}$$

Next, a constraint-feasible guess for (x, y) is found via the solution of the following NLP:

$$\begin{aligned} (\tilde{x}_k, \tilde{y}_k) := \quad \underset{x,y}{\operatorname{argmin}} \quad & \tfrac{1}{2} \left\| y - \bar{y}_k \right\|_2^2 \\ \text{s.t.} \quad & g(x, y) \leq 0 \\ & x \in \mathbb{R}^{n_x},\ y \in \mathbb{R}^{n_y} \end{aligned} \tag{$\mathcal{P}_5$}$$

If $\bar{y}_k \in Y$ then $\tilde{y}_k = \bar{y}_k$ and the point $(\tilde{x}_k, \tilde{y}_k)$ is feasible for $\mathcal{P}_{\text{MI}}$, therefore, the algorithm stops. Otherwise, a new set of linear approximations, is obtained as follows:

$$\begin{gathered} L_{x,k+1} := \begin{bmatrix} L_{x,k} \\ \dfrac{\partial g}{\partial x}(\tilde{x}_k, \tilde{y}_k) \end{bmatrix} \qquad L_{y,k+1} := \begin{bmatrix} L_{y,k} \\ \dfrac{\partial g}{\partial y}(\tilde{x}_k, \tilde{y}_k) \end{bmatrix} \\ l_{k+1} := \begin{bmatrix} l_k \\ g(\tilde{x}_k, \tilde{y}_k) - \dfrac{\partial g}{\partial x, y}(\tilde{x}_k, \tilde{y}_k) \begin{bmatrix} \tilde{x}_k \\ \tilde{y}_k \end{bmatrix} \end{bmatrix} \end{gathered} \tag{3.1}$$

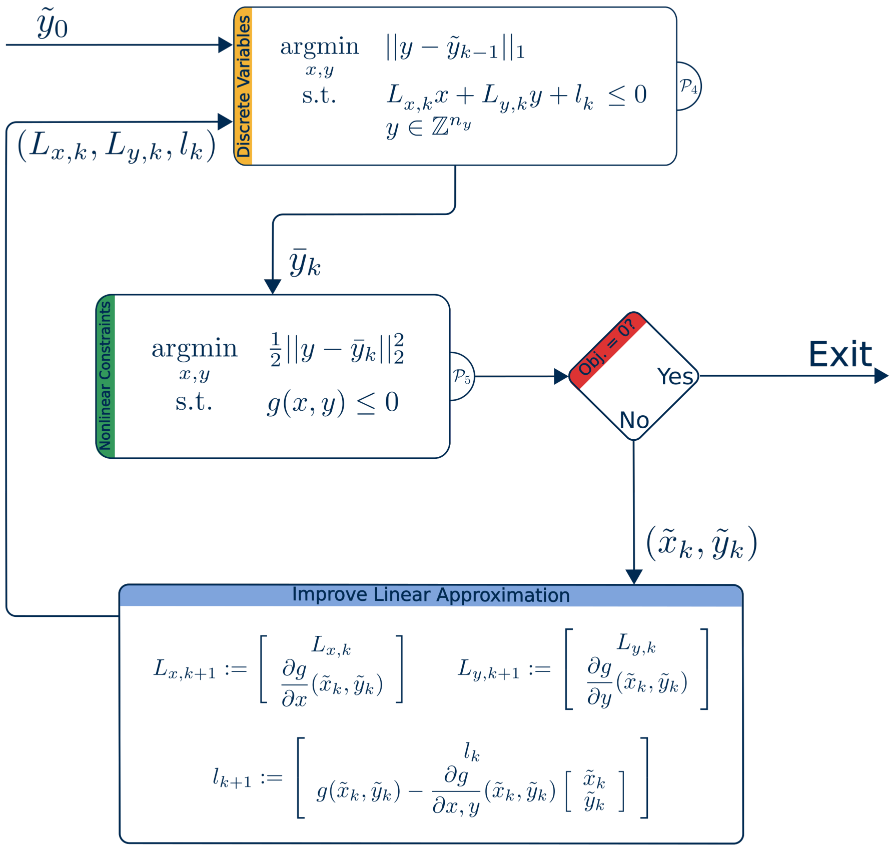
and a new iteration starts.

Note that $\mathcal{P}_4$ can be interpreted as an advanced rounding technique that takes into account the current linear approximation of the constraint set of $\mathcal{P}_{\text{MI}}$. Therefore, it is absolutely feasible to substitute the rounding step at the start-up of OA with few iterations of FP as for example in [9].

$\mathcal{P}_4$ can also be interpreted as an operator that projects the current constraint-feasible solution into the intersection between the domain of the discrete variables and the feasible set defined by the current set of approximating linear constraints. Similarly, $\mathcal{P}_5$ can be interpreted as projecting the current integer-feasible solution into the feasible set of $\mathcal{P}_{\text{MI}}$. This has led some researchers to interpret FP as an "alternating direction method" ([42],[30]).

FP has proven to be a very effective primal heuristic for MICPs but the feasible points it finds are usually of limited quality in terms of optimality. This is due to the fact that the original objective-function is considered exclusively during the generation of the first discrete assignment.

Figure 3.1: Schematic Representation of Non-Linear Feasibility Pump


$\tilde{y}_0$
Discrete Variables
$\underset{x,y}{\text{argmin}} \quad ||y - \tilde{y}_{k-1}||_1$
$\text{s.t.} \quad L_{x,k}x + L_{y,k}y + l_k \leq 0$
$y \in \mathbb{Z}^{n_y}$
$\mathcal{P}_4$
$(L_{x,k}, L_{y,k}, l_k)$
$\bar{y}_k$
Nonlinear Constraints
$\underset{x,y}{\text{argmin}} \quad \frac{1}{2}||y - \bar{y}_k||_2^2$
$\text{s.t.} \quad g(x,y) \leq 0$
$\mathcal{P}_5$
Obj. = 0?
Yes
No
Exit
$(\tilde{x}_k, \tilde{y}_k)$
Improve Linear Approximation
$L_{x,k+1} := \begin{bmatrix} L_{x,k} \\ \frac{\partial g}{\partial x}(\tilde{x}_k, \tilde{y}_k) \end{bmatrix}$
$L_{y,k+1} := \begin{bmatrix} L_{y,k} \\ \frac{\partial g}{\partial y}(\tilde{x}_k, \tilde{y}_k) \end{bmatrix}$
$l_{k+1} := \begin{bmatrix} l_k \\ g(\tilde{x}_k, \tilde{y}_k) - \frac{\partial g}{\partial x, y}(\tilde{x}_k, \tilde{y}_k) \begin{bmatrix} \tilde{x}_k \\ \tilde{y}_k \end{bmatrix} \end{bmatrix}$

3.4 Proximal Outer Approximation

3.4.1 Structure of POA

Just like OA, POA starts by computing $(\tilde{x}_0, \tilde{y}_0)$ as the solution of the continuous relaxation of $\mathcal{P}_{\text{MI}}$, and it obtains a guess $\bar{y}_0$ by rounding $\tilde{y}_0$. Furthermore, POA defines the same initial set of linear approximations that OA defines:

$$\begin{array}{rcl} L_{x,0} & := & \frac{\partial g}{\partial x}(\tilde{x}_0, \tilde{y}_0) \\ L_{y,0} & := & \frac{\partial g}{\partial y}(\tilde{x}_0, \tilde{y}_0) \\ l_0 & := & g(\tilde{x}_0, \tilde{y}_0) - \dfrac{\partial g}{\partial x, y} g(\tilde{x}_0, \tilde{y}_0) \left[\begin{array}{c} \tilde{x}_0 \\ \tilde{y}_0 \end{array} \right] \end{array} \tag{3.2}$$

Then, the following bounds are set:

- A lower bound on the optimal value for $\mathcal{P}_{\mathrm{MI}}$:

$$V^{\mathrm{LB}} := C_x^T \tilde{x}_0 + C_y^T \tilde{y}_0$$

- An upper bound on the optimal value for $\mathcal{P}_{\mathrm{MI}}$:

$$V^{\mathrm{UB}} := \infty$$

Next, a constraint feasible guess for (x, y) is obtained by solving the following FP-like NLP:

$$\begin{array}{rcl} (\tilde{x}_k, \tilde{y}_k) := & \underset{x,y}{\operatorname{argmin}} & \frac{1}{2} \left\| y - \bar{y}_{k-1} \right\|_2^2 \\ & \text{s.t.} & C_x^T x + C_y^T y \leq V^{\mathrm{UB}} - \epsilon_{\mathrm{abs}} \\ & & g(x, y) \leq 0 \\ & & x \in \mathbb{R}^{n_x}, \ y \in \mathbb{R}^{n_y} \end{array} \tag{$\mathcal{P}_6$}$$

where ϵ_{abs} it is the user provided absolute optimality tolerance[1].

Now, we can distinguish two cases:

1. The optimal objective of $\mathcal{P}_6$ is greater than zero or, equivalently, $\tilde{y}_k \neq \bar{y}_{k-1}$. Then, $\bar{y}_{k-1}$ is infeasible or, in other words, there exists no point $(x, \bar{y}_{k-1})$ simultaneously satisfying $g(x, \bar{y}_{k-1}) \leq 0$ and $C_x^T \tilde{x} + C_y^T \bar{y}_{k-1} \leq V^{\mathrm{UB}} - \epsilon_{\mathrm{abs}}$.

---

[1] It is common practice, while solving MICPs, to define positive absolute and relative optimality gap tolerances. Such quantities are used to stop the algorithm whenever the desired solution accuracy is met. Moreover, setting the optimality gap tolerances to zero in a OA-like algorithm might lead to numerical difficulties as, normally, the NLP solvers can guarantee constraint satisfaction and optimality only up to a certain tolerance.

In POA, ϵ_{abs} is used in the MILP and NLP formulation to implement a strict inequality on the objective-value. As a positive side effect, the algorithm is forced to ignore all the feasible assignments that have no chance of improving the current best objective of at least the defined absolute tolerance.

2. The optimal objective of $\mathcal{P}_6$ is equal to zero or, equivalently, $\tilde{y}_k = \bar{y}_{k-1}$. Then, $(\tilde{x}_k, \tilde{y}_k)$ is a feasible point for $\mathcal{P}_{\text{MI}}$ yielding a better objective-value than the one yielded by the incumbent solution.

In the second case, POA attempts to refine $\tilde{x}_k$ via the solution of problem $\mathcal{P}_1$, just like OA (with the difference that, at this point, the feasibility of $\mathcal{P}_1$ is guaranteed). Consequently, V^{UB} and the incumbent solution is updated as follows:

$$\begin{aligned} V^{\text{UB}} &:= C_x^T \tilde{x}_k + C_y^T \tilde{y}_k \\ I &:= (\tilde{x}_k, \tilde{y}_k) \end{aligned} \tag{3.3}$$

Then, a new set of linearizations is defined as follows:

$$\begin{aligned} L_{x,k} &:= \begin{bmatrix} L_{x,k-1} \\ \frac{\partial g}{\partial x}(\tilde{x}_k, \tilde{y}_k) \end{bmatrix} \qquad L_{y,k} := \begin{bmatrix} L_{y,k-1} \\ \frac{\partial g}{\partial y}(\tilde{x}_k, \tilde{y}_k) \end{bmatrix} \\ l_k &:= \begin{bmatrix} l_{k-1} \\ g(\tilde{x}_k, \tilde{y}_k) - \frac{\partial g}{\partial x, y}(\tilde{x}_k, \tilde{y}_k) \begin{bmatrix} \tilde{x}_k \\ \tilde{y}_k \end{bmatrix} \end{bmatrix} \end{aligned} \tag{3.4}$$

Next, a new integer-feasible guess is generated via the solution of the following MILP:

$$\begin{aligned} (\bar{x}_k, \bar{y}_k) := \quad & \underset{x,y}{\text{argmin}} \quad \tfrac{\alpha_k}{N_{o,k}}(C_x^T x + C_y^T y) + \tfrac{(1-\alpha_k)}{N_{d,k}} \|y - \tilde{y}_k\|_1 \\ & \text{s.t.} \quad C_x^T x + C_y^T y \leq V^{\text{UB}} - \epsilon_{\text{abs}} \\ & \qquad\; L_{x,k} x + L_{y,k} y + l_k \leq 0 \\ & \qquad\; x \in \mathbb{R}^{n_x}, \; y \in \mathbb{Z}^{n_y} \end{aligned} \tag{$\mathcal{P}_7$}$$

where: $N_{o,k}$ and $N_{d,k}$ are normalization parameters and $\alpha_k \in [0, 1]$ is used to determine a convex combination of the two normalized objectives.

If α is set to one and $\mathcal{P}_7$ results feasible, $(\bar{x}_k, \bar{y}_k)$ yields a lower bound for the globally optimal objective-value of $\mathcal{P}_{\text{MI}}$. Therefore, in such case, the lower bound V^{LB} can be updated as follows:

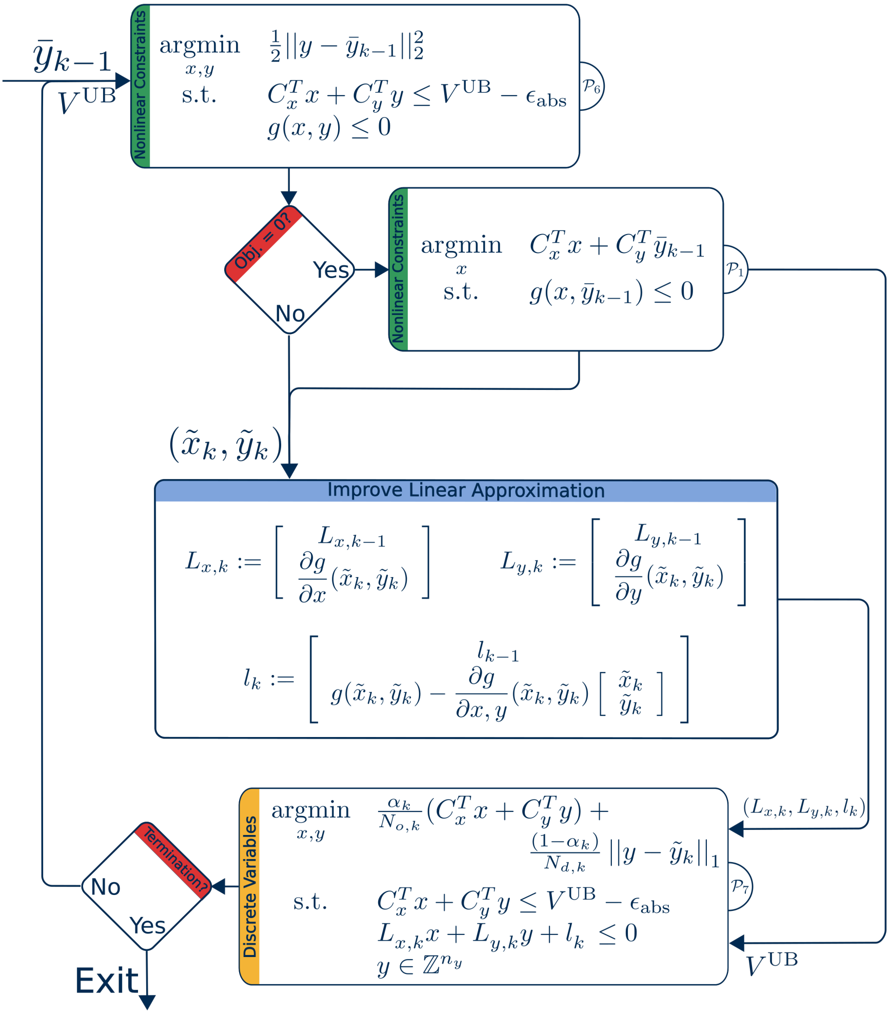
$$V^{\text{LB}} := C_x^T \bar{x}_k + C_y^T \bar{y}_k$$

Eventually, POA stops whenever $\mathcal{P}_7$ the following condition is met:

$$\mathcal{P}_7 \text{ results infeasible} \quad \text{or} \quad V^{\text{UB}} - V^{\text{LB}} \leq \epsilon_{\text{abs}} \quad \text{or} \quad \frac{V^{\text{UB}} - V^{\text{LB}}}{10^{-10} + |V^{\text{UB}}|} \leq \epsilon_{\text{rel}}$$

In any case, if at the time of termination, if an incumbent solution is present, such is an optimal point (up to the defined tolerances) for $\mathcal{P}_{\mathrm{MI}}$, otherwise, $\mathcal{P}_{\mathrm{MI}}$ is proven infeasible. A schematic representation of the POA algorithm can be found in fig. 3.2.

Figure 3.2: Schematic Representation of POA


$\bar{y}_{k-1}$
V^{UB}
Nonlinear Constraints
$\underset{x,y}{\operatorname{argmin}} \quad \frac{1}{2}\|y-\bar{y}_{k-1}\|_2^2$
s.t. $C_x^T x + C_y^T y \leq V^{\mathrm{UB}} - \epsilon_{\mathrm{abs}}$
$g(x,y) \leq 0$
$\mathcal{P}_6$
Obj. = 0?
Yes
No
Nonlinear Constraints
$\underset{x}{\operatorname{argmin}} \quad C_x^T x + C_y^T \bar{y}_{k-1}$
s.t. $g(x, \bar{y}_{k-1}) \leq 0$
$\mathcal{P}_1$
$(\tilde{x}_k, \tilde{y}_k)$
Improve Linear Approximation
$L_{x,k} := \begin{bmatrix} L_{x,k-1} \\ \frac{\partial g}{\partial x}(\tilde{x}_k,\tilde{y}_k) \end{bmatrix}$
$L_{y,k} := \begin{bmatrix} L_{y,k-1} \\ \frac{\partial g}{\partial y}(\tilde{x}_k,\tilde{y}_k) \end{bmatrix}$
$l_k := \begin{bmatrix} l_{k-1} \\ g(\tilde{x}_k,\tilde{y}_k) - \frac{\partial g}{\partial x,y}(\tilde{x}_k,\tilde{y}_k)\begin{bmatrix}\tilde{x}_k \\ \tilde{y}_k\end{bmatrix} \end{bmatrix}$
$(L_{x,k}, L_{y,k}, l_k)$
Discrete Variables
$\underset{x,y}{\operatorname{argmin}} \quad \frac{\alpha_k}{N_{o,k}}(C_x^T x + C_y^T y) + \frac{(1-\alpha_k)}{N_{d,k}}\|y-\tilde{y}_k\|_1$
s.t. $C_x^T x + C_y^T y \leq V^{\mathrm{UB}} - \epsilon_{\mathrm{abs}}$
$L_{x,k}x + L_{y,k}y + l_k \leq 0$
$y \in \mathbb{Z}^{n_y}$
$\mathcal{P}_7$
V^{UB}
Termination?
No
Yes
Exit

It is important to note that the first constraint in $\mathcal{P}_7$ has two functions:

1. It prevents POA from discovering the same integer assignment twice (so that the finite termination of the algorithm can be guaranteed).
2. Forces the MILP to ignore clearly suboptimal integer assignments also in case $\alpha_k < 1$.

Moreover, the convexity assumption **(A-1)** guarantees that the feasible set defined by $L_{x,k}x + L_{y,k}y + l_k \leq 0$ strictly contains the one defined by $g(x, y) \leq 0$. Consequently, the possible infeasibility of $\mathcal{P}_7$ implies the nonexistence of any assignment for (x, y) simultaneously respecting the non-linear constraints of $\mathcal{P}_{\mathrm{MI}}$ and providing an objective-value lower than the current upper bound.

If, instead, $\mathcal{P}_7$ is feasible and $\alpha_k = 1$, then the optimal objective-value of $\mathcal{P}_7$ is a lower bound on the optimal objective-value of $\mathcal{P}_{\mathrm{MI}}$, and:

$$V^{\mathrm{LB}} := C_x^T \bar{x}_k + C_y^T \bar{y}_k \tag{3.5}$$

3.4.2 POA compared to OA

Although the general structure of POA is very similar to the one of OA, the two algorithms differ in two important aspects.

Discrete assignment guessing: OA selects, as guess, always the discrete assignment providing the best objective-value in the MILP stage. Therefore, when the available set of linearizations is a poor approximation of the original non-linear set of constraints, OA tends to generate guesses which are far from the non-linear feasible set and thus scarcely informative. Conversely, POA tries to balance the relaxed objective-value of assignments with their closeness to the non-linear feasible set. Consequently, POA generates guesses which are closer to the non-linear feasible set and therefore more informative in the sense that they generally provide tighter outer approximations.

Linear approximations generation: In case of infeasibility of a discrete assignment guess y, OA solves problem $\mathcal{P}_2$ in order to find a continuous assignment x such that (x, y) minimizes a certain infeasibility measure. The result of this operation is a linearization point that lies outside of the feasible set of $\mathcal{P}_{\mathrm{MI}}$ and whose location depends heavily on the weighting vector W. As shown in [57], via a numerical experiment, generating linearizations for a non-linear constraint set from outside its feasible set might lead to loose linear approximations. Consequently, OA might occasionally perform more iterations than necessary and a bad scaling of the non-linear constraints set can severely hinder the convergence of the algorithm. Differently, in the same circumstances, POA finds a linearization point which lies right at the boundary of the feasible set of $\mathcal{P}_{\text{MI}}$ via the projection step $\mathcal{P}_6$. Moreover, at the linearization point found by POA all components of $g(x, y) \leq 0$ making y infeasible are active. As a result, independently from the scaling of $g(x, y)$, POA generates tighter linear approximations with respect to OA.

3.4.3 Convergence and Exactness of POA

POA converges in finite time

The bounds defined on y are assumed finite. Therefore, to prove the finite convergence of POA is enough to prove that, denoting with $\bar{y}_k$ the discrete assignment found by the MILP ($\mathcal{P}_7$) at the k-th iteration, namely MILP_k, the following condition holds:

$$\bar{y}_{k+h} \neq \bar{y}_{k-1} \ : \ \forall h \in \mathbb{N}$$

Since no constraint is ever removed from the linearization set, the above condition is guaranteed to hold if the linear approximations generated during the k-th iteration suffice to make any point $(x, \bar{y}_{k-1}) \ : \ x \in \mathbb{R}^{n_x}$ infeasible for MILP_k.

Now, let us distinguish two cases, the case where $\bar{y}_{k-1}$ is feasible, or $\bar{y}_{k-1} \in Y$, and the case where it is not, or $\bar{y}_{k-1} \notin Y$. In both cases, we will prove that, for all x such that $C_x^T x + C_y^T \bar{y}_{k-1} < V^{\text{UB}}$, $(x, \bar{y}_{k-1})$ violates the first order Taylor approximation of at least one of the components of $g(x, y) \leq 0$ which are strictly active at the solution of either $\mathcal{P}_1$ or $\mathcal{P}_6$.

Lemma 1: Let $\bar{y}_{k-1} \in Y$ and let $\tilde{x}_k$ be the solution of $\mathcal{P}_1$ then: for all $x \in \mathbb{R}^{n_x}$ with $C_x^T x < C_x^T \tilde{x}_k$ there exists $i \in \{1, \ldots, n_y\}$ such that the constraint $g_i(x, y)$ is strictly active at $(\tilde{x}_k, \bar{y}_{k-1})$ and it holds:

$$\frac{\partial g_i}{\partial x}(\tilde{x}_k, \bar{y}_{k-1})(x - \tilde{x}_k) > 0 \tag{3.6}$$

Proof: From the KKT conditions applied to $\mathcal{P}_1$, we obtain:

$$\begin{cases} \displaystyle\sum_i \mu_i \frac{\partial g_i}{\partial x}(\tilde{x}_k, \bar{y}_{k-1}) = -C_x^T \\ \mu_i \geq 0 \ \ \forall i \in \{1, \ldots, n_g\} \end{cases} \tag{3.7}$$

Therefore, for $x \in \mathbb{R}^{n_x}$ such that $C_x^T x < C_x^T \tilde{x}_k$, we have:

$$\sum_i \mu_i \frac{\partial g_i}{\partial x}(\tilde{x}_k, \bar{y}_{k-1})(x - \tilde{x}_k) = -C_x^T (x - \tilde{x}_k) > 0$$

Then, the desired result follows directly from the non-negativity of the Lagrangian multipliers. ■

Lemma 2: Let $\bar{y}_{k-1} \notin Y$ and let $(\tilde{x}_k, \tilde{y}_k)$ be the solution of $\mathcal{P}_6$. Then, there exists $i \in \{1, \ldots, n_y\}$ such that the constraint $g_i(x, y)$ is strictly active at $(\tilde{x}_k, \tilde{y}_k)$ and, $\forall x \in \mathbb{R}^{n_x}$, holds:

$$\frac{\partial g_i}{\partial x}(\tilde{x}_k, \tilde{y}_k)(x - \tilde{x}_k) + \frac{\partial g_i}{\partial y}(\tilde{x}_k, \tilde{y}_k)(\bar{y}_{k-1} - \tilde{y}_k) > 0 \tag{3.8}$$

Proof: Denoting as $D(x, y)$ the function $\frac{1}{2}\left\|y - \bar{y}_{k-1}\right\|_2^2$, we get:

$$\begin{cases} \dfrac{\partial D}{\partial x}(\tilde{x}_k, \tilde{y}_k) = 0 \\ \dfrac{\partial D}{\partial y}(\tilde{x}_k, \tilde{y}_k) = (\tilde{y}_k - \bar{y}_{k-1})^T \end{cases}$$

moreover, from the KKT conditions we obtain that:

$$\begin{cases} -\dfrac{\partial D}{\partial x}(\tilde{x}_k, \tilde{y}_k) = \displaystyle\sum_i \mu_i \frac{\partial g_i}{\partial x}(\tilde{x}_k, \tilde{y}_k) \\ -\dfrac{\partial D}{\partial y}(\tilde{x}_k, \tilde{y}_k) = \displaystyle\sum_i \mu_i \frac{\partial g_i}{\partial y}(\tilde{x}_k, \tilde{y}_k) \\ \mu_i \geq 0 \;\; \forall i \in \{1, \ldots, n_g\} \end{cases} \tag{3.9}$$

Therefore, for any assignment $(x, \bar{y}_{k-1})$, with $x \in \mathbb{R}^{n_x}$, we have:

$$\begin{gathered} \sum_i \mu_i \left(\frac{\partial g_i}{\partial x}(\tilde{x}_k, \tilde{y}_k)(x - \tilde{x}_k) + \frac{\partial g_i}{\partial y}(\tilde{x}_k, \tilde{y}_k)(\bar{y}_{k-1} - \tilde{y}_k) \right) = \\ -\left(\frac{\partial D}{\partial x}(\tilde{x}_k, \tilde{y}_k)(x - \tilde{x}_k) + \frac{\partial D}{\partial y}(\tilde{x}_k, \tilde{y}_k)(\bar{y}_{k-1} - \tilde{y}_k) \right) = \\ (\tilde{y}_k - \bar{y}_{k-1})^T (\tilde{y}_k - \bar{y}_{k-1}) > 0 \end{gathered}$$

Finally, the desired result follows from the non-negativity of the Lagrangian multipliers. ■

Now, let i be an index such that $g_i(x,y) \leq 0$ is strictly active at the solution of either $\mathcal{P}_1$ or $\mathcal{P}_6$. Possibly renaming $\bar{y}_{k-1}$ to $\tilde{y}_k$, we have: $g_i(\tilde{x}_k, \tilde{y}_k) = 0$. Therefore, taking the first order Taylor expansion of $g_i(x,y) \leq 0$ around $(\tilde{x}_k, \tilde{y}_k)$, we get the following linear constraint:

$$\frac{\partial g_i}{\partial x}(\tilde{x}_k, \tilde{y}_k)(x - \tilde{x}_k) + \frac{\partial g_i}{\partial y}(\tilde{x}_k, \tilde{y}_k)(y - \tilde{y}_k) \leq 0$$

The two lemmas above prove that for any point $(x, \bar{y}_{k-1}) \; : \; x \in \mathbb{R}^{n_x}$ it exists one index i^* such that $(x, \bar{y}_{k-1})$ violates either the above linear constraint or the upper-bound defined in (3.3). Therefore, thanks to (3.4), the discrete assignment $\bar{y}_{k-1}$ is infeasible for every $\mathrm{MILP}_{k+h} \; : \; h \in \mathbb{N}$. As a consequence, $\bar{y}_{k-1}$ cannot be discovered twice and, thanks to the boundedness of the set Y, this guarantees the finite convergence of POA.

For the sake of completeness, we would like to refer the interested reader to [39] and [54] where similar properties are proven in a similar context using a different approach.

POA converges to an optimal point for the MICP

POA stops whenever MILP_k, for some k, results infeasible, i.e., whenever:

$$\nexists (x,y) \in \mathbb{R}^{n_x} \times \mathbb{Z}^{n_y} \text{ s.t. } \begin{cases} L_{x,k}x + L_{y,k}y + l_k \leq 0 \\ C_x^T x + C_y^T y < V^{\mathrm{UB}} \end{cases}$$

Now, given the convexity of the non-linear constraints set $g(x,y) \leq 0$, we have that the linear constraints $L_{x,k}x + L_{y,k}y + l_k \leq 0$ form a relaxation of $g(x,y) \leq 0$. Then, it follows:

$$\nexists (x,y) \in \mathbb{R}^{n_x} \times \mathbb{Z}^{n_y} \text{ s.t. } \begin{cases} g(x,y) \leq 0 \\ C_x^T x + C_y^T y < V^{\mathrm{UB}} \end{cases}$$

Therefore:

- If $V^{\mathrm{UB}} = \infty$, there is no incumbent solution and the original MICP is infeasible.
- If V^{UB} is finite, the incumbent solution, whose objective-value is V^{UB}, is globally optimal for the original MICP.

3.4.4 Setting the parameters

At each iteration, POA solves a MILP in order to find a new guess on the optimal discrete assignment for $\mathcal{P}_{\text{MI}}$. The objective function in $\mathcal{P}_7$ is a combination of the objective of $\mathcal{P}_{\text{MI}}$ and a L_1 distance term. To each term a different relative importance is given depending on the value of the parameter α_k. The idea behind such structure is to look for new discrete assignments trading off between their potential objective-value and their chances of constraint-feasibility.

In fact, whenever the current linear approximations set is a poor approximation of the original non-linear constraints set, it is beneficial to focus more on constraint-feasibility chances in order to rapidly retrieve good solutions. Contrarily, if the generated linearizations form a good approximation of the nonlinear feasible set, focusing on minimal objective-value may speed up the convergence towards a globally optimal point.

In the next paragraphs, an attempt to put some basis for the development of a general rule for the α_k parameter will be presented.

Normalization Constants

First of all, we need to minimize the impact of the scaling of the objective of $\mathcal{P}_{\text{MI}}$ on its importance in $\mathcal{P}_7$. Therefore, we aim to find a couple of iteration dependent constants, $N_{o,k}$ and $N_{d,k}$, such that the normalized terms $\frac{1}{N_{o,k}}(C_x^T x + C_y^T y)$ and $\frac{1}{N_{d,k}}\left\|y - \tilde{y}_k\right\|_1$ have, approximately, the same magnitude. A classical solution to this problem consists in normalizing the two terms using their values at the extremes of the pareto-optimal curve arising from varying the value of $\alpha_k \in [0, 1]$. Clearly, such approach is impractical as it requires the solution of two additional MILPs, one with $\alpha_k = 1$ and the other with $\alpha_k = 0$, at each iteration. A cheaper approximate method consists in setting:

$$\begin{aligned} N_{o,k} &= C_x^T(\tilde{x}_{k-1} - \tilde{x}_0) + C_y^T(\tilde{y}_{k-1} - \tilde{y}_0) \\ N_{d,k} &= \|\tilde{y}_0 - \tilde{y}_{k-1}\|_1 \end{aligned} \tag{3.10}$$

This approximation performs well in practice and requires almost no additional computational load.

Setting α_k

It is not straightforward to a priori determine a general rule for the α_k values. In this section we will limit ourselves to present some of the expertise on the matter gained during the development phase of the presented work.

First of all, we need to define a measurable quantity to monitor in order to assess the quality of the linear approximations set at each iteration. Let us, for the moment, denote such quantity with q_k and assume that q_k tends to zero as the inaccuracy of the linear constraint set $(L_{x,k}, L_{y,k}, l_k)$ is iteratively reduced. Then, it is necessary to map q_k into a suitable value of α_k. In order to do so, we exploited the following algorithm depending on two parameters (τ, λ):

$$
\begin{aligned}
Q_k &:= \begin{cases} 0 & \text{if } k = 0 \\ (\lambda - 1)Q_{k-1} + \lambda q_k & \text{otherwise} \end{cases} \\
\tilde{\alpha}_k &:= \frac{\tau}{\tau + Q_k} \\
\alpha_k &:= \begin{cases} 1 & \text{if } \tilde{\alpha}_k \geq 0.99 \\ \tilde{\alpha}_k & \text{otherwise} \end{cases}
\end{aligned}
\tag{$\alpha_{\text{alg}}(\tau, \lambda)$}
$$

where Q_k is some sort of integrated measurement or running average with λ as damping parameter and τ is used as scaling factor for Q_k in the soft thresholding function $\frac{1}{1+x}$. It is important to note that the hard thresholding in the last step of $\alpha_{\text{alg}}(\tau, \lambda)$ serves the vital purpose of allowing POA to set α_k to exactly one when q_k approaches zero. In fact, when $\alpha_k = 1$ problem $\mathcal{P}_7$ provides a lower bound on the optimal objective-value for $\mathcal{P}_{\text{MI}}$ and such lower bound can be used to trigger the termination of the algorithm.

Coming back to the definition of q_k, a number of metrics have been tested out. For each of the metrics, several tests have been performed varying the values of the two parameters in $\alpha_{\text{alg}}(\tau, \lambda)$. Each test consisted in solving a number of problem instances taken from MICPLib [93] and storing the yielded performances for each of the considered instances. At the end of the testing phase the most promising approaches resulted to be the following two:

1. L_1 averaged distance along the discrete directions between the solution of $\mathcal{P}_7$ and its projection onto the non-linear feasible set:

$$q_k = \frac{||\bar{y}_{k-1} - \tilde{y}_k||_1}{n_y} \tag{3.11}$$

2. Average violation of the non-linear constraints at the solution of $\mathcal{P}_7$:

$$q_k = \frac{||\max(g(\bar{x}_{k-1}, \bar{y}_{k-1}), 0)||_1}{n_g} \tag{3.12}$$

Consequently, in the next section only the two above metrics will be considered to showcase the possible advantages in using the hereby presented POA algorithm.

3.5 Numerical Evaluation

This section presents a comparison between the performances of four different algorithms applied to a number of problem instances taken from MINLPLib [93]. The four considered algorithms are:

1. OA: a basic Outer Approximation implementation as described in the second chapter of this thesis.
2. FP + OA: the algorithm performs Feasibility Pump iterations until a first feasible solution is found then it continues applying Outer Approximation until finding a solution or exceeding the time limit.
3. $\mathrm{POA}_1(1, 1/3)$: Proximal Outer Approximation where alpha is computed according to algorithm $\alpha_{\mathrm{alg}}(\tau, \lambda)$ with q_k computed as in (3.11) and $(\tau := 1, \lambda := 1/3)$
4. $\mathrm{POA}_2(100, 1/3)$: Proximal Outer Approximation where alpha is computed according to algorithm $\alpha_{\mathrm{alg}}(\tau, \lambda)$ with q_k computed as in (3.12) and $(\tau := 100, \lambda := 1/3)$

The tests hereby presented were performed on a personal computer equipped with a 3.7GHz octa-core processor (AMD Ryzen 7 2700X) and 16 Gb of RAM, selecting some instances from MINLPLib ([93]) according to the following rules:

1. Mixed-integer/mixed-boolean variable set and non-linear constraints set and/or objective.
2. Proven convex instances with known globally optimal objective (found by at least 3 solvers).
3. Not relaxations of other problems in the set (to the authors knowledge).
4. Having a ".nl" representation (14 instances excluded for this reason).

For each problem instance and each algorithm a maximum of 300 seconds of computation was allowed, the primal and the integer tolerances where set to 10^{-6} while the absolute and relative gap tolerances were set to 10^{-4}. In order to obtain a fair comparison all the algorithms share the same base code and general sub-solver settings. All the tested algorithms use Cplex v12.9 ([90]) and Ipopt v3.12.11([83]). In the appendix at the end of this chapter a table can be found showing the computational time and the number iterations each algorithm needs to obtain the global optimal solution for each problem instance.

A problem instance is considered successfully solved by an algorithm if the algorithm runtime does not exceed 303 seconds (+1% of the imposed limit to account for possible non-instantaneous terminations) and the obtained objective-value falls within a relative or absolute gap of 10^{-3} from the known globally optimal objective-value of the instance. In the table, all the unsuccessful runs are denoted with the symbol – in place of the number of iterations and the runtime.[3]

3.5.1 Analysis Via Shifted Geometric Mean

In this subsection, the global performances of the tested algorithms are summarized and compared with the aid of two metrics: number of instances solved and the shifted geometric mean of both the computational times and the numbers of iterations. In the following, $SGM(q,s)$ will indicate the shifted geometric mean of the quantity q given the shift s:

$$SGM(q,s) := \left(\prod_{i=1}^{n_q}(s+q_i)\right)^{1/n_q} - s \ : \ q \in \mathbb{R}^{n_q}, \ s \in [0,\infty)$$

In the coming evaluation, we consider only those problem instances where all the algorithms were successful. The shift used for each quantity was selected in order to limit the impact of very easy problems (solved in less of one second and in one or two iterations) in the final comparison. Table 3.1 summarizes the collected data.

Metric	OA	FP + OA	POA_1(1,1/3)	POA_2(100,1/3)
# of solved instances	210	210	213	212
$SGM(\text{times}, 0.5)$	1.83	1.74	1.66	1.62
$SGM(\#\ \text{iterations}, 1.5)$	6.95	7.46	6.21	5.80

Table 3.1: Absolute performances of the analyzed algorithms (full dataset).

According to the averages presented in Table 3.1, it is possible to compare each algorithm against the classical OA. The result of such comparison can be found in Table 3.2.

Evaluating the complexity of each instance using as reference the performances of the algorithm that performed the best on them (according to the considered

---

[3] The appendix contains also the full timing and number of iterations data relative to Bonmin. The data shows how, in terms of number of iterations, our test code is capable of matching the performances of a state-of-the-art OA implementation. However, the data shows also that Bonmin has a good margin of advantage over our test code in terms of execution times. This is due to the fact our test code was built as a proof of concept and, while our code was implemented in Python, Bonmin enjoys a fairly optimized C implementation.

metric), Table 3.3 shows the comparison obtained considering only the most time-demanding 25% of the instances while Table 3.4 considers only the most iteration-intesive 25% of the instances.

Performances VS OA	FP + OA	$POA_1(1,1/3)$	$POA_2(100,1/3)$
Complexity (time)	-4.66 %	-9.38 %	-11.57 %
Complexity (# iterations)	+7.48 %	-10.65 %	-16.49 %

Table 3.2: Performance comparison versus classical OA (full dataset).

Performances VS OA	FP + OA	$POA_1(1,1/3)$	$POA_2(100,1/3)$
Complexity (time)	-7.18 %	-11.00 %	-8.31 %
Complexity (# iterations)	+10.81 %	-13.64 %	-16.24 %

Table 3.3: Performance comparison versus classical OA (time-demanding instances).

Performances VS OA	FP + OA	$POA_1(1,1/3)$	$POA_2(100,1/3)$
Complexity (time)	-1.90 %	-41.17 %	-42.85 %
Complexity (# iterations)	+2.69 %	-46.35 %	-48.55 %

Table 3.4: Performance comparison versus classical OA (iteration-intensive instances).

On the one hand, Table 3.2 shows how, especially on time-demanding problems, prepending an FP run to the OA iterations is likely to reduce the average time needed to find the solution, explaining the popularity of such approach. On the other hand, Table 3.4 shows that the advantages of FP tend to be nullified over the more iteration-intensive problems. This is probably due to the fact that, for those problems, the effort necessary to find the first feasible solution is overshadowed by the large number of iterations required in order to close the optimality gaps.

Regarding POA, we can see that both the tested versions maintain a good performance edge over the classical algorithm in all analyzed scenarios, with $POA_2(100,1/3)$ being more effective than $POA_1(1,1/3)$ in general but slightly less effective on the more time-intensive instances. The most noteworthy finding comes from Table 3.4. The table shows how the presented algorithm is capable of providing very large computational savings in those problems where, due to the high level of non-linearity in the constraint set, the construction of an accurate enough linear relaxation requires a large number of linearization steps.

3.5.2 Analysis Via Performance Profiles

This subsection offers a different approach on the analysis of the collected data. The technique we will use is called Performance Profiles. A Performance Profiles plot compares the performances of different solvers in terms of probability of solving a given problem instance in within a certain amount of time. In order to handle differences in the complexity of the considered instances, the time limits for each instance are normalized by the minimum time for soulution obtained on such instance among all the tested solvers. More specifically, let S denote the set of solvers to compare and let P denote the set of considered problem instances. Then, for all $s \in S$ and $p \in P$ define $t(s,p)$ as the time needed by solver s to solve problem p. Next, define:

$$t^*(p) := \min_{s \in S} t(s,p) \quad \text{and} \quad r(s,p) := \frac{t(s,p)}{t^*(s)}$$

where $r(s,p)$ can be referred to as "ratio of solver s on problem p". Finally, for each solver, consider the following function:

$$\pi_s(\rho) := \frac{|\{p \in P \text{ s.t. } r(s,p) \leq \rho\}|}{|P|} \; : \; \rho \in [1, \infty]$$

where $|.|$ denotes the cardinality of a set. The function $\pi_s(.)$ is the "performance profile" of solver s and plotting the performace profiles of all solvers one against the other we obtain a Performance Profiles plot.

In our case, Figure 3.3 depicts the Performance Profile plot obtained by considering OA, FP+OA and $\text{POA}_2(100, 1/3)$. The plot shows that OA is in general the solver with the highest probability of being the fastest while POA has a slightly lower mean ratio and it appears more robust in the sense that, for instance, it has a better probability of solving a given instance within 150% of the minimum time obtained for such instance.

Furthering the analysis, it is possible to see that the higher rate of victories of OA is mainly established on the easier problems in the test set. In fact, if we consider only the problems that required more than a fraction of second to be solved, like shown in Figure 3.4, the "probability of victory" advantage of OA tends to be nullified and even reversed while the "robustness" advantage of POA becomes more pronounced. Moreover, in these conditions, the mean ratio of OA is substantially higher than the one of POA.

In conclusion, if on the one hand, this second analysis shows that POA enjoys more robust convergence properties with respect to OA, on the other hand, it suggests that OA may be considered better suited for dealing with the less challenging problems.

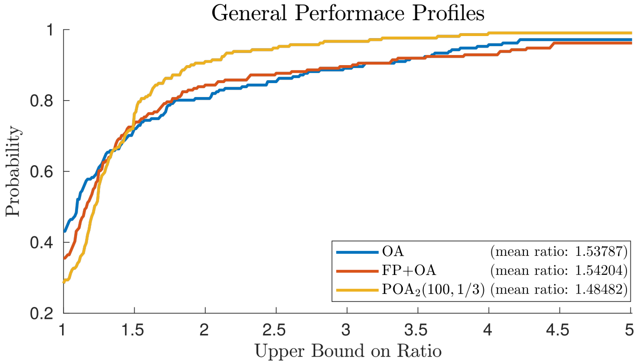


Figure 3.3: Performance Profiles plot obtained considering all instances.

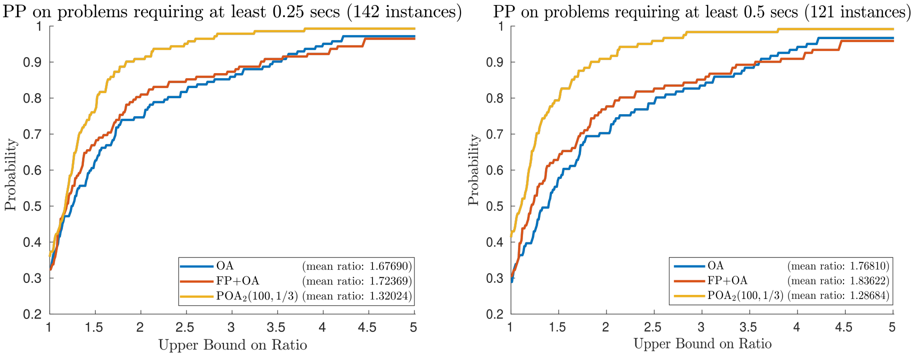


Figure 3.4: Performance Profiles plot obtained considering less trivial instances.

3.6 Summary and Final Comments

In this chapter we presented a new solution algorithm for convex mixed-integer non-linear programming named Proximal Outer Approximation. The proposed algorithm combines ideas and properties of Outer Approximation and Non-Linear Feasibility Pump to construct an integrated iterative solution scheme.

The aim of this work was to create an Outer Approximation scheme with improved generation of approximating linear constraints and better discrete assignment candidates selection. The performed computational experiments have shown the advantages of the new scheme over both the classical Outer Approximation and Outer Approximation warm-started by Feasibility Pump especially on the more complex instances of the test set. Moreover, the comparison between two versions of POA suggested that, through a careful selection of parameters, an additional performance improvement is attainable.

In this chapter, we have found $\text{POA}_2(100,1/3)$ having the best average and peak performances among all the analyzed algorithms. However, this does not exclude the possibility that for different classes of problems different parameter choices could happen to be more efficient.

3.7 Appendix: Table of Results

Examples		Bonmin		OA		OA + FP		POA$_1$		POA$_2$	
Id	Name	#It.	Time	#It.	Time	#It.	Time	#It.	Time	#It.	Time
1	batch	-	-	-	-	-	-	-	-	-	-
2	batch0812	-	-	-	-	-	-	-	-	-	-
3	batchdes	-	-	-	-	-	-	-	-	-	-
4	batchs101006m	-	-	-	-	-	-	-	-	-	-
5	batchs121208m	-	-	-	-	-	-	-	-	-	-
6	batchs151208m	-	-	-	-	-	-	-	-	-	-
7	batchs201210m	-	-	-	-	-	-	-	-	-	-
8	clay0203h	9	0.56	8	0.55	8	0.48	9	0.49	11	0.69
9	clay0203hfsg	8	0.49	6	0.39	7	0.38	6	0.34	6	0.35
10	clay0203m	9	0.35	8	0.33	9	0.35	11	0.5	10	0.51
11	clay0204h	3	0.55	3	0.35	1	0.13	6	0.52	4	0.41
12	clay0204hfsg	3	0.45	3	0.32	4	0.31	4	0.78	4	0.47
13	clay0204m	3	0.15	4	0.24	5	0.22	5	0.26	7	0.41
14	clay0205h	4	5.83	5	2.85	4	2.47	5	3.74	8	6.11
15	clay0205m	6	1.34	6	1.3	9	1.78	9	1.81	8	1.65
16	clay0303h	9	0.95	7	0.69	7	0.77	11	0.67	13	0.81
17	clay0303hfsg	6	0.44	5	0.46	6	0.42	7	0.45	7	0.50
18	clay0303m	10	0.5	8	0.38	13	0.5	9	0.43	11	0.57
19	clay0304h	12	2.32	15	2.84	11	2.03	14	1.97	17	2.12
20	clay0304hfsg	9	1.63	10	1.72	11	1.47	8	1.02	10	1.41
21	clay0304m	18	1.58	17	1.33	21	1.44	16	1.5	21	2.02
22	clay0305h	6	7.62	6	3.32	5	2.76	9	5.87	8	2.68
23	clay0305m	6	1.55	7	1.66	10	2.12	9	2.47	11	2.19
24	cvxnonsep_normcon20	413	9.58	1	0.02	1	0.03	1	0.02	1	0.02
25	cvxnonsep_normcon30	1088	58.21	727	61.95	70	1.49	186	15.24	3	0.08
26	cvxnonsep_normcon40	1876	120.45	-	-	-	-	-	-	3	0.07
27	cvxnonsep_nsig20	309	6.56	266	8.73	266	8.65	65	2.32	66	2.13
28	cvxnonsep_nsig30	-	-	1	0.03	1	0.05	1	0.03	1	0.04
29	cvxnonsep_nsig40	-	-	690	43.27	693	43.69	686	109.77	93	3.33
30	cvxnonsep_pcon20	-	-	56	1.91	59	2.24	30	0.87	26	0.76
31	cvxnonsep_pcon30	-	-	142	7.55	158	7.23	77	2.9	65	2.19
32	cvxnonsep_pcon40	-	-	317	21.5	292	19.82	167	8.43	160	6.44
33	cvxnonsep_psig20	531	11.12	1	0.02	1	0.02	1	0.02	1	0.02
34	cvxnonsep_psig30	704	40.28	630	31.6	600	31.1	177	11.48	152	5.66
35	cvxnonsep_psig40	-	-	-	-	-	-	-	-	-	-
36	enpro48pb	-	-	-	-	-	-	-	-	-	-
37	enpro56pb	-	-	-	-	-	-	-	-	-	-
38	ex1223	3	0.06	3	0.06	3	0.07	4	0.12	3	0.07
39	ex1223a	1	0.04	1	0.03	1	0.02	1	0.02	1	0.02
40	ex1223b	3	0.05	4	0.1	4	0.09	6	0.16	3	0.10
41	ex4	2	0.11	3	0.14	3	0.14	3	0.18	5	0.30
42	fac1	3	0.07	-	-	-	-	3	0.79	24	1.25
43	fac2	6	1.52	-	-	-	-	5	0.3	49	1.60
44	flay02h	2	0.09	3	0.18	3	0.64	3	0.15	3	0.15
45	flay02m	2	0.06	3	0.56	2	0.57	4	0.15	5	0.21

Examples		Bonmin		OA		OA + FP		POA$_1$		POA$_2$	
Id	Name	#It.	Time	#It.	Time	#It.	Time	#It.	Time	#It.	Time
46	flay03h	8	0.31	9	0.6	8	1.17	9	0.71	9	0.91
47	flay03m	8	0.19	9	0.4	9	0.36	9	1.21	9	1.17
48	flay04h	23	4.69	22	4.36	24	6.12	22	5.75	21	4.86
49	flay04m	20	2.01	21	1.88	22	2.15	25	2.42	24	2.34
50	flay05h	-	-	57	265.25	73	281.65	38	278.95	38	208.16
51	flay05m	-	-	40	32.01	36	35.8	38	45.93	37	40.42
52	fo7	3	5.39	3	9.9	5	13.88	4	10.54	3	9.22
53	fo7_2	2	3.08	2	2.36	3	2.57	2	2.26	2	2.70
54	fo7_ar2_1	2	2.91	3	4.71	4	3.08	4	3.87	3	4.70
55	fo7_ar25_1	5	5.57	6	10.17	6	10.06	6	9.07	6	10.62
56	fo7_ar3_1	3	5.1	4	7.92	4	5.52	4	10.27	4	10.08
57	fo7_ar4_1	2	3.05	3	4.14	5	4.48	3	4.87	3	5.04
58	fo7_ar5_1	1	1.8	2	2.12	1	0.81	2	2.32	2	2.27
59	fo8	2	7.09	3	17.1	4	11.08	2	10.26	2	10.18
60	fo8_ar2_1	2	12.68	4	41.04	4	13.51	4	26.72	4	38.24
61	fo8_ar25_1	3	15.37	4	26.92	7	36.77	4	28.16	4	35.07
62	fo8_ar3_1	1	1.99	2	3.29	4	3.22	2	2.9	3	4.43
63	fo8_ar4_1	1	4.21	2	3.8	3	3.01	2	4.18	2	4.24
64	fo8_ar5_1	1	4.18	2	8.74	4	8.04	3	9.52	3	9.37
65	fo9	-	-	3	45.08	4	53.2	3	46.3	3	45.43
66	fo9_ar3_1	1	4.75	2	3.92	3	6.03	2	7.37	2	5.32
67	fo9_ar4_1	1	3.53	2	7.68	4	11.45	2	10.31	2	9.91
68	fo9_ar5_1	3	23.05	4	24.87	4	26.17	4	40.77	4	40.78
69	jit1	5	0.31	1	0.02	-	-	1	0.02	1	0.02
70	m3	1	0.04	1	0.03	1	0.04	1	0.04	1	0.06
71	m6	2	0.19	2	0.11	4	0.14	2	0.19	2	0.13
72	m7	1	0.26	2	0.18	3	0.22	2	0.23	2	0.26
73	m7_ar2_1	1	0.35	2	0.54	3	0.62	2	0.62	2	0.53
74	m7_ar25_1	1	0.21	2	0.21	3	0.29	2	0.23	2	0.30
75	m7_ar3_1	1	0.52	2	0.58	4	0.37	2	0.63	2	0.60
76	m7_ar4_1	1	0.2	2	0.19	3	0.22	3	0.41	2	0.22
77	m7_ar5_1	1	0.15	2	0.29	4	0.35	2	0.61	2	0.33
78	no7_ar2_1	1	2.31	2	3.41	3	5.72	2	3.14	2	5.55
79	no7_ar25_1	2	6.3	4	16.59	5	21.21	3	10.35	4	17.64
80	no7_ar3_1	4	24.37	5	48.51	4	31.37	5	66.58	5	66.35
81	no7_ar4_1	4	15.81	6	65.98	6	38.25	6	72.78	6	73.61
82	no7_ar5_1	7	27.06	7	72.39	7	57.49	7	67.18	7	67.95
83	o7_2	-	-	6	280.21	5	157.73	5	223.75	6	268.26
84	o7_ar2_1	1	10.17	2	12.3	6	15.21	2	13.02	2	14.92
85	o7_ar25_1	-	-	5	139.07	6	90.03	5	125.5	5	208.92
86	o7_ar5_1	-	-	5	247.14	7	238.31	5	291.85	5	289.30
87	portfol_buyin	7	0.09	6	0.11	6	0.1	5	0.12	5	0.13
88	portfol_card	4	0.16	5	0.11	6	0.12	6	0.16	5	0.13
89	ravempb	-	-	-	-	-	-	-	-	-	-
90	risk2bpb	1	0.15	2	0.29	2	0.2	2	0.41	2	0.44

Examples		Bonmin		OA		OA + FP		POA$_1$		POA$_2$	
Id	Name	#It.	Time	#It.	Time	#It.	Time	#It.	Time	#It.	Time
91	rsyn0805h	1	0.11	2	0.17	3	0.18	3	0.34	2	0.21
92	rsyn0805m	2	0.2	2	0.2	4	0.31	3	0.25	3	0.33
93	rsyn0805m02h	5	0.54	4	0.75	4	0.53	3	0.67	3	0.53
94	rsyn0805m02m	4	1.08	5	6.19	6	1.74	6	2.56	5	1.63
95	rsyn0805m03h	3	0.55	4	1.13	3	0.8	3	0.98	3	0.88
96	rsyn0805m03m	2	1.35	3	1.43	4	1.96	4	1.99	3	1.64
97	rsyn0805m04h	2	0.53	3	1.45	3	1.54	2	1.3	2	1.39
98	rsyn0805m04m	2	1.72	3	1.62	4	1.75	4	2.28	4	2.70
99	rsyn0810h	1	0.11	2	0.16	2	0.18	3	0.32	2	0.18
100	rsyn0810m	2	0.17	2	0.23	3	0.33	3	0.31	3	0.28
101	rsyn0810m02h	3	0.43	3	2.48	3	0.7	4	2.56	4	2.66
102	rsyn0810m02m	4	1.2	4	1.6	5	1.42	4	1.23	4	1.64
103	rsyn0810m03h	3	0.78	3	1.28	4	1.2	3	1.37	4	1.48
104	rsyn0810m03m	3	2.02	4	2.74	5	4.72	4	2.71	4	3.05
105	rsyn0810m04h	3	0.9	3	1.3	4	1.68	4	1.69	3	1.55
106	rsyn0810m04m	4	2.24	4	2.35	3	1.68	5	2.78	4	2.41
107	rsyn0815h	1	0.14	2	0.28	3	0.24	3	0.41	2	0.28
108	rsyn0815m	2	0.21	2	0.23	4	0.36	3	0.33	3	0.42
109	rsyn0815m02h	3	0.44	4	0.94	3	0.74	3	0.86	2	0.63
110	rsyn0815m02m	5	1.26	6	1.78	3	0.79	3	0.92	3	0.93
111	rsyn0815m03h	5	1.12	5	2.2	4	1.36	4	1.98	3	1.64
112	rsyn0815m03m	5	2.38	6	6.7	5	3.28	5	3.97	4	3.47
113	rsyn0815m04h	3	1.04	4	2.47	3	1.7	3	2.06	2	1.71
114	rsyn0815m04m	4	2.91	5	5.05	4	3.59	4	3.22	4	2.96
115	rsyn0820h	3	0.23	3	0.38	3	0.33	5	0.44	5	0.43
116	rsyn0820m	2	0.19	2	0.22	3	0.27	2	0.25	2	0.28
117	rsyn0820m02h	3	0.46	3	0.87	3	0.88	6	1.24	6	1.31
118	rsyn0820m02m	3	0.93	4	1.52	4	1.34	4	1.51	4	1.95
119	rsyn0820m03h	2	0.69	3	1.49	4	2.79	4	1.64	3	1.74
120	rsyn0820m03m	3	2.76	4	3.33	5	3.71	4	3.28	4	3.44
121	rsyn0820m04h	4	1.68	4	3	4	2.95	3	2.39	4	2.45
122	rsyn0820m04m	3	2.83	4	7.3	5	7.53	4	6.33	4	5.90
123	rsyn0830h	4	0.3	4	0.46	4	0.48	5	0.82	4	0.48
124	rsyn0830m	4	0.3	5	0.46	5	0.67	5	0.53	5	0.59
125	rsyn0830m03h	2	0.7	3	2.06	4	3.49	3	2.15	3	2.42
126	rsyn0830m03m	4	1.89	4	3.38	5	6.2	4	3.65	4	4
127	rsyn0830m04h	3	1.44	3	2.95	4	3.29	3	3.03	3	3.19
128	rsyn0830m04m	4	3.76	5	13.77	5	10.37	5	23.54	5	17.70
129	rsyn0840h	2	0.21	3	0.38	4	0.43	3	0.45	3	0.48
130	rsyn0840m	2	0.23	3	0.34	4	0.6	3	0.44	3	0.55
131	rsyn0840m03m	4	1.94	5	4.57	4	2.38	5	4.63	5	4.86
132	rsyn0840m04h	2	1.42	3	3.9	3	5.65	3	4.15	4	4.91
133	rsyn0840m04m	4	3.81	5	17.03	4	7.43	5	10.07	5	10.36
134	smallinvDAXr1b010-011	43	2.88	38	2.7	37	2.59	18	1.18	18	1.30
135	smallinvDAXr1b020-022	44	6.25	46	3.42	42	2.63	15	1.31	32	2.74

Examples		Bonmin		OA		OA + FP		POA$_1$		POA$_2$	
Id	Name	#It.	Time	#It.	Time	#It.	Time	#It.	Time	#It.	Time
136	smallinvDAXr1b050-055	67	6.98	70	3.45	62	3.5	27	1.69	29	2.05
137	smallinvDAXr1b100-110	92	9.16	88	3.78	95	4.3	37	2.05	26	1.21
138	smallinvDAXr1b150-165	103	9.26	101	4.64	119	5.26	23	1.24	28	1.29
139	smallinvDAXr1b200-220	111	24.56	-	-	115	4.67	22	1.21	20	0.89
140	smallinvDAXr2b010-011	43	2.9	35	2.18	37	2.75	18	0.92	20	1.68
141	smallinvDAXr2b020-022	48	5.75	42	3.35	48	2.97	19	1.4	32	2.39
142	smallinvDAXr2b050-055	70	6.29	68	3.63	67	3.72	29	1.99	31	2.11
143	smallinvDAXr2b100-110	92	8.38	93	4.04	103	4.5	37	1.99	20	1.01
144	smallinvDAXr2b150-165	104	10.6	93	4.3	76	3.34	48	2.52	25	1.19
145	smallinvDAXr2b200-220	97	20.44	116	4.79	126	5.73	25	4.33	27	1.29
146	smallinvDAXr3b010-011	43	3.99	38	2.63	40	2.62	18	1.16	20	1.05
147	smallinvDAXr3b020-022	48	5.94	39	2.88	38	2.71	19	1.34	30	2.53
148	smallinvDAXr3b050-055	67	6.69	65	3.77	77	3.88	26	1.95	31	2.15
149	smallinvDAXr3b100-110	87	8.56	92	4.16	87	3.57	36	1.89	29	1.71
150	smallinvDAXr3b150-165	104	10.47	94	4.43	113	5.3	46	2.73	30	1.58
151	smallinvDAXr3b200-220	106	29.22	109	4.92	100	4.78	22	1.55	22	1.44
152	smallinvDAXr4b010-011	43	3.74	39	2.88	40	2.96	19	1.07	20	1.61
153	smallinvDAXr4b020-022	50	7.31	44	3.14	35	2.82	17	1.33	28	2.12
154	smallinvDAXr4b050-055	66	6.54	73	3.79	70	3.72	26	2.51	31	2.21
155	smallinvDAXr4b100-110	99	10.56	92	4.07	97	4.39	37	2.26	29	1.91
156	smallinvDAXr4b150-165	103	11.45	85	3.71	109	5.2	34	1.54	19	1.36
157	smallinvDAXr4b200-220	113	25.51	129	5.82	101	4.19	27	1.59	28	1.38
158	smallinvDAXr5b010-011	41	3.02	37	2.62	39	2.65	17	1.06	20	1.28
159	smallinvDAXr5b020-022	46	4.89	40	2.66	39	2.95	18	1.73	24	1.54
160	smallinvDAXr5b050-055	62	5.86	68	4.22	77	4.3	28	1.81	29	2.01
161	smallinvDAXr5b100-110	82	8.63	99	4.66	82	3.48	41	2.67	25	1.91
162	smallinvDAXr5b150-165	114	12.07	99	4.5	106	4.78	27	1.42	22	1.15
163	smallinvDAXr5b200-220	105	30.35	114	5.76	129	6.07	25	1.23	21	1.37
164	sssd08-04	6	0.51	7	0.65	8	0.51	8	0.54	8	0.66
165	sssd12-05	8	3.18	6	1.78	7	1.37	8	3.62	8	2.52
166	sssd15-04	9	1.73	7	2.01	7	1.35	7	1.23	8	1.63
167	sssd15-06	11	13.71	11	28.94	9	9.69	9	7.82	9	75.55
168	sssd15-08	15	19.71	11	205.1	8	100.55	7	83.58	-	-
169	sssd16-07	16	27.77	13	142.49	10	45.75	10	36.64	9	50.72
170	sssd18-06	11	11.54	10	11.93	10	31.03	9	7.39	8	31.01
171	sssd20-04	10	4.42	9	5.14	9	2.08	8	1.61	8	1.73
172	sssd25-04	9	2.73	6	2.76	13	14.17	8	1.62	9	4.46
173	st_e14	3	0.05	3	0.09	3	0.07	4	0.22	3	0.12
174	syn05h	2	0.04	2	0.04	2	0.04	2	0.06	2	0.05
175	syn05hfsg	2	0.04	2	0.04	2	0.04	2	0.05	2	0.06
176	syn05m	2	0.03	2	0.05	2	0.04	2	0.05	2	0.05
177	syn05m02h	1	0.04	1	0.04	1	0.05	1	0.06	1	0.06
178	syn05m02hfsg	1	0.05	1	0.04	1	0.05	1	0.06	1	0.06
179	syn05m02m	1	0.17	2	0.06	2	0.06	2	0.08	2	0.08
180	syn05m03h	1	0.05	1	0.05	1	0.07	1	0.08	1	0.08

Examples		Bonmin		OA		OA + FP		POA$_1$		POA$_2$	
Id	Name	#It.	Time	#It.	Time	#It.	Time	#It.	Time	#It.	Time
181	syn05m03hfsg	1	0.05	1	0.06	1	0.07	1	0.09	1	0.08
182	syn05m03m	1	0.06	2	0.08	3	0.09	2	0.1	2	0.10
183	syn05m04h	1	0.06	1	0.08	1	0.1	1	0.11	1	0.10
184	syn05m04m	1	0.06	2	0.11	2	0.1	2	0.14	2	0.15
185	syn10h	1	0.03	1	0.04	1	0.05	1	0.05	1	0.05
186	syn10hfsg	1	0.03	1	0.04	1	0.04	1	0.05	1	0.05
187	syn10m	2	0.04	2	0.05	2	0.04	2	0.07	2	0.06
188	syn10m02h	1	0.06	2	0.1	2	0.12	2	0.14	2	0.15
189	syn10m02hfsg	1	0.17	2	0.13	2	0.12	2	0.14	2	0.15
190	syn10m02m	2	0.07	2	0.11	2	0.11	2	0.16	2	0.17
191	syn10m03h	1	0.09	2	0.15	2	0.24	2	0.22	2	0.21
192	syn10m03m	1	0.06	2	0.16	2	0.21	2	0.21	2	0.19
193	syn10m04h	1	0.09	2	0.18	3	0.26	3	0.4	2	0.21
194	syn10m04m	1	0.08	2	0.23	2	0.21	2	0.28	2	0.33
195	syn15h	1	0.05	1	0.06	1	0.08	1	0.08	1	0.09
196	syn15hfsg	1	0.04	1	0.07	1	0.08	1	0.08	1	0.08
197	syn15m	2	0.05	2	0.07	2	0.08	2	0.1	2	0.10
198	syn15m02h	1	0.07	2	0.16	2	0.14	2	0.21	2	0.20
199	syn15m02m	1	0.06	2	0.33	3	0.19	3	0.42	2	0.36
200	syn15m03h	1	0.09	2	0.23	2	0.21	3	0.32	3	0.32
201	syn15m03m	2	0.11	2	0.21	3	0.24	3	0.95	2	0.84
202	syn15m04h	1	0.12	2	0.29	3	0.4	10	0.58	3	0.38
203	syn15m04m	2	0.13	3	0.47	3	0.43	4	0.71	3	0.54
204	syn20h	2	0.08	2	0.1	2	0.12	5	0.3	2	0.11
205	syn20hfsg	2	0.08	2	0.09	2	0.11	5	0.3	2	0.11
206	syn20m	2	0.06	2	0.07	2	0.1	2	0.12	2	0.13
207	syn20m02h	2	0.13	2	0.21	2	0.21	2	0.24	2	0.25
208	syn20m02m	2	0.09	3	0.71	3	0.19	3	0.68	3	0.69
209	syn20m03h	1	0.12	2	0.29	3	0.39	3	0.63	2	0.36
210	syn20m03m	2	0.14	3	0.36	3	0.33	3	0.45	3	0.50
211	syn20m04h	1	0.15	2	0.46	3	0.73	7	0.72	3	0.62
212	syn20m04m	2	0.16	3	0.55	3	0.65	3	0.78	3	0.73
213	syn30h	3	0.11	3	0.19	3	0.17	48	4.18	48	4.21
214	syn30m	3	0.2	4	0.19	4	0.44	4	0.25	4	0.25
215	syn30m02h	3	0.2	3	0.39	3	0.42	6	0.69	6	0.69
216	syn30m03h	3	0.29	3	0.63	9	1.45	6	1.05	5	1.11
217	syn30m04h	3	0.37	3	0.87	5	1.19	3	1.22	7	1.86
218	syn40h	4	0.18	6	0.44	4	0.29	9	0.69	9	0.73
219	syn40m03h	4	0.52	4	1.05	4	1.02	8	1.98	8	2.16
220	synthes1	3	0.1	3	0.04	3	0.04	3	0.06	3	0.06
221	synthes2	3	0.09	3	0.09	3	0.07	3	0.07	3	0.08
222	synthes3	6	0.12	6	0.17	6	0.17	7	0.22	8	0.25
223	tls2	7	0.18	8	0.26	11	0.24	8	0.19	8	0.19
224	tls4	-	-	78	17.91	90	22.38	77	24.86	82	22.48
-	**SGM(time=0.5,iter.=1.5)**	**1.38**	**5.98**	**1.49**	**6.28**	**1.42**	**6.75**	**1.34**	**5.65**	**1.34**	**5.32**

Chapter 4

Mixed-Integer Convex Model Predictive Control: The MIRT-OC Algorithm

Recall from Chapter 2 that, on the one hand, the quality of control actions obtained via MPC increases with the length of the time interval considered at each iteration. On the other hand, uncertainties on the system dynamics and possible disturbances often require the control action to be computed in a on-line and closed-loop fashion. This creates a conflict of goals which is especially challenging in mixed-integer MPC due to the high computational cost of mixed-integer programming. Nevertheless, the specific structure of MPC might help alleviating this conflict. In fact, in MPC, the subproblem to solve at a given iteration is very similar to the one solved in the previous iteration. This allows for the development of strategies capable of reducing the computational cost of each iteration by reusing the information generated during the last iterations. In the continuous setting, such techniques are generally referred to as real-time techniques for MPC and a large body of literature on the topic exists ([32],[50],[44],[71]). However, to the author's knowledge far less efforts have been spent on the development of similar strategies in the mixed-integer context ([48],[64]).

The current chapter presents the latest developments regarding the "Mixed-Integer Real Time Optimal Control" (MIRT-OC) algorithm. MIRT-OC is a technique for Convex Mixed-Integer MPC aimed at reducing the cost of each MPC iteration by reusing the information generated during the past iterations. The idea behind MIRT-OC is to extend the concept of single-tree optimization[1], which has found good success in the context of mixed-integer non-linear optimization, to mixed-integer MPC. In essence, MIRT-OC tackles a whole mixed-integer non-linear MPC process as a unique linear/quadratic B&B process where the operations needed to enforce the satisfaction of the non-linear constraints and the transition between one MPC subproblem to the next (MPC-shift) are streamlined directly in B&B.

This new version of MIRT-OC combines the results from two different sources: [28] which introduced the first version of MIRT-OC and [64] where a strategy similar to the one used by MIRT-OC was proposed for the solution of mixed-integer linear-quadratic MPC. The result is an algorithm that can deal with a large class of mixed-integer convex MPC problems as the first version of MIRT-OC, while being robust to model mismatch like the algorithm presented in [64]. Additionally, the algorithm hereby presented is capable of handling general convex tail costs whereas its earlier version made use of more restrictive assumptions. In the following sections, the functioning of MIRT-OC is presented in detail and its performances are demonstrated with the help of a practical case of study involving the control of a hybrid electric vehicle.

4.1 Problem Definition

Let us start from problem $\mathcal{P}''_{\text{DMS}}$:

$$
\begin{aligned}
(x^*, y^*, u^*, v^*) := \quad & \underset{x,y,u,v}{\text{argmin}} \quad \sum_{k=0}^{N-1} \mathbf{s}_k(x_k, y_k, u_k, v_k) \\
& \text{s.t.} \quad x_0 = \bar{x}_0 \\
& \qquad\; y_0 = \bar{y}_0 \\
& \qquad\; \begin{bmatrix} x_{k+1} = \mathbf{d}_k(x_k, y_k, u_k, v_k) \\ y_{k+1} = \mathbf{tr}_k(x_k, y_k, u_k, v_k) \\ \mathbf{p}_k(x_k, y_k, u_k, v_k) \leq 0 \\ (y_k, v_k) \in D_y \times D_v \subset \mathbb{Z}^{n_y + n_v} \end{bmatrix}_{k=0}^{N-1}
\end{aligned}
$$

and assume that:

A-1 The transition functions $\mathbf{tr}_k$ do not depend on the continuous states.

---

[1] A single-tree approach for mixed-integer optimization in a scheme where the optimal solution is found via the exploration of a single search tree. This concept is used in opposition to multi-tree approaches (e.g. OA) where, instead, a number of, possibly smaller, trees are explored. More information on the topic can be found in [6].

A-2 The discretized dynamic constraints, $x_{k+1} = \mathbf{d}_k(x_k, u_k, v_k)$, are linear, i.e. $\mathbf{d}_k(x_k, u_k, v_k) := A_k x_k + B_k \left[{u_k}^T \; {v_k}^T\right]^T$ for some matrices A_k and B_k.

A-3 The path-cost functions $\mathbf{s}_k$ and the path-constraint functions $\mathbf{p}_k$ are convex and continuously differentiable.

A-4 The continuous controls are fully bounded, i.e. the path constraints entail:

$$\forall k : \; \exists \underline{u}_k, \bar{u}_k \text{ s.t. } \underline{u}_k \leq u_k \leq \bar{u}_k \text{ and } \left\|\underline{u}_k\right\|_\infty, \left\|\bar{u}_k\right\|_\infty < \infty$$

A-5 The domain of the discrete controls D_v can be expressed as the intersection of $\mathbb{Z}^{n_v}$ with a rectangular subset of $\mathbb{R}^{n_v}$:

$$D_v := \{v \in \mathbb{Z} \text{ s.t. } \underline{v} \leq v \leq \bar{v}\}$$

Then, defining $w_k := \left[{u_k}^T \; {v_k}^T\right]^T$ and reformulating the discrete states y_k as continuous states (by enlarging the set of discrete controls, as shown in appendix A), it is possible to obtain the following specialized problem formulation:

$$\begin{aligned} (x^*, w^*) := \quad & \underset{x,u,v}{\operatorname{argmin}} \quad \sum_{k=0}^{N-1} \mathbf{s}_k(x_k, w_k) \\ & \text{s.t.} \quad x_0 = \bar{x}_0 \\ & \qquad \begin{bmatrix} A_k x_k + B_k w_k = x_{k+1} \\ \mathbf{p}_k(x_k, w_k) \leq 0 \\ \underline{w}_k \leq w_k \leq \bar{w}_k \\ [w_k]_{k=n_v}^{n_u+n_v} \in \mathbb{Z}^{n_v} \end{bmatrix}_{k=0}^{N-1} \end{aligned} \tag{$\mathcal{P}_{\text{MIRT-OC}}$}$$

Consequently, tackling the resulting control problem in an MPC fashion, we can assume the MPC subproblems to have the following from:

$$\begin{aligned} \min_{x,w} \quad & \sum_{k=j}^{j+h-1} \mathbf{s}_k(x_k, w_k) + \mathbf{f}_{j+h}(x_{j+h}) \\ \text{s.t.} \quad & x_j = \bar{x}_j \\ & \begin{bmatrix} A_k x_k + B_k w_k = x_{k+1} \\ \mathbf{p}_k(x_k, w_k) \leq 0 \\ \underline{w}_k \leq w_k \leq \bar{w}_k \\ [w_k]_{k=n_v}^{n_u+n_v} \in \mathbb{Z}^{n_v} \end{bmatrix}_{k=j}^{j+h-1} \end{aligned} \tag{$\mathcal{P}_{\text{NL}}^{(j)}$}$$

where $h > 1$ is the constant horizon-length of the subproblems.

Introducing a set of slack variables: $\{\eta_k\}_{k=j}^{j+h}$, we can rewrite the problem in such a way that all the non-linearity occurs in the constraint set:

$$\begin{aligned} \min_{x,w,\eta} \quad & \sum_{k=j}^{j+h} \eta_k \\ \text{s.t.} \quad & x_j = \bar{x}_j \\ & \begin{bmatrix} A_k x_k + B_k w_k = x_{k+1} \\ \mathbf{p}_k(x_k, w_k) \leq 0 \\ \mathbf{s}_k(x_k, w_k) - \eta_k \leq 0 \\ \underline{w}_k \leq w_k \leq \bar{w}_k \\ [w_k]_{k=n_v}^{n_u+n_v} \in \mathbb{Z}^{n_v} \end{bmatrix}_{k=j}^{j+h-1} \\ & \mathbf{f}_{j+h}(x_{j+h}) - \eta_{j+h} \leq 0 \end{aligned} \tag{$\mathcal{P}_{\mathrm{NL}}^{(j)}$}$$

Clearly, each of the above problems can be readily solved as a stand-alone problem using any off-the-shelves mixed-integer convex solver. However, given the high degree of similarity between two subsequent subproblems, it is arguably beneficial to reuse some of the information generated during the solution of one subproblem in order to speed-up the solution process for the next.

Before diving into the procedures that MIRT-OC uses to exploit such similarity, it is necessary to have a clearer picture of the functioning of the convex mixed-integer programming algorithm named LP/NLP Branch&Bound.

4.2 LP/NLP Branch&Bound

LP/NLP Branch&Bound (LP/NLP-B&B) was introduced by Quesada and Grossman in 1992 ([70]) as a variation of the OA algorithm where, instead of performing a sequence MILP steps, one single more complex MILP was solved. The following discussion on the structure of LP/NLP-B&B serves the purpose of setting the bases for the presentation of MIRT-OC.

LP/NLP-B&B constitutes the basic building block for MIRT-OC, thus, it is convenient to expose the behaviour of LP/NLP-B&B as the algorithm is applied to $\mathcal{P}_{\mathrm{NL}}^{(j)}$. To this end, consider the following linear relaxation of $\mathcal{P}_{\mathrm{NL}}^{(j)}$:

$$\begin{array}{ll}
\min\limits_{x,w,\eta} & \sum\limits_{k=j}^{j+h} \eta_k \\
\text{s.t.} & x_j = \bar{x}_j \\
& F_{j+h} x_{j+h} - \underline{1}\,\eta_{j+h} \leq f_{j+h} \\
& \left[\begin{array}{l} A_k x_k + B_k w_k = x_{k+1} \\ P_{x,k} x_k + P_{w,k} w_k \leq p_k \\ S_{x,k} x_k + S_{w,k} w_k - \underline{1}\,\eta_k \leq s_k \\ \underline{w}_k \leq w_k \leq \bar{w}_k \\ [w_k]_{k=n_v}^{n_u+n_v} \in \mathbb{Z}^{n_v} \end{array}\right]_{k=j}^{j+h-1}
\end{array} \qquad (\mathcal{P}_{\mathrm{L}}^{(j)})$$

where the nonlinear functions $\mathbf{p}_k$, $\mathbf{s}_k$ and $\mathbf{f}_k$ have been replaced by a set of first-order Taylor approximations for them ($\underline{1}$ denotes the unitary vector of appropriate dimension). LP/NLP-B&B starts by defining $\mathcal{P}_{\mathrm{L}}^{(j)}$ using a single linearization point so to obtain a rough relaxation of $\mathcal{P}_{\mathrm{NL}}^{(j)}$. Then, the algorithm begins solving the obtained relaxation using B&B. Every time a new integer-feasible point for the current definition of $\mathcal{P}_{\mathrm{L}}^{(j)}$ is found, the B&B process is paused and the point is tested against the non-linear constraints of the original problem. If the point satisfies also the non-linear constraints it is marked as the new incumbent solution and removed from the tree, otherwise, the point is reinserted in the tree. Next, an OA-like NLP step is performed in order to obtain a new set of suitable linearizations for the non-linear constraints of $\mathcal{P}_{\mathrm{NL}}^{(j)}$. Finally, $\mathcal{P}_{\mathrm{L}}^{(j)}$ is refined adding to its constraint set the obtained linearizations and the B&B process is resumed. LP/NLP-B&B stops whenever the B&B-tree for the current relaxation is fully explored or a user-defined time limit is exceeded. At this point, the incumbent solution I, if any, is returned as a feasible point for $\mathcal{P}_{\mathrm{NL}}^{(j)}$. Additionally, if the algorithm terminates before hitting the time limit, I is guaranteed to be an globally optimal point for $\mathcal{P}_{\mathrm{NL}}^{(j)}$. A schematic representation of LP/NLP-B&B can be found in Algorithm 3[2].

---

[2] The LP/NLP-B&B algorithm presented here is a slightly modified version of the LP/NLP Branch&Bound algorithm. In the standard version of LP/NLP-B&B, the infeasible and sub-optimal nodes are simply deleted. Contrarily, in the version presented here, such nodes are stored in a separated container named: *Inactive*. This modification does not affect the general structure of the algorithm nor has a significant impact on its performances, however, it is necessary for the development of MIRT-OC.

Algorithm 3: LP/NLP Branch&Bound

```
Input  : Mixed-integer convex problem $\mathcal{P}_{\text{NL}}^{(j)}$, a time limit $\bar{T}$
Output : If $\bar{T}$ large enough, a globally optimal point for $\mathcal{P}_{\text{NL}}^{(j)}$

Generate a rough linear-relaxation $\mathcal{P}_{\text{L}}^{(j)}$ of $\mathcal{P}_{\text{NL}}^{(j)}$ ;
Insert into Queue the continuous relaxation of $\mathcal{P}_{\text{L}}^{(j)}$;
$UpB := \infty$ ;
while Queue is not empty and time $\leq \bar{T}$ do
    Extract the first problem $\hat{\mathcal{P}}$ from Queue;
    Solve $\hat{\mathcal{P}}$ until convergence or until proving $\mathcal{O}_{\text{obj.}}(\hat{\mathcal{P}}) \geq UpB$;
    if $\hat{\mathcal{P}}$ is feasible and $\mathcal{O}_{obj.}(\hat{\mathcal{P}}) < UpB$ then
        if $\mathcal{O}_{sol.}(\hat{\mathcal{P}})$ is integer-feasible then
            if $\mathcal{O}_{sol.}(\hat{\mathcal{P}})$ is feasible also for $\mathcal{P}_{NL}^{(j)}$ then
                $I := \mathcal{O}_{\text{sol.}}(\hat{\mathcal{P}}),\ UpB := \mathcal{O}_{\text{obj.}}(\hat{\mathcal{P}})$;
                Insert $\hat{\mathcal{P}}$ into Inactive;
            else
                Reinsert $\hat{\mathcal{P}}$ into Queue;
            end
            Obtain a suitable linearization of the non-linear constraints of
              $\mathcal{P}_{\text{NL}}^{(j)}$ via an OA-like NLP step;
            Refine $\mathcal{P}_{\text{L}}^{(j)}$ by adding to its constraint set the new linearizations;
        else
            Children := Branch($\hat{\mathcal{P}}$);
            for $\hat{\mathcal{P}}_n \in$ Children($\hat{\mathcal{P}}$) do
                Insert $\hat{\mathcal{P}}_n$ into Queue;
            end
        end
    else
        Insert $\hat{\mathcal{P}}$ into Inactive;
    end
end
```

Similarly to B&B, LP/NLP-B&B computes the optimal objective value of each node in the B&B tree and uses those values to avoid exploring useless portions of the tree. However, differently from B&B, the constraint set considered by LP/NLP-B&B keeps increasing in size during the execution. Clearly, after each constraints insertion, the already computed objective values are not valid anymore. Nevertheless, LP/NLP-B&B can still use those values to safely prune the B&B-tree because the optimal objective value computed for a node before the constraint addition remains lower-bound for the optimal objective value of the node after the constraint addition. This is due to the fact that a constraint addition cannot improve the optimal objective value of an optimization problem.

LP/NLP-B&B is the prototypical OA-based single-tree (one only B&B-tree is explored) approach, whereas the classical OA is a multi-tree approach. In many circumstances, it appears that single-tree approaches have a performance edge over their competitor algorithms. In fact, according to the recent surveys by Kronqvist and al. ([56]), at the state of the art, the most performative solvers for mixed-integer non-linear convex problems use a single-tree approach. Therefore, it can be expected that selecting an LP/NLP-B&B-like approach over an OA one in the context of mixed-integer MPC does not entail a loss in terms of performance.

4.3 The Algorithmic Structure of MIRT-OC

As said, the driving idea behind MIRT-OC is to extend the concept of single-tree non-linear mixed-integer optimization to non-linear mixed-integer MPC. The present section consists of a high-level description of the procedures employed by MIRT-OC to do so.

During the j-th iteration a LP/NLP-B&B process is run on $\mathcal{P}_{\mathrm{NL}}^{(j)}$ until either the algorithm terminates or a given computation time limit is met. Assume that, at the end of the iteration at least one feasible point (x', w') for $\mathcal{P}_{\mathrm{NL}}^{(j)}$ is known. At this point, the control action w'_j is sent to the plant and a measurement on the state is taken. Denoting the measured state with $\tilde{x}_j$, the initial value for the state in $\mathcal{P}_{\mathrm{NL}}^{(j+1)}$ is defined as $\bar{x}_{j+1} := A_j\tilde{x}_j + B_j w'_j$. After that, two subroutines are called in order to transform part of the information collected during the solution of $\mathcal{P}_{\mathrm{NL}}^{(j)}$ into information useful for speeding-up the solution process for $\mathcal{P}_{\mathrm{NL}}^{(j+1)}$ and a new iteration begins.

The first of the two subroutines, “Relaxation-Shift”, transforms the linearly-constrained relaxation built for $\mathcal{P}_{\mathrm{NL}}^{(j)}$ into an initial linearly-constrained relaxation for $\mathcal{P}_{\mathrm{NL}}^{(j+1)}$, namely: $\mathcal{P}_{\mathrm{L}}^{(j+1)}$. The second subroutine, “Tree-Shift”, transforms the B&B-tree built during the solution of $\mathcal{P}_{\text{NL}}^{(j)}$ into a partially explored tree for $\mathcal{P}_{\text{L}}^{(j+1)}$. A concise representation of MIRT-OC can be found in Algorithm 4.

Algorithm 4: MIRT-OC

```
Input  : Mixed-integer convex OCP $\mathcal{P}_{\text{OC}}$,
         A time per iteration limit $\bar{T}$
Output : A control strategy for $\mathcal{P}_{\text{OC}}$
Define $\mathcal{P}_{\text{NL}}^{(0)}$ and generate a rough linear-relaxation $\mathcal{P}_{\text{L}}^{(0)}$ for it;
Insert into $Queue_0^{(0)}$ the continuous relaxation of $\mathcal{P}_{\text{L}}^{(0)}$;
for $j \in \{0, \cdots, M\}$ do
    $\begin{bmatrix} (x', w') \\ \mathcal{P}_{\text{L}}^{(j)} \\ Queue^{(j)} \\ Inactive^{(j)} \end{bmatrix}$ := LP/NLP-B&B($\mathcal{P}_{\text{NL}}^{(j)}, \mathcal{P}_{\text{L}}^{(j)}, Queue_0^{(j)}, \bar{T}$);
    Send $w'_j$ to the plant and receive $\tilde{x}_j$;
    Define $\mathcal{P}_{\text{NL}}^{(j+1)}$ using $\bar{x}_{j+1} := A_j \tilde{x}_j + B_j w'_j$;
    $\mathcal{P}_{\text{L}}^{(i+1)}$ := Relaxation-Shift($\mathcal{P}_{\text{L}}^{(j)}$, $x_{j+1}^{(j)}$);
    $Queue_0^{(j+1)}$ := Tree-Shift($Queue^{(j)}$, $Inactive^{(j)}$, $\mathcal{P}_{\text{L}}^{(j+1)}$, $w_j^{(j)}$);
end
```

4.3.1 Linearization-Shift

The first operation needed is the generation of an initial linearly-constrained relaxation for $\mathcal{P}_{\text{NL}}^{(j+1)}$. Since most of the constraint sets of $\mathcal{P}_{\text{NL}}^{(j)}$ and $\mathcal{P}_{\text{NL}}^{(j+1)}$ coincide, it is natural to consider propagating the linearizations generated for the former to the latter. In fact, thanks to the structure of $\mathcal{P}_{\text{NL}}^{(\cdots)}$ being almost decoupled in time, all the constraints relative to the time-steps from the $(j+1)$-th to the $(j+h-1)$-th can be propagated forward with no additional computation.

However, before proceeding with the construction of the desired linear relaxation, it is necessary to obtain a set of linearizations for the new non-linear constraints:

$$\begin{aligned} &\mathbf{p}_{j+h}(x_{j+h}, w_{j+h}) \leq 0 \\ &\mathbf{s}_{j+h}(x_{j+h}, w_{j+h}) - \eta_{j+h} \leq 0 \\ &\mathbf{f}_{j+h+1}(x_{j+h+1}) - \eta_{j+h+1} \leq 0 \end{aligned} \tag{4.1}$$

For such purpose, assume that the LP/NLP-B&B process for $\mathcal{P}_{\mathrm{NL}}^{(j)}$ has identified L feasible primal points, namely: $\{(x^{(l)}, w^{(l)})\}_{l=1}^{L}$, and conjecture (possibly incorrectly) that the point:

$$
\begin{gathered}
(x_{j+1}^{(l)}, w_{j+1}^{(l)}, \cdots, w_{j+h-1}^{(l)}, x_{j+h}^{(l)}, w_{j+h}^{(l)}, \tilde{x}_{j+h+1}^{(l)}) \\
\text{where: } \tilde{x}_{j+h+1}^{(l)} := A_{j+h} x_{j+h}^{(l)} + B_{j+h} w_{j+h-1}^{(l)}
\end{gathered}
$$

is feasible for $\mathcal{P}_{\mathrm{NL}}^{(j+1)}$. Next, consider linearizing the above constraints around each of the obtained hypothetical feasible points. Then, $\forall l \in \{1, \cdots, L\}$, we obtain the following linearizations:

$$
\begin{aligned}
\tilde{P}_{x,j+h}^{(l)} &:= \frac{\partial \mathbf{p}_{j+h}}{\partial x_{j+h}}(x_{j+h}^{(l)}, w_{j+h-1}^{(l)}) \\
\tilde{P}_{w,j+h}^{(l)} &:= \frac{\partial \mathbf{p}_{j+h}}{\partial w_{j+h}}(x_{j+h}^{(l)}, w_{j+h-1}^{(l)}) \\
\tilde{p}_{j+h}^{(l)} &:= \frac{\partial \mathbf{p}_{j+h}}{\partial \cdot}(x_{j+h}^{(l)}, w_{j+h-1}^{(l)}) \begin{bmatrix} x_{j+h}^{(l)} \\ w_{j+h-1}^{(l)} \end{bmatrix} - \mathbf{p}_{j+h}(x_{j+h}^{(l)}, w_{j+h-1}^{(l)}) \\
\\
\tilde{S}_{x,j+h}^{(l)} &:= \frac{\partial \mathbf{s}_{j+h}}{\partial x_{j+h}}(x_{j+h}^{(l)}, w_{j+h-1}^{(l)}) \\
\tilde{S}_{w,j+h}^{(l)} &:= \frac{\partial \mathbf{s}_{j+h}}{\partial w_{j+h}}(x_{j+h}^{(l)}, w_{j+h-1}^{(l)}) \\
\tilde{s}_{j+h}^{(l)} &:= \frac{\partial \mathbf{s}_{j+h}}{\partial \cdot}(x_{j+h}^{(l)}, w_{j+h-1}^{(l)}) \begin{bmatrix} x_{j+h}^{(l)} \\ w_{j+h-1}^{(l)} \end{bmatrix} - \mathbf{s}_{j+h}(x_{j+h}^{(l)}, w_{j+h-1}^{(l)}) \\
\\
\tilde{F}_{j+h+1}^{(l)} &:= \frac{\partial \mathbf{f}_{j+h+1}}{\partial \cdot}(\tilde{x}_{j+h+1}^{(l)}) \\
\tilde{f}_{j+h+1}^{(l)} &:= \frac{\partial \mathbf{f}_{j+h+1}}{\partial \cdot}(\tilde{x}_{j+h+1}^{(l)})\tilde{x}_{j+h+1}^{(l)} - \mathbf{f}_{j+h+1}(\tilde{x}_{j+h+1}^{(l)})
\end{aligned}
\tag{4.2}
$$

where:

$$
\begin{aligned}
\frac{\partial g}{\partial \cdot} &:= \text{“Jacobian of function } g \text{ with respect all occuring variables”} \\
\frac{\partial g}{\partial y} &:= \text{“Jacobian of function } g \text{ with respect to variable } y\text{”}
\end{aligned}
$$

At this point, it is possible to construct a linearly-constrained relaxation for $\mathcal{P}_{\text{NL}}^{(j+1)}$ by manipulating the current linearization for $\mathcal{P}_{\text{NL}}^{(j)}$ in the following manner:

1. In the objective, subtract the term: η_{j+h}, and add the term: η_{j+h+1}.
2. In the constraint set, replace:

$$\begin{aligned}
&x_j = \bar{x}_j\\
&A_j x_j + B_j w_j = x_{j+1}\\
&\underline{w}_j \leq w_j \leq \bar{w}_j\\
&P_{x,j} x_j + P_{w,j} w_j \leq c_j\\
&S_{x,j} x_j + S_{w,j} w_j - \underline{1}\,\eta_j \leq s_j\\
&M_j w_j \leq m_j
\end{aligned}$$

with:

$$x_{j+1} = \bar{x}_{j+1}$$

3. Finally, define:

$$\begin{aligned}
(P_{x,j+h}, P_{w,j+h}, p_{j+h}) &:= \left(\left[\tilde{P}_{x,j+h}^{(l)}\right]_{l=1}^{L}, \left[\tilde{P}_{w,j+h}^{(l)}\right]_{l=1}^{L}, \left[\tilde{p}_{j+h}^{(l)}\right]_{l=1}^{L}\right)\\
(S_{x,j+h}, S_{w,j+h}, s_{j+h}) &:= \left(\left[\tilde{S}_{x,j+h}^{(l)}\right]_{l=1}^{L}, \left[\tilde{S}_{w,j+h}^{(l)}\right]_{l=1}^{L}, \left[\tilde{s}_{j+h}^{(l)}\right]_{l=1}^{L}\right)\\
(F_{j+h+1}, f_{j+h+1}) &:= \left(\left[\tilde{F}_{j+h+1}^{(l)}\right]_{l=1}^{L}, \left[\tilde{f}_{j+h+1}^{(l)}\right]_{l=1}^{L}\right)
\end{aligned}$$

and replace:

$$F_{j+h} x_{j+h} - \underline{1}\,\eta_{j+h} \leq f_{j+h}$$

with:

$$\begin{aligned}
&A_{j+h} x_{j+h} + B_{j+h} w_{j+h} = x_{j+h+1}\\
&\underline{w}_{j+h} \leq w_{j+h} \leq \bar{w}_{j+h}\\
&P_{x,j+h} x_{j+h} + P_{w,j+h} w_{j+h} \leq p_{j+h}\\
&S_{x,j+h} x_{j+h} + S_{w,j+h} w_{j+h} - \underline{1}\,\eta_{j+h} \leq s_{j+h}\\
&F_{j+h+1} x_{j+h+1} - \underline{1}\,\eta_{j+h+1} \leq f_{j+h+1}
\end{aligned}$$

4.3.2 Tree-Shift

First of all, it is worth to specify what a Tree-Shift actually implies. As observable in Algorithm 3, LP/NLP-B&B simultaneously builds and explores a B&B-tree. However, the tree is never fully stored in memory as all the information useful to the optimization is, instant by instant, stored into the "frontier" nodes of the current B&B-tree[3]. A node belongs to the "frontier" if it belongs to the *Queue* (i.e. the algorithm still has to branch on it) or to the *Inactive* set (i.e. it was found infeasible or suboptimal). As a consequence, in the following, the expression "all the nodes in the B&B-tree" actually refers to the "frontier" nodes only.

Now, since the choice of applying to the plant the control w'_j is irreversible, all the nodes in the B&B-tree for $\mathcal{P}_{\mathrm{L}}^{(j)}$ where the variable w_j is constrained to take some value different form w'_j can be safely deleted. After that, in order to obtain a B&B-tree for $\mathcal{P}_{\mathrm{L}}^{(j+1)}$, the remaining nodes have to be adapted to the new problem formulation. From an implementation point of view, each B&B-node contains all the information necessary to define the subproblem it represent and some auxiliary information. In order to not to complicate the discourse unnecessarily, assume that each node contains only the following information:

- The branching constraints that define the associated subproblem.
- A primal/dual guess for the associated subproblem (for hot-starting).
- A lower-bound on the optimal objective value of the associated subproblem.

Then, in order to adapt each node after the shift, it is sufficient to remove from it the branching constraints relative to the j-th time step, to deduce a new primal/dual guess (it does not need to be feasible) and to set the the lower-bound to $-\infty$. However, for the efficiency of LP/NLP-B&B, it is important that each node is provided with the tightest objective lower bound possible. One obvious way to obtain such bounds would be to re-solve all the transformed B&B-nodes up to some optimality tolerance. Clearly, such approach would be very computationally expensive. Thus, instead, it could be beneficial to find approximate lower bounds using cheaper techniques. The next section explains the procedure used by MIRT-OC to this end.

4.4 Obtaining Lower-Bounds for The Shifted Nodes

The present section discusses the theoretical devices that allow MIRT-OC to compute new approximate objective lower-bounds for the all nodes in the shifted B&B-tree at the cost of solving a small number of linear problems.

---

[3] In fact, as already proven in [28] and in [64], the union of the feasible sets of the subproblems represented by the 'frontier" nodes of a B&B-tree always contains the feasible set of the original problem.

4.4.1 Branch&Bound Sub-Problems

The theoretical results to be presented in this section will make use of the specific structure of the B&B-subproblems induced by the formulation of $\mathcal{P}_{\mathrm{L}}^{(j)}$. Consequently, it is convenient to take a closer look at such structure.

The subproblem stored at each node in the B&B tree for $\mathcal{P}_{\mathrm{L}}^{(j)}$ reads as follows :

$$\begin{aligned}
\min_{x,w,\eta} \quad & \sum_{k=j}^{j+h} \eta_k \\
\text{s.t.} \quad & x_j = \bar{x}_j \\
& \left[\begin{array}{l}
A_k x_k + B_k w_k = x_{k+1} \\
P_{x,k} x_k + P_{w,k} w_k \leq p_k \\
S_{x,k} x_k + S_{w,k} w_k - \underline{1}\,\eta_k \leq s_k \\
\underline{w}_k \leq w_k \leq \bar{w}_k \\
M_k w_k \leq m_k
\end{array}\right]_{k=j}^{j+h-1} \\
& F_{j+h} x_{j+h} - \underline{1}\,\eta_{j+h} \leq f_{j+h}
\end{aligned} \qquad (\mathcal{P}_{\mathrm{BB}}^{(j)})$$

where the branching constraints, $M_k w_k \leq m_k$, have replaced the discrete requirements on the control variables present in $\mathcal{P}_{\mathrm{L}}^{(j)}$.

Additionally, since $\mathcal{P}_{\mathrm{BB}}^{(j)}$ is a linear problem, its dual reads as follows:

$$\begin{aligned}
\max_{\substack{\alpha,\beta,\pi,\\ \sigma,\mu,\phi}} \quad & -\bar{x}_j^T \alpha_j - \bar{w}^T \max\{\beta, 0\} - \underline{w}^T \min\{\beta, 0\} \\
& \qquad -c^T \pi - s^T \sigma - m^T \mu - f_{j+h}^T \phi \\
\text{s.t.} \quad & \\
& \left[\begin{array}{l}
\alpha_k - A_k^T \alpha_{k+1} + P_{x,k}^T \pi_k + S_{x,k}^T \sigma_k = 0 \\
\beta_k - B_k^T \alpha_{k+1} + P_{w,k}^T \pi_k + S_{w,k}^T \sigma_k + M_k^T \mu_k = 0 \\
\underline{1}^T \sigma_k = 1; \quad \pi_k, \sigma_k, \mu_k \geq 0
\end{array}\right]_{k=j}^{j+h-1} \\
& \alpha_{j+h} + F_{j+h}^T \phi = 0 \\
& \underline{1}^T \phi = 1; \quad \phi \geq 0
\end{aligned} \qquad (\mathcal{D}_{\mathrm{BB}}^{(j)})$$

$\mathcal{D}_{\mathrm{BB}}^{(j)}$ is important for two reasons:

1. The objective value of any feasible solution for $\mathcal{D}_{\mathrm{BB}}^{(j)}$ is a lower bound on the optimal objective value of $\mathcal{P}_{\mathrm{BB}}^{(j)}$ (by weak duality).

2. A feasible dual point for a B&B-subproblem is also a feasible dual point for all the descendants of the subproblem.[4]

This two characteristics make of dual problems a powerful instrument for MIRT-OC and in the remainder of this chapter it will be presumed that for each subproblem in the B&B-tree a feasible dual point is available (even in case of primal infeasibility). This assumption is not very restrictive as already most of the B&B implementations to date employ dual algorithms (e.g. dual simplex) for the solution of the B&B-subproblems. In fact, this practice has two advantages: firstly, it allows for early detection of suboptimality, and secondly, it makes possible to hot-start the solution process for a subproblem using the optimal dual solution of its parent.

4.4.2 Computing a New Feasible Dual-Point

Given the availability of dual solutions for the nodes in the B&B-tree the task of computing new objective lower-bounds after an MPC-shift can be naturally carried out leveraging on the dual of the B&B-subproblems. From the last iteration, for each of the nodes in the B&B-tree, we have a dual point:

$$d' := \left(\{\alpha'_k\}_{k=j}^{j+h}, \{\beta'_k\}_{k=j}^{j+h}, \{\pi'_k\}_{k=j}^{j+h-1}, \{\mu'_k\}_{k=j}^{j+h-1}, \{\sigma'_k\}_{k=j}^{j+h-1}, \phi'\right) \tag{4.3}$$

satisfiying the constraint set of $\mathcal{D}_{\mathrm{BB}}^{(j)}$. After the shift, it is necessary to extend d' with a set of multipliers for the new portion of the problem while ensuring the feasibility of the resulting point with respect to the subproblem represented by the shifted node. The strategy that MIRT-OC uses to do so consists in first trivially adapting d', and then, for ensuring feasibility, summing to the result a correction term. Clearly, the correction term has to be tailored to each individual node. However, MIRT-OC obtains each term as a linear combination of a small set of dual points that can be generated upfront. In this way, the computational cost of obtaining the new lower-bounds does not significantly depend on the size of the B&B-tree and essentially equals the cost of solving a small number of linear optimization problems.

First, let us adapt d' by setting the missing multipliers $(\beta_{j+h}, \pi_{j+h}, \cdots)$ to zero and by dropping the obsolete ones $(\alpha'_j, \beta'_j, \cdots)$. The result is a point:

$$d'' := \left(\{\alpha''_k\}_{k=j+1}^{j+h+1}, \{\beta''_k\}_{k=j+1}^{j+h}, \{\pi''_k\}_{k=j+1}^{j+h}, \{\sigma''_k\}_{k=j+1}^{j+h}, \{\mu''_k\}_{k=j+1}^{j+h}, \phi''\right) \tag{4.4}$$

---

[4] This is due to the fact that each non-root node is obtained from its parent via the addition of a branching constraint whose dual multiplier can be assumed to be initially zero.

for which it holds:

$$\begin{aligned}
&\left[\begin{array}{l} \alpha''_k - A_k^T \alpha''_{k+1} + P_{x,k}^T \pi''_k + S_{x,k}^T \sigma''_k = 0 \\ \beta''_k - B_k^T \alpha''_{k+1} + P_{w,k}^T \pi''_k + S_{w,k}^T \sigma''_k = 0 \end{array}\right]_{k=j+1}^{j+h-1} \\
&\alpha''_{j+h} - A_{j+h}^T \alpha''_{j+h+1} + P_{x,j+h}^T \pi''_{j+h} + S_{x,j+h}^T \sigma''_{j+h} = -F_{j+h}^T \phi' \\
&\beta''_{j+h} - B_{j+h}^T \alpha''_{j+h+1} + P_{w,j+h}^T \pi''_{j+h} + S_{w,j+h}^T \sigma''_{j+h} = 0 \\
&\alpha''_{j+h+1} + F_{j+h+1}^T \phi'' = 0
\end{aligned} \tag{4.5}$$

Now, d'' is not dual feasible for the the subproblem represented from the shifted node because of the extra term: $-F_{j+h}^T$. Additionally, setting the newly introduced dual variables to zero generally results in overly conservative lower bounds. Therefore, we are looking for a correction term that can solve both issues. With this in mind, assume $F_{j+h} \in \mathbb{R}^{N_r} \times \mathbb{R}^{n_x}$ and denote with $F_{j+h}^{[r]}$ the r-th row of F_{j+h}. Then, for $r \in \{1, \cdots, N_r\}$, consider the following problems:

$$\begin{aligned}
\min_{x,w} \quad & \eta_{j+h} + \eta_{j+h+1} - F_{j+h}^{[r]} x_{j+h} \\
\text{s.t.} \quad & x_{j+1} = \bar{x}_{j+1} \\
& \left[\begin{array}{l} A_k x_k + B_k w_k = x_{k+1} \\ \underline{w}_k \le w_k \le \bar{w}_k \\ P_{x,k} x_k + P_{w,k} w_k \le p_k \end{array}\right]_{k=j+1}^{j+h} \\
& S_{x,j+h} x_{j+h} + S_{w,j+h} w_{j+h} - \underline{1}\,\eta_{j+h} \le s_{j+h} \\
& F_{j+h+1} x_{j+h+1} - \underline{1}\,\eta_{j+h+1} \le f_{j+h+1}
\end{aligned} \tag{$\mathcal{P}_{\text{Corr.}}^{(j+1,r)}$}$$

The duals of the $\mathcal{P}_{\text{Corr.}}^{(j+1,r)}$ read as:

$$\begin{aligned}
\max_{\substack{\alpha,\beta,\pi,\\ \sigma,\phi}} \quad & -\bar{x}_{j+1}^T \alpha_{j+1} - \bar{w}^T \max\{\beta, 0\} - \underline{w}^T \min\{\beta, 0\} \\
& \qquad -c^T \pi - s_{j+h}^T \sigma_{j+h} - f_{j+h+1}^T \phi \\
\text{s.t.} \quad & \\
& \left[\begin{array}{l} \alpha_k - A_k^T \alpha_{k+1} + P_{x,k}^T \pi_k = 0 \\ \beta_k - B_k^T \alpha_{k+1} + P_{w,k}^T \pi_k = 0 \\ \pi_k \ge 0 \end{array}\right]_{k=j+1}^{j+h-1} \\
& \alpha_{j+h} - A_{j+h}^T \alpha_{j+h+1} + P_{x,j+h}^T \pi_{j+h} + S_{x,j+h}^T \sigma_{j+h} = F_{j+h}^{[r]\,T} \\
& \beta_{j+h} - B_{j+h}^T \alpha_{j+h+1} + P_{w,j+h}^T \pi_{j+h} + S_{w,j+h}^T \sigma_{j+h} = 0 \\
& \underline{1}^T \sigma_{j+h} = 1; \;\; \pi_{j+h}, \sigma_{j+h} \ge 0 \\
& \alpha_{j+h+1} + F_{j+h+1}^T \phi = 0 \\
& \underline{1}^T \phi = 1; \;\; \phi \ge 0
\end{aligned} \tag{$\mathcal{D}_{\text{Corr.}}^{(j+1,r)}$}$$

Denote the dual solution of each of the above problems as follows:

$$\left\{ \left(\{\alpha_k^{[r]}\}_{k=j}^{j+h+1}, \{\beta_k^{[r]}\}_{k=j}^{j+h}, \{\pi_k^{[r]}\}_{k=j}^{j+h}, \sigma_{j+h}^{[r]}, \phi^{[r]} \right) \right\}_{r=1}^{N_r} \tag{4.6}$$

and extend it defining the following dummy multipliers:

$$\left\{ \{\sigma_k^{[r]} := 0\}_{k=j}^{j+h}, \{\mu_k^{[r]} := 0\}_{k=j}^{j+h} \right\}$$

As a result, we now have N_r points:

$$\Delta d^{[r]} := \left(\{\alpha_k^{[r]}\}_{k=j}^{j+h+1}, \{\beta_k^{[r]}\}_{k=j}^{j+h}, \{\pi_k^{[r]}\}_{k=j}^{j+h}, \{\sigma_k^{[r]}\}_{k=j}^{j+h}, \{\mu_k^{[r]}\}_{k=j}^{j+h}, \phi^{[r]} \right) \tag{4.7}$$

with the dimensionality of a dual point for a shifted node.

Next, with a slight abuse of notation, we can define the following linear combination of the obtained dual points:

$$\Delta d(w) = \sum_{r=1}^{N_r} w_r \Delta d^{[r]} \ : \ w \in \mathbb{R}^{N_r} \tag{4.8}$$

Then, since for each node it holds: $1^T \phi' = 1$ and $\phi' \geq 0$, we have:

$$\begin{array}{l}
\left[\begin{array}{l}
\alpha_k(\phi') - A_k^T \alpha_{k+1}(\phi') + P_{x,k}^T \pi_k(\phi') + S_{x,k}^T \sigma_k(\phi') = 0 \\
\beta_k(\phi') - B_k^T \alpha_{k+1}(\phi') + P_{w,k}^T \pi_k(\phi') + S_{w,k}^T \sigma_k(\phi') = 0 \\
\underline{1}^T \sigma_k(\phi') = 1; \ \ \pi_k(\phi'), \sigma_k(\phi') \geq 0
\end{array} \right]_{k=j+1}^{j+h-1} \\
\alpha_{j+h}(\phi') - A_{j+h}^T \alpha_{j+h+1}(\phi') + P_{x,j+h}^T \pi_{j+h}(\phi') + S_{x,j+h}^T \sigma_{j+h}(\phi') = F_{j+h}^T \phi' \\
\beta_{j+h}(\phi') - B_{j+h}^T \alpha_{j+h+1}(\phi') + P_{w,j+h}^T \pi_{j+h}(\phi') + S_{w,j+h}^T \sigma_{j+h}(\phi') = 0 \\
\underline{1}^T \sigma_{j+h}(\phi') = 1; \ \ \pi_{j+h}(\phi'), \sigma_{j+h}(\phi') \geq 0 \\
\alpha_{j+h+1}(\phi') + F_{j+h+1}^T \phi(\phi') = 0 \\
\underline{1}^T \phi(\phi') = 1; \ \ \phi(\phi') \geq 0
\end{array} \tag{4.9}$$

Finally, it is easy to see that the dual point:

$$d''' := d'' + \Delta d(\phi') \tag{4.10}$$

is feasible for the problem represented by the shifted node.

4.4.3 Computing a New Lower-Bound

Denote the optimal objective value of a convex problem $\mathcal{P}$ with $\mathcal{V}(\mathcal{P})$ and with $\underline{\mathcal{V}}(\mathcal{P})$ any lower bound for it. Using the dual point built in the last section, it is now possible to obtain an objective lower bound for the shifted node:

$$\underline{\mathcal{O}}_{\text{obj.}}(\mathcal{P}_{\text{BB}}^{(j+1)}) := \underline{\mathcal{O}}_{\text{obj.}}(\mathcal{P}_{\text{BB}}^{(j)}) + \mathcal{O}_{\text{contraction}} + \mathcal{O}_{\text{expansion}} \tag{4.11}$$

where:

$$\mathcal{O}_{\text{contraction}} := -\eta_j + \left(A_j \bar{x}_j + B_j w'_j - \bar{x}_{j+1}\right)^T \alpha'_{j+1} \tag{4.12}$$

derives from the removal of the initial constraints and from the imposition of the new initial state, and where:

$$\mathcal{O}_{\text{expansion}} := \left(\left[\mathcal{O}_{\text{obj.}}(\mathcal{P}_{\text{Corr.}}^{(j+1,1)}) \;\cdots\; \mathcal{O}_{\text{obj.}}(\mathcal{P}_{\text{Corr.}}^{(j+1,N_r)})\right] + f_{j+h}^T\right)\phi' \tag{4.13}$$

is due to the removal of the old tail cost and the addition of the new stage cost and the new tail cost.

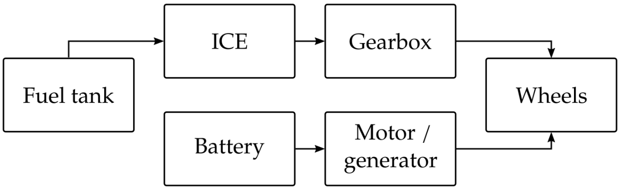
While the expression for $\mathcal{O}_{\text{expansion}}$ can be obtained straightforwardly from the objective function of the dual of the shifted node, the derivation of $\mathcal{O}_{\text{contraction}}$ is less direct and deserves some explanation. We refer the interested reader to Appendix B at the end of this chapter for more information.

4.5 Numerical Validation

4.5.1 The Case of Study

The model used for the numerical evaluation of the performances of MIRT-OC is a convex dynamical model of a Parallel Hybrid Electric Vehicle (PHEV). PHEVs are vehicles whose power-train features one or more electric motors/generators and where both the engines and the motors/generators are mechanically connected to the wheels. Many possible ways of designing a PHEV power-train exist, however only the most common of the topologies will be considered: Figure 4.5.1.

Figure 4.1: Parallel Hybrid Electric Vehicle Topology



During normal operation, road vehicles are constantly aware of their instantaneous speed and continuously receive a torque/power request from the driver. Assuming that the requested power profile is followed exactly, the control optimization problem considered in this section consists in minimizing the fuel consumption of the vehicle while ensuring the stability of the battery state of charge around a predefined set-point $\mathrm{E}^{\mathrm{des}}$.[5]

In order to model the road characteristics together with the driver's inputs over time, a standard drive cycle was used. The considered drive cycle is a derivation of the "EPA Urban Dynamometer Driving Schedule" called FTP-75 (or Federal Test Procedure 75) and it is commonly used for emission certification and fuel economy testing of light-duty vehicles in the United States (Fig. 4.2).

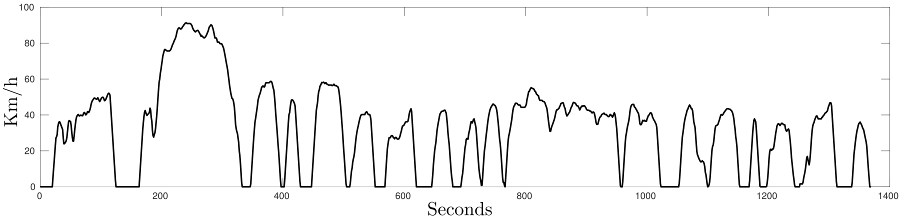


Figure 4.2: the Federal Test Procedure 75

Vehicle Body

The HEV considered in this case of study is a small consumer car. Given the vehicle speed profile over time, $\mathrm{v}^{\mathrm{cycle}}(t)$, the longitudinal dynamics of the car can be modeled as follows:

$$\begin{aligned}\mathrm{F}^{\mathrm{req}}(t)\ [\mathrm{N}] := \quad & \mathrm{M}\cdot\dot{\mathrm{v}}^{\mathrm{cycle}}(t) + \mathrm{F}_{\mathrm{friction}}\cdot\chi^{+}(\mathrm{v}^{\mathrm{cycle}}(t)) + \\ & 0.5\cdot\rho_{\mathrm{air}}\cdot\mathrm{C}_{\mathrm{drag}}\cdot\mathrm{A}_{\mathrm{frontal}}\cdot(\mathrm{v}^{\mathrm{cycle}}(t))^{2}\end{aligned} \tag{4.14}$$

where $\chi^{+}(x)$ is a function whose value is one if $x > 0$ and zero otherwise, and the following parameters were used:

Parameter	M	$\mathrm{F}_{\mathrm{friction}}$	ρ_{air}	$\mathrm{C}_{\mathrm{drag}}$	$\mathrm{A}_{\mathrm{frontal}}$
Value	971.5	0.005	1.225	0.35	2
Units	kg	N	$\mathrm{kg/m^3}$	-	$\mathrm{m^2}$

From (4.14) and considering $\mathrm{R}^{\mathrm{wheels}} := 0.285m$, we can deduce the following requirements on the vehicle power-train:

$$\begin{cases}\mathrm{P}^{\mathrm{req}}(t)\ [\mathrm{W}] := \mathrm{F}^{\mathrm{req}}(t)\cdot\mathrm{v}^{\mathrm{cycle}}(t) \\ \omega^{\mathrm{cycle}}(t)\ [\mathrm{rad/s}] := \mathrm{v}^{\mathrm{cycle}}(t)/\mathrm{R}^{\mathrm{wheels}}\end{cases} \tag{4.15}$$

---

[5] In our case, $\mathrm{E}^{\mathrm{des}}$ is the 65% of the nominal maximum charge of the battery.

Finally, the accessory systems of the vehicle as the external lighting, the internal air conditioning, the on-board electronics and the various control systems, are assumed to continuously draw from the battery a power $\mathrm{P}^{\mathrm{aux.}}$ equal to 0.3kW.

Battery

The battery pack was modeled with the help of the simple open-circuit-voltage based model that can be found in [51]. The battery size and output voltage were obtained emulating the newest Toyota Yaris Hybrid. The result is a battery pack with the capacity of approximately 1kWh operating at approximately 145 volts and obtained connecting in parallel three stripes of 44 battery cells (connected in series) each. Denoting with E the residual energy in the battery and with $\mathrm{P}^{\mathrm{BAT}}$ the power being drawn from the battery (before losses), the equations modelling the discharge dynamics and the output power for the considered battery pack read as follows:

$$\begin{aligned}
&\dot{\mathrm{E}}\ [\mathrm{W}] = \mathrm{P}^{\mathrm{BAT}} \\
&\boldsymbol{Pout}^{\mathrm{BAT}}(\mathrm{P}^{\mathrm{BAT}}, \mathrm{E})\ [\mathrm{W}] := \mathrm{P}^{\mathrm{BAT}} - \frac{R{\cdot}C{\cdot}\mathrm{P}^{\mathrm{BAT}^2}}{2{\cdot}\mathrm{E} - 132{\cdot}\mathrm{U}_0^2{\cdot}C} \\
&132{\cdot}\mathrm{E}_{\mathrm{cell,max}} \leq \mathrm{E}\ [\mathrm{J}] \leq 132{\cdot}\mathrm{E}_{\mathrm{cell,max}}
\end{aligned} \tag{4.16}$$

where E, $\mathrm{P}^{\mathrm{BAT}}$, represent the residual stored energy and the internal power of the battery, and C, R, U_0 and $\mathrm{E}_{\mathrm{cell,max}}$ represent, respectively, the energy stored in the capacitance, the internal resistance, the open-circuit-voltage and the capacity of a single battery cell.

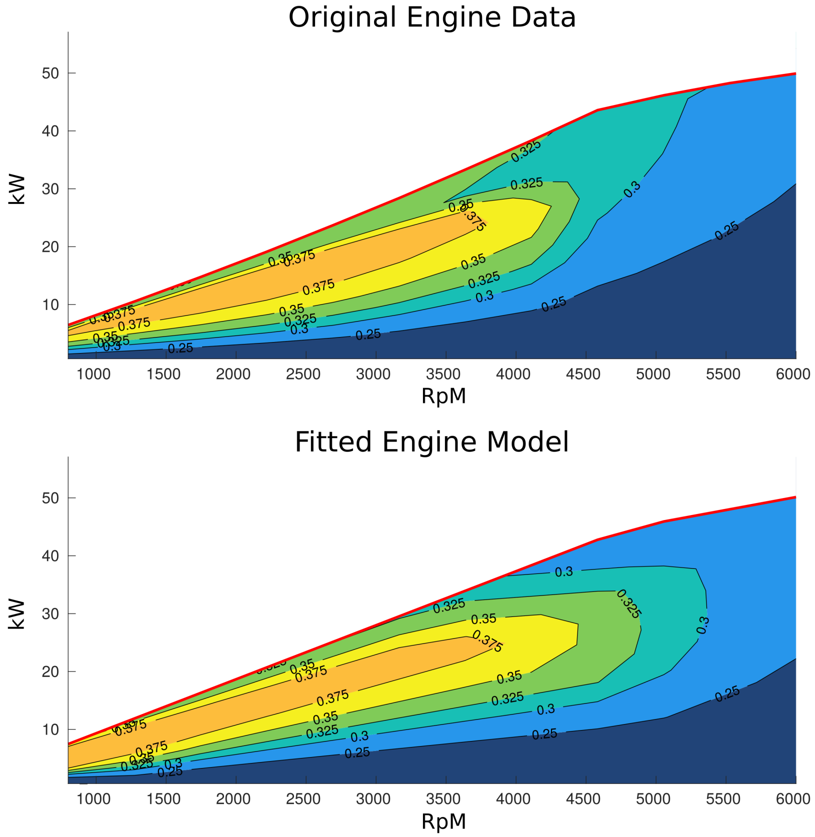
Parameter	C	R	U_0	$\mathrm{E}_{\mathrm{cell,max}}$
Value	51782	0.01	3.3	27854
Units	Farad	Ω	V	J

Note that the resulting battery model is concave in $\mathrm{P}^{\mathrm{BAT}}$.

Internal Combustion Engine

For the PHEV model a 50kW Internal Combustion Engine (ICE) was chosen. The ICE transforms the chemical energy of the fuel into mechanical energy and heat. The efficiency of this process depends mostly on the temperature of the engine, its operating speed and the power output. In order to model such efficiency profile a set of measurements taken from a real engine model [94] was used. The dataset relative to the engine fuel consumption while running

Figure 4.3: Engine:Fitted Model VS Original Data (Efficiency)



Original Engine Data

Fitted Engine Model

at approximately 80-90^{o}C along with the maximum possible power output is represented in figure 4.5.1.

The data were fitted using MATLAB curve fitting toolbox obtaining the following fuel consumption map:

$$\boldsymbol{dFin}^{\mathrm{ICE}}(\mathrm{P}^{\mathrm{ICE}}, \omega^{\mathrm{ICE}})\ [\mathrm{g}] := \max_{l \in \{1,\cdots,5\}} \boldsymbol{dFin}_l^{\mathrm{ICE}}(\mathrm{P}^{\mathrm{ICE}}, \omega^{\mathrm{ICE}}) \tag{4.17}$$

where:

$$\begin{aligned} \boldsymbol{dFin}_l^{\mathrm{ICE}}(\mathrm{P}^{\mathrm{ICE}}, \omega^{\mathrm{ICE}}) := \quad & f_{(l,2,0)}\omega_n^2 + f_{(l,0,2)}\mathrm{P}_n^2 + f_{(l,1,1)}\omega_n \mathrm{P}_n + \\ & f_{(l,1,0)}\omega_n + f_{(l,0,1)}\mathrm{P}_n + f_{(l,0,0)} \end{aligned}$$

$$\mathrm{P}_n := \frac{\mathrm{P}^{\mathrm{ICE}}}{50e3} \quad \text{and} \quad \omega_n := \frac{\omega^{\mathrm{ICE}}}{628.32}$$

and:

l	$f_{(l,2,0)}$	$f_{(l,0,2)}$	$f_{(l,1,1)}$	$f_{(l,1,0)}$	$f_{(l,0,1)}$	$f_{(l,0,0)}$
1	1.8771	2.0258	0.1057	-1.0349	1.2862	0.2166
2	1.0000	9.3144	-5.4182	0.0546	-1.2264	0.1662
3	0.0746	4.0246	-1.0000	0.0000	1.5890	0.0000
4	0.0000	0.0000	0.0000	0.6964	2.2740	-0.0483
5	0.0000	0.0000	0.0000	-2.5000	6.0000	-0.07

Similarly, the measurements on the maximum output power were fitted, obtaining:

$$\boldsymbol{P}^{\mathrm{ICE}}_{\max}(\omega^{\mathrm{ICE}})\ [\mathrm{W}] := 50e3 \cdot \min\left(1.12\omega_n, 0.536\omega_n + 0.4668\right) \tag{4.18}$$

Electric Motor/Generator

For our HEV model a 25kW Electric Motor/Generator (EMG) was chosen. The EMG can either draw electric power from the battery pack to give power to the wheels or draw power from the wheels to recharge the battery pack. The procedure followed to model the efficiency and power limits of the EMG is analogous to the one used to model the ICE.

The fitting resulted in the following power absorption map:

$$\boldsymbol{Pin}^{\mathrm{EMG}}(\mathrm{P}^{\mathrm{EMG}}, \omega^{\mathrm{EMG}})\ [\mathrm{W}] := \max_{l \in \{1,2,3\}} \boldsymbol{Pin}^{\mathrm{EMG}}_l(\mathrm{P}^{\mathrm{EMG}}, \omega^{\mathrm{EMG}}) \tag{4.19}$$

where:

$$\boldsymbol{Pin}^{\mathrm{EMG}}_l(\mathrm{P}^{\mathrm{EMG}}, \omega^{\mathrm{EMG}}) := p_{(l,2,0)}\omega_n^2 + p_{(l,0,2)}\mathrm{P}_n^2 + p_{(l,1,0)}\omega_n + p_{(l,0,1)}\mathrm{P}_n$$

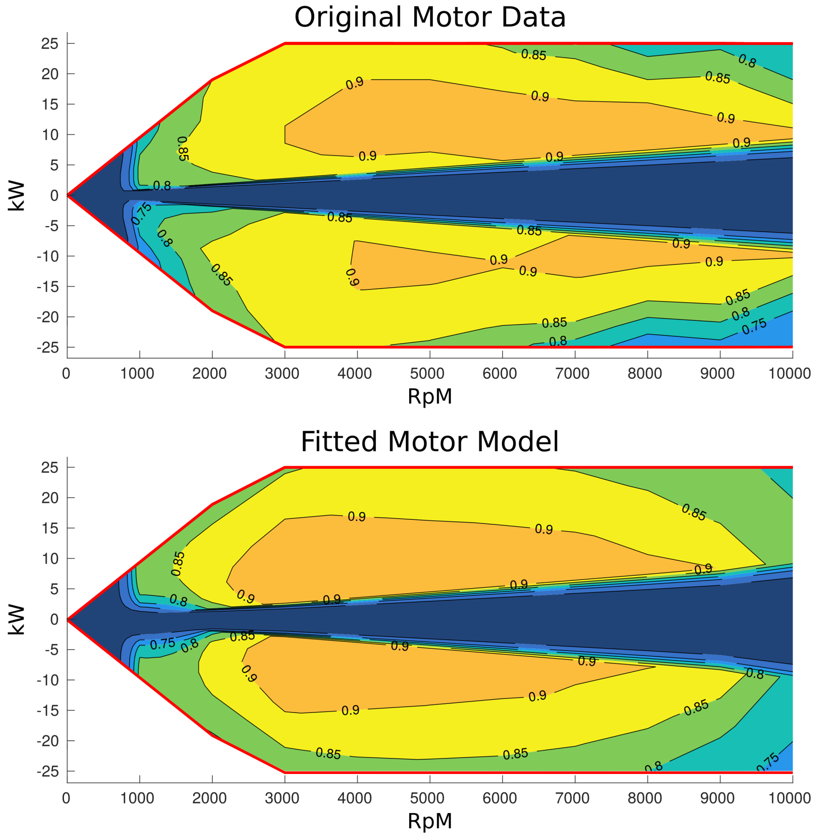
with:

$$\mathrm{P}_n := \frac{\mathrm{P}^{\mathrm{EMG}}}{25e3} \quad \text{and} \quad \omega_n := \frac{\omega^{\mathrm{EMG}}}{1047.2}$$

and:

l	$p_{(l,2,0)}$	$p_{(l,0,2)}$	$p_{(l,1,0)}$	$p_{(l,0,1)}$
1	0.6	3652.9	513.6	25000.0
2	-66725.0	6062.6	15247.5	25000.0
3	9860.0	2191.8	-9757.5	-29375.0

Figure 4.4: Electric Motor: Fitted Model VS Original Data (Efficiency)



Original Motor Data

Fitted Motor Model

In this case, the resulting input-power map is non-convex but can still be used in our convex model. In fact, the map depends convexely on the output power and the speed of the motor/generator is at each instant is a constant deriving from the driving cycle.

For the maximum-power curve we obtained the following function:

$$\boldsymbol{P}^{\mathrm{EMG}}_{\max}(\omega^{\mathrm{EMG}})\ [\mathrm{W}] := \min(25000, 94645\omega_n) \tag{4.20}$$

Gearbox and Torque Coupling

For each gear in the gearbox a reference vehicle speed set-point was chosen and a gear-ratio was computed so that the engine would operate at 2500rpm whenever the speed of the vehicle would equal the given set-point. The resulting ratios can be found in the following table:

Gear	1st	2nd	3rd	4th	5th
Ref. Speed (km/h)	15	30	55	85	115
Ratio	17.72	8.86	4.83	3.13	2.31

The fixed EMG-to-wheels ratio, was designed to keep the EMG speed within the maximal torque region in most city-like working conditions:

$$\mathrm{R}^{\mathrm{EMG}} := 5.3721 \text{ (i.e. 3000rpm at 60km/h)}$$

Remark: The modeling of the mechanical drive-train hereby presented is the result of a series of practical considerations. Thus, it is possible that a better design could be found via optimization or more expert insight. Anyway, the obtained model is functional enough to serve the purpose of showcasing the properties of MIRT-OC.

Discrete Controls

In order to model the discrete behavior of the power-train, some integer variables have to be introduced. Thus, consider the following binary controls:

$$\begin{aligned} \mathrm{Off}_k &: \mathrm{Off}_k = 1 \text{ means that the engine is off and the gearbox is in neutral.} \\ [\mathrm{G}_{i,k}]_{i=1}^{5} &: \mathrm{G}_{i,k} = 1 \text{ means that the engine is on and the } i\text{-th gear is in use.} \end{aligned}$$

The above controls are mutually exclusive. Therefore, the following “Special Ordered Set” constraint needs to be added to the formulation:

$$\mathrm{Off}_k + \sum_{j=1}^{5} \mathrm{G}_{i,k} = 1$$

Finally, in order to assign a fuel cost to the turning on and off of the engine an auxiliary state pOff can be defined in such a way that:

$$\mathrm{pOff}_{k+1} = \mathrm{Off}_k$$

4.5.2 The resulting model

Once all the components have been modeled we can finally define the optimal control problem of interest. In order to get a manageable formulation for the subproblem to solve at each MPC iteration we will use DMS with a time discretization resolution of one second. The result reads as follows:

$$
\begin{array}{ll}
\min & \displaystyle\sum_{k=j}^{j+h-1} \left(\mathrm{dF}_k + K^{\mathrm{E2F}}\mathrm{P}_k^{\mathrm{BAT}}\right) + \eta \\
\text{s.t.} & \text{\# Initial state} \\
& \left[\mathrm{F}_j, \mathrm{E}_j, \mathrm{pOff}_j\right]^T = \left[\bar{\mathrm{F}}_j, \bar{\mathrm{E}}_j, \mathrm{p\bar{O}ff}_j\right]^T \\
& \left[\begin{array}{l}
\text{\# Dynamic Constraints} \\
\mathrm{F}_{k+1} = -\mathrm{dF}_k + \mathrm{F}_k \\
\mathrm{E}_{k+1} = -\mathrm{P}_k^{\mathrm{BAT}} + \mathrm{E}_k \\
\mathrm{pOff}_{k+1} = \mathrm{Off}_k \\
\text{\# Integer Requirements and SoS1 Constraints} \\
\mathrm{Off}_k \in \{0,1\}; \quad [\mathrm{G}_{i,k} \in \{0,1\}]_{i=1}^{5} \\
\mathrm{Off}_k + \sum_{j=1}^{5} \mathrm{G}_{i,k} = 1 \\
\text{\# Battery and Electric Motor/Generator} \\
\mathrm{E}^{\min} \leq \mathrm{E} \leq \mathrm{E}^{\max} \\
\boldsymbol{Pin}^{\mathrm{EMG}}(\mathrm{P}_k^{\mathrm{EMG}}, \mathrm{R}^{\mathrm{EMG}}\omega_k^{\mathrm{cycle}}) + \mathrm{P}^{\mathrm{aux.}} \leq \boldsymbol{Pout}^{\mathrm{BAT}}(\mathrm{P}_k^{\mathrm{BAT}}, \mathrm{E}_k) \\
\left|\mathrm{P}_k^{\mathrm{EMG}}\right| \leq \boldsymbol{P}_{\max}^{\mathrm{EMG}}(\mathrm{R}^{\mathrm{EMG}}\omega_k^{\mathrm{cycle}}) \\
\text{\# Internal Combustion Engine} \\
\sum_{j=1}^{5} \mathrm{G}_{j,k}\mathrm{R}_j^{\mathrm{GBX}}\omega_k^{\mathrm{cycle}} \leq \omega_{\max}^{\mathrm{ICE}} \\
\mathrm{P}^{\mathrm{ICE}} \leq \boldsymbol{P}_{\max}^{\mathrm{ICE}}(\sum_{j=1}^{5} \mathrm{G}_{j,k}\mathrm{R}_j^{\mathrm{GBX}}\omega_k^{\mathrm{cycle}}) \\
\mathrm{dF}_k \geq \boldsymbol{dFin}^{\mathrm{ICE}}(\mathrm{P}^{\mathrm{ICE}}, \sum_{j=1}^{5} \mathrm{G}_{j,k}\omega_k^{\mathrm{cycle}}\mathrm{R}_j^{\mathrm{GBX}}) + \\
\qquad\qquad \mathrm{Off}_k \cdot ((1 - \mathrm{pOff}_k) \cdot f_{\mathrm{start}}/2) \\
\text{\# Power Requirement} \\
\mathrm{P}_k^{\mathrm{req}} \leq \mathrm{P}_k^{\mathrm{EMG}} + \mathrm{P}_k^{\mathrm{ICE}}
\end{array}\right]_{k=j}^{j+h-1} \\
& \text{\# Tail Cost} \\
& \boldsymbol{K}^{\mathrm{tail}}(\bar{\mathrm{E}}_j)(\mathrm{E}^{\mathrm{des}} - \mathrm{E}_{j+h})^2 \leq \eta
\end{array}
\qquad (\mathcal{P}_{\mathrm{CoS}}^{(j)})
$$

where $K^{\mathrm{E2F}} := 200\mathrm{g/kWh}$ is a rough estimation of the the maximum efficiency of the engine and:

$$
\boldsymbol{K}^{\mathrm{tail}}(\bar{\mathrm{E}}_j) := 10\frac{\mathrm{E}^{\max} - \bar{\mathrm{E}}_j}{(\mathrm{E}^{\max} - \mathrm{E}^{\min}) + (\mathrm{E}^{\max} - \bar{\mathrm{E}}_j)} \tag{4.21}
$$

is a time-varying parameter that influences the relative importance of the stabilizing terminal cost over the stage costs and prevents the controller from planning a waste of electric energy if the state of charge surpasses the desired set-point.

4.5.3 Numerical Results

For this evaluation we considered two different algorithms:

1. **Classical MI-MPC**: Each of the MPC subproblems is solved as a stand-alone problem with LP/NLP-B&B.
2. **MIRT-OC**: The full MIRT-OC algorithm, with both linearization and tree shifting procedure.

Each of the algorithms have been tested using 5, 10 and 15 seconds of prediction length (1 step per second) and performing 120 MPC iterations in three different scenarios:

- No disturbance: The dynamic model of the system is exact and no measurement disturbance is added.
- Medium disturbance: The actual dynamic of the system is:

  $$x_{k+1} := (\mathbb{I} + \mathcal{N}(10\%))(A_k x_k + B_k w_k)$$

  Moreover, the state measurements are disturbed as follows:

  $$\tilde{x}_k := x_k + \mathcal{N}(10\%)(\max(x_{k+1}) - \min(x_{k+1}))$$

  where $\mathcal{N}(x\%)$ is a normally distributed variable with mean zero and standard deviation: $\frac{x}{100}$.
- High disturbance: The real dynamic of the system is:

  $$x_{k+1} := (\mathbb{I} + \mathcal{N}(100\%))(A_k x_k + B_k w_k)$$

  Moreover, the state measurements are disturbed as follows:

  $$\tilde{x}_k := x_k + \mathcal{N}(10\%)(\max(x_{k+1}) - \min(x_{k+1}))$$

Samples of the resulting solutions (using 10 seconds of prediction) can be found in Figure 4.5.

For the sake of comparison, in all the tests performed B&B (as implemented in OpenBB [87]) was run in its pure form, disabling any kind of heuristic. In fact, heuristic methods can have very different outcomes even when applied to just slightly different problem instances, hindering the assessment of the relative performances among the schemes. During the testing phase, ipopt ([83]) was used as NLP solver and CLP ([88]) as LP subsolver. The obtained results are summarized in Table 4.1.

No disturbance:

	Classical	**MIRT-OC**	**Classical**	**MIRT-OC**	**Classical**	**MIRT-OC**
#LPs	1100	329	4044	1251	8471	2855
#NLPs	138	132	159	143	165	141
	5s of prediction		**10s of prediction**		**15s of prediction**	

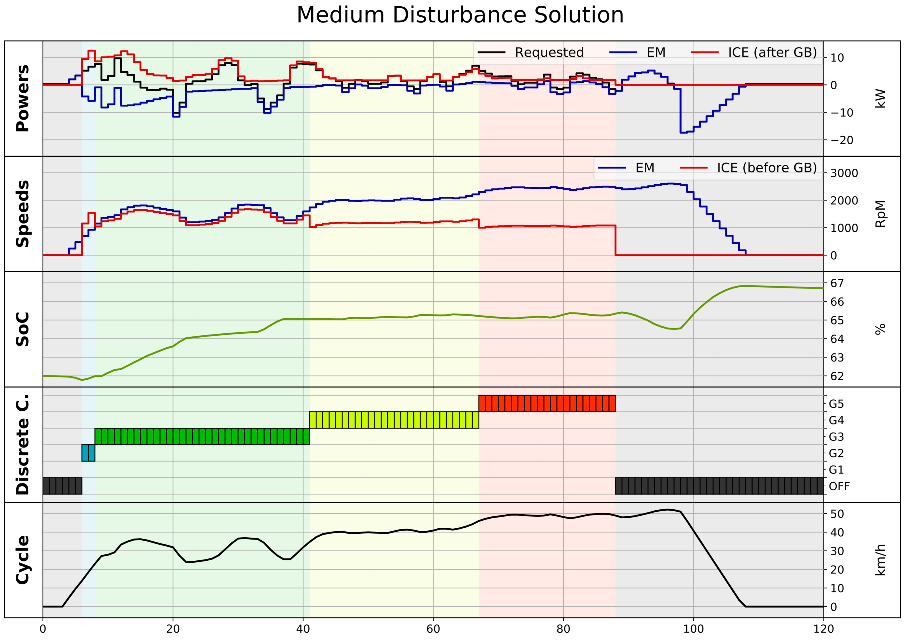
Medium disturbance:

	Classical	**MIRT-OC**	**Classical**	**MIRT-OC**	**Classical**	**MIRT-OC**
#LPs	1111	341	3898	1441	7426	3357
#NLPs	137	133	161	146	159	141
	5s of prediction		**10s of prediction**		**15s of prediction**	

High disturbance:

	Classical	**MIRT-OC**	**Classical**	**MIRT-OC**	**Classical**	**MIRT-OC**
#LPs	1126	345	3390	1318	13861	5534
#NLPs	137	132	153	142	282	221
	5s of prediction		**10s of prediction**		**15s of prediction**	

Table 4.1: Comparison between MIRT-OC and the classical MI-MPC approach for different scenarios/prediction lengths



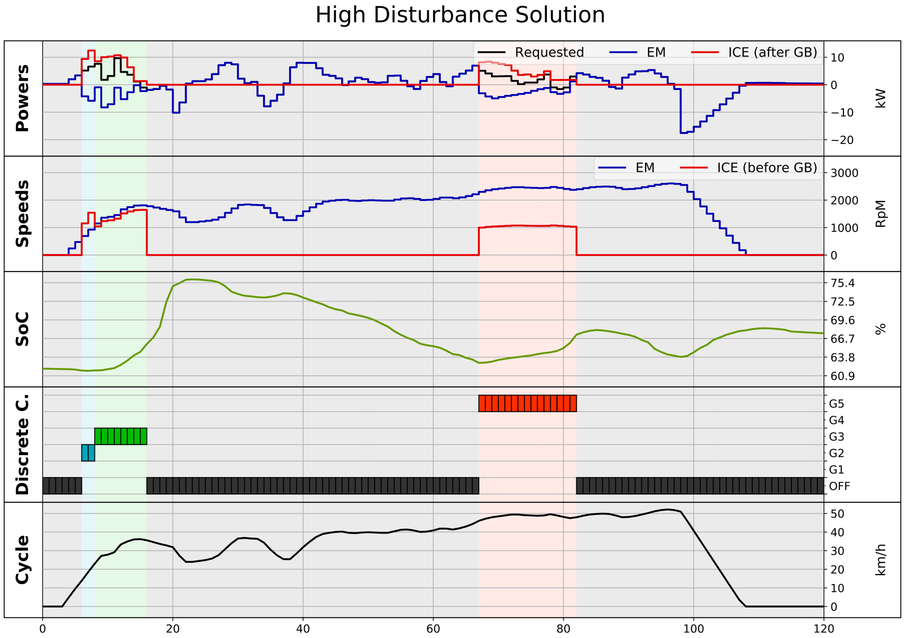


Figure 4.5: Sample solutions with 15 seconds of prediction

4.6 Summary and Final Comments

In this chapter we have detailed the functioning of MIRT-OC: a novel approach for mixed-integer non-linear MPC aimed at reducing the computational cost of each MPC iteration by reusing the information generated during the previous iterations.

The collected evidence shows how, in the analyzed scenarios, MIRT-OC is capable of largely reducing the complexity of the MPC iterations regardless of the presence of heavy model mismatch and disturbances. Such characteristics might allow the user to define larger MPC subproblems while respecting the predefined time-per-iteration limit. In turn, this can allow the MPC scheme to produce higher quality control actions for the benefit of the performances of the controlled system.

The performed tests resulted in a very promising picture on the performances of MIRT-OC. However, the current version of the algorithm is based on a pure B&B implementation. Consequently, the author has reason to believe that, with additional testing/development, it is possible to develop/select heuristic methods and preprocessing techniques capable of providing MIRT-OC with an increased performance edge.

4.7 Appendix A: Getting Rid Of Discrete States

Consider the following discrete-time mixed-integer linear dynamic constraints:

$$\begin{array}{l} x_0 = \bar{x}_0; \;\; y_0 = \bar{y}_0 \\ \left[\begin{array}{l} x_{k+1} = A_{x,x} x_k + A_{x,y} y_k + B_x w_k \\ y_{k+1} = A_{y,y} y_k + B_x w_k \\ y_k \in [\underline{y}_k, \bar{y}_k] \cap \mathbb{N} \end{array} \right]_{k=0}^{N} \\ y_N \in [\underline{y}_N, \bar{y}_N] \cap \mathbb{N} \end{array}$$

Now, adding an auxiliary control variable, Δy_k, it is easy to obtain the following equivalent set of constraints:

$$\begin{array}{l} x_0 = \bar{x}_0; \;\; y_0 = \bar{y}_0 \\ \left[\begin{array}{l} x_{k+1} = A_{x,x} x_k + A_{x,y} y_k + B_x w_k \\ y_{k+1} = y_k + \Delta y_k \\ \Delta y_k = (A_{y,y} - \mathbb{I}) y_k + B_y w_k \\ y_k \in [\underline{y}_k, \bar{y}_k] \cap \mathbb{N} \end{array} \right]_{k=0}^{N} \\ y_N \in [\underline{y}_N, \bar{y}_N] \cap \mathbb{N} \end{array}$$

where $\mathbb{I}$ represents the identity matrix of appropriate dimensionality. Then, since: $y_{k+1}, y_k \in \mathbb{N}$, we have: $y_{k+1} - y_k \in \mathbb{N}$. Therefore, assuming $\bar{y}_0 \in \mathbb{N}$ (feasibility), the above constraints are equivalent to:

$$\begin{array}{l} x_0 = \bar{x}_0; \;\; y_0 = \bar{y}_0 \\ \left[\begin{array}{l} x_{k+1} = A_{x,x} x_k + A_{x,y} y_k + B_x w_k \\ y_{k+1} = y_k + \Delta y_k \\ \Delta y_k = (A_{y,y} - \mathbb{I}) y_k + B_y w_k \\ y_k \in [\underline{y}_k, \bar{y}_k]; \;\; \Delta y_k \in \mathbb{N} \end{array} \right]_{k=0}^{N} \\ y_N \in [\underline{y}_N, \bar{y}_N] \end{array}$$

which describes a system with no discrete states but one additional set of discrete controls Δy.

4.8 Appendix B: Derivation of $\mathcal{O}_{\text{contraction}}$

Comparing the dual objective each subproblem with the one of its transformed version we can define:

$$\begin{aligned}\mathcal{O}_{\text{contraction}} :=\ & \bar{x}_j{}^T \alpha'_j + \bar{w}_j{}^T \max\{\beta'_j, 0\} + \underline{w}_j{}^T \min\{\beta'_j, 0\} \\ & + c_j^T \pi'_j + s_j^T \sigma'_j + m_j^T \mu'_j - \bar{x}_{j+1}{}^T \alpha'_{j+1}\end{aligned}$$

Now, denoting with $\delta_j := \bar{x}_{j+1} - A_j \bar{x}_j - B_j w'_j$ the model mismatch detected in the j-th iteration, and substituting $\bar{x}_{j+1}$ with $A_j \bar{x}_j + B_j w'_j + \delta_j$, we obtain:

$$\begin{aligned}\mathcal{O}_{\text{contraction}} =\ & \bar{x}_j{}^T \alpha'_j + \bar{w}_j{}^T \max\{\beta'_j, 0\} + \underline{w}_j{}^T \min\{\beta'_j, 0\} \\ & + c_j^T \pi'_j + s_j^T \sigma'_j + m_j^T \mu'_j \\ & - \bar{x}_j^T A_j^T \alpha'_{j+1} - w'_j{}^T B_j^T \alpha'_{j+1} - \delta_j{}^T \alpha'_{j+1}\end{aligned}$$

Furthermore, using the constraints defined in $\mathcal{D}_{\text{BB}}^{(j)}$ to expand $A_j^T \alpha'_{j+1}$ and $B_j^T \alpha'_{j+1}$, the above expression can be rewritten as follows:

$$\begin{aligned}\mathcal{O}_{\text{contraction}} =\ & -\left(P_{x,j} \bar{x}_j + P_{w,j} u'_j - c_j\right)^T \pi'_j \\ & -\left(S_{x,j} \bar{x}_j + S_{w,j} u'_j - s_j\right)^T \sigma'_j \\ & -\left(w'_j - \bar{w}_j\right)^T \max\{\beta'_j, 0\} - \left(w_j - \underline{w}_j\right)^T \min\{\beta'_j, 0\} \\ & -\left(M_j w'_j - m_j\right)^T \mu'_j - \delta_j{}^T \alpha'_{j+1}\end{aligned}$$

Then, assuming that the original node was solved up to optimality and using the complementary slackness property of the dual solution, we obtain:

$$\mathcal{O}_{\text{contraction}} = -\left(S_{x,j} \bar{x}_j + S_{w,j} u'_j - s_j\right)^T \sigma'_j - \delta_j{}^T \alpha'_{j+1}$$

or equivalently:

$$\mathcal{O}_{\text{contraction}} = -\left(S_{x,j} \bar{x}_j + S_{w,j} u'_j - s_j - \underline{1}\, \eta'_j\right)^T \sigma'_j - \eta'_j \underline{1}^T \sigma'_j - \delta_j{}^T \alpha'_{j+1}$$

Now, recalling that the constraints in $\mathcal{D}_{\text{BB}}^{(j)}$ impose: $\underline{1}^T \sigma'_j = 1$, and applying the complementary slackness property once again we can rewrite the above expression as follows:

$$\mathcal{O}_{\text{contraction}} = -\eta'_j - \delta_j{}^T \alpha'_{j+1}$$

Therefore, the term $\mathcal{O}_{\text{contraction}}$ consists of the negative of the portion of the objective relative to the first time step and of a term proportional to the model mismatch.

4.9 Appendix C: Extensions and Different Applications

Delayed MPC and Adaptive Prediction Lengths

Observing the linearization and node shifting procedure it is possible to distinguish two distinct transformations being applied. In the first, the initial portion of the problem (the first time step) is removed from the problem and the effect of model mismatch is accounted for. In the second, the old tail cost is replaced by a new portion of control problem. For simplicity, this two steps have been presented as adding/removing one time step to/from the problem, however, this is not necessary to their functioning. In fact, it is easy to extend those operations to consider shifts of more than one step (delayed MPC) and it is even possible have them consider different shifts. In the latter case, the length of the prediction horizon can change from one MPC iteration to the next. This property can be used to adapt the length of the prediction horizon at run-time in order to find an adaptive compromise between the quality of the control actions to find and the computational cost of each iteration.

Quadratic Stage Costs

In the case in which the defined stage costs contain convex quadratic terms it might be beneficial to treat such terms directly in B&B without defining an epigraph reformulation for them. For instance, assume that the objective of $\mathcal{P}_{\text{NL}}^{(j)}$ would be defined as follows:

$$\sum_{k=j}^{j+h-1} \left(\frac{x_k^T Q_k x_k}{2} + \frac{w_k^T R_k w_k}{2} + \mathbf{s}_k(x_k, w_k) \right) + \mathbf{f}_{j+h}(x_{j+h}) \tag{4.22}$$

In such case, the strategy presented above for obtaining a feasible dual point for each shifted node will still hold in essence, requiring only minor adjustments. However, the objective lower bound for each of the shifted nodes would read as follows:

$$\underline{\mathcal{O}}_{\text{obj.}}(\mathcal{P}_{\text{BB}}^{(j+1)}) := \underline{\mathcal{O}}_{\text{obj.}}(\mathcal{P}_{\text{BB}}^{(j)}) + \mathcal{O}_{\text{contraction}} + \mathcal{O}_{\text{expansion}}$$

where:

$$\mathcal{O}_{\text{contraction}} := -\eta_j + \frac{y_j^T Q_j y_j}{2} + \frac{v_j^T R_j v_j}{2} + \left(A_j x_j^{(i-1)} + B_j w'_j - x_{j+1}^{(j)} \right)^T \alpha'_{j+1}$$

$$\begin{aligned}\mathcal{O}_{\text{expansion}} := \quad & \left(\left[\mathcal{O}_{\text{obj.}}(\mathcal{P}_{\text{Corr.}}^{(j+1,1)}) \ \cdots \ \mathcal{O}_{\text{obj.}}(\mathcal{P}_{\text{Corr.}}^{(j+1,N_r)})\right] + f_{j+h}^T\right)\phi' \\ & + \tfrac{1}{2}\textstyle\sum_{r=1}^{N_r} \phi'_r \left({y_{j+h}^{[r]}}^T Q_{j+h} y_{j+h}^{[r]} + {v_{j+h}^{[r]}}^T R_{j+h} v_{j+h}^{[r]}\right) \\ & - \tfrac{1}{2}\left(\textstyle\sum_{r=1}^{N_r} \phi'_r y_{j+h}^{[r]}\right)^T Q_{j+h} \left(\textstyle\sum_{r=1}^{N_r} \phi'_r y_{j+h}^{[r]}\right) \\ & - \tfrac{1}{2}\left(\textstyle\sum_{r=1}^{N_r} \phi'_r v_{j+h}^{[r]}\right)^T R_{j+h} \left(\textstyle\sum_{r=1}^{N_r} \phi'_r v_{j+h}^{[r]}\right)\end{aligned}$$

Discounted MPC

Assume the objective function of $\mathcal{P}_{\text{NL}}^{(j)}$ to be:

$$\sum_{k=j}^{j+h-1} \rho^{k-j}\mathbf{s}_k(x_k, w_k) + \rho^h \mathbf{f}_{j+h}(x_{j+h}) \quad : \quad \rho \in (0,1)$$

Then, the considered formulation would fall into the category of discounted MPC. MIRT-OC can be easily extended in order to deal with this new problem type. In fact, all it is necessary to do is to slightly modify the shifting strategy for the dual point stored in each node so that it applies multiplicative factor ρ^{-1} on the existing pre-shift dual multipliers, and to redefine the correction problems as follows:

$$\begin{aligned}\min_{x,w} \quad & \rho^{h-1}(\eta_{j+h} - F_{j+h}^{[r]} x_j) + \rho^h \eta_{j+1} \\ \text{s.t.} \quad & x_{j+1} = \bar{x}_{j+1} \\ & \begin{bmatrix} A_k x_k + B_k w_k = x_{k+1} \\ \underline{w}_k \le w_k \le \bar{w}_k \\ P_{x,k} x_k + P_{w,k} w_k \le p_k \end{bmatrix}_{k=j+1}^{j+h} \\ & S_{x,j+h} x_{j+h} + S_{w,j+h} w_{j+h} - \underline{1}\,\eta_{j+h} \le s_{j+h} \\ & F_{x,j+h+1} x_{j+1} - \eta_{j+h+1} \le f_{j+h+1}\end{aligned} \qquad (\mathcal{P}_{\text{Corr.}}^{(j+1,r)})$$

As a consequence, the computed objective lower bound for each of the shifted nodes becomes:

$$\underline{\mathcal{O}}_{\text{obj.}}(\mathcal{P}_{\text{BB}}^{(j+1)}) := \rho^{-1}(\underline{\mathcal{O}}_{\text{obj.}}(\mathcal{P}_{\text{BB}}^{(j)}) + \mathcal{O}_{\text{contraction}}) + \mathcal{O}_{\text{expansion}} \tag{4.23}$$

with $\mathcal{O}_{\text{contraction}}$ and $\mathcal{O}_{\text{expansion}}$ defined as in Section 4.4.2.

4.10 Appendix D: Variants for the Shifting Procedure

A More Costly Shifting Procedure (but Sometimes Advantageous)

In general, in the context of mixed-integer linear/quadratic programming, it is arguably beneficial to solve the B&B subproblems using a dual optimization algorithm (e.g. dual-simplex). If this is the case, given the fact that the dual point obtained in 4.4.2 is feasible, running a few (e.g. 1 to 5) iterations of the mentioned algorithm on each of the shifted nodes might largely improve the quality of the obtained lower bounds. Such procedure clearly increases the cost of the shifting step but may dramatically reduce the number of nodes to be solved during the actual B&B step. A trade-off in this direction can be sought on a case-by-case basis.

Using A-Priori Knowledge (Inherited Feature)

In some applications as, for example, Approximate Dynamic Programming ([13]), Optimal Control with Reinforcement Learning ([20]) and MPC with time invariant tail cost, it is reasonable to assume the a-priori knowledge of a $\epsilon_j \in \mathbb{R}$ such that:

$$\begin{array}{lll} F_{j+h}(x) & \leq \min\limits_{w_j} & S_j(x, w_j) + F_{j+1}(A_j x + B_j w_j) + \epsilon_j \\ & \text{s.t.} & \underline{w}_k \leq w_k \leq \bar{w}_k \\ & & C_j(x, w_j) \leq 0 \end{array}$$

Then, as proven in [28], it is possible to substitute the term $\mathcal{O}_{\text{expansion}}$ in (4.11) with ϵ_j, if appropriate, possibly helping the shifting procedure to obtain better lower bounds.

Chapter 5

Tackling Non-Convexity: Disjunctive Outer Approximation for Optimal Control

Although many interesting optimal control problems can be defined using convex functions, there are cases in which deriving a convex model for a dynamical system is not possible. This chapter focuses on extending the OA framework from convex mixed-integer optimal control to a broader class of problems, namely Mixed-Integer Semi-Convex Optimal Control Problems (MISC-OCPs). MISC-OCPs are mixed-integer optimal control problems where the model dynamics and the constraint/objective functions are required to be, respectively, linear and convex solely with respect to the continuous variables of the problem.

In their base form, MISC-OCPs need to be solved using some sort non-convex mixed-integer solver like B&B with added spatial-branching. However, doing so implies a formidable challenge in terms of computational complexity. Luckily, many MISC-OCPs can be reformulated to fall in a different class of problems for which it is possible to devise more efficient solution algorithms. Let us identify such class as the class of Disjunctive-Convex Optimal Control Problems (DC-OCPs).

In order to introduce the concept of DC-OCP, consider a system that can assume a number of different dynamic behaviors according to the selected control strategy. Further, suppose that each of the possible behaviors can be modeled using a linear dynamical equation and a set of convex constraints. For example, an istance of control-based disjunctive behavior can be found in traditional and hybrid cars where the change of dynamic model is due to control choices (e.g. the selection of gear ratio or driving mode).

In this chapter, after giving a formal definition of DC-OCP and presenting a MISC-OCP to DC-OCP translation technique, an efficient solution scheme for DC-OCPs will be presented. The proposed algorithm takes the name of Disjunctive Outer Approximation for Optimal Control (or DOA-OC in short) and consists of a simple strategy allowing us to extend the OA framework to the newly introduced class of problems with minimal effort.

The idea of disjunctive programming it is not new in the literature and many approaches to it exist ([46],[77],[82]). However, to the author knowledge, no efforts have been spent in developing global solvers exploiting the particular structure of disjunctive optimal control problems[1].

5.1 Problem Definition

Let us start by defining a canonical form for a DC-OCP:

$$\begin{aligned} \min_{\tilde{x},\tilde{u},\tilde{v}} \quad & \textstyle\sum_{i=1}^{n_v} \int\limits_{t=0}^{T} \tilde{v}_i(t)\, \mathbf{s}_i(\tilde{x}(t),\tilde{u}(t),t)\,\mathrm{d}t \\ \text{s.t.} \quad & \tilde{x}(0) = \bar{x}_0 \\ & \left.\begin{aligned} & \textstyle\sum_{i=1}^{n_v} \tilde{v}_i(t)\left(\dot{\tilde{x}}(t) - \mathbf{d}_i(\tilde{x}(t),\tilde{u}(t),t)\right) = 0 \\ & \textstyle\sum_{i=1}^{n_v} \tilde{v}_i(t)\, \mathbf{c}_i(\tilde{x}(t),\tilde{u}(t),t) \le 0 \\ & \textstyle\sum_{i=1}^{n_v} \tilde{v}_i(t) = 1; \quad \tilde{v}(t) \in \{0,1\}^{n_v} \end{aligned} \right|_{t\in[0,T]} \end{aligned} \qquad (\mathcal{P}_{\text{DC-OCP}})$$

where:

A-1 $\forall i \in \{1,\cdots,n_v\}$: the functions $\mathbf{s}_i$ and $\mathbf{c}_i$ are convex and continuously-differentiable, while the function $\mathbf{d}_i$ is linear.

Next, applying DMS[2] to each of the models that can be obtained by setting $\tilde{v}_{i^*} \equiv 1$ and $\{\tilde{v}_i \equiv 0\}_{i \neq i^*}$, for some $i^* \in \{1,\cdots,n_v\}$, and reassembling together

---

[1] Although, a quite effective heuristic approach for disjunctive optimal control problems can be found in [75].

the results using the piece-wise constant controls v_k we obtain:

$$\begin{aligned} \min_{x,u,v} \quad & \textstyle\sum_{i=1}^{n_v} \sum_{k=1}^{N} v_{k,i} \mathbf{s}_{k,i}(x_k, u_k) \\ \text{s.t.} \quad & x_0 = \bar{x}_0 \\ & \begin{bmatrix} \sum_{i=1}^{n_v} v_{k,i} \left(x_{k+1} - A_{k,i} x_k - B_{k,i} u_k\right) = 0 \\ \sum_{i=1}^{n_v} v_{k,i}\, \mathbf{c}_{k,i}(x_k, u_k) \le 0 \\ \sum_{i=1}^{n_v} v_{k,i} = 1; \quad v_k \in \{0,1\}^{n_v} \end{bmatrix}_{k=1}^{N} \end{aligned} \tag{$\mathcal{P}_{\text{DC}}$}$$

which we will consider the canonical form of a discrete-time DC-OCP.

Finally, as usual, before applying Outer Approximation to a MINLP it is necessary to reformulate the problem in such a way that all the non-linearity occurs in the constraint set of the problem. In our case, this leads to the following equivalent formulation for $\mathcal{P}_{\text{DC}}$:

$$\begin{aligned} \min_{x,u,v,\eta} \quad & \textstyle\sum_{k=1}^{N-1} \eta_k \\ \text{s.t.} \quad & x_0 = \bar{x}_0 \\ & \begin{bmatrix} \sum_{i=1}^{n_v} v_{k,i} \left(x_{k+1} - A_{k,i} x_k - B_{k,i} u_k\right) = 0 \\ \sum_{i=1}^{n_v} v_{k,i}\, \mathbf{c}_{k,i}(x_k, u_k) \le 0 \\ \sum_{i=1}^{n_v} v_{k,i} \left(\mathbf{s}_{k,i}(x_k, u_k) - \eta_k\right) \le 0 \\ \sum_{i=1}^{n_v} v_{k,i} = 1; \quad v_k \in \{0,1\}^{n_v} \end{bmatrix}_{k=1}^{N} \end{aligned} \tag{$\mathcal{P}_{\text{DC}}$}$$

Clearly, $\mathcal{P}_{\text{DC}}$, is a non-convex problem and thus cannot be solved using any out-off-the-shelf OA-like algorithm. In particular, OA is not applicable to non-convex problems for two reasons:

1. Applying OA to an arbitrary non-convex MINLP possibly results in non-convex problems to be considered in the NLP step. This non-convexity might provoke errors in the assessment of the feasibility of a discrete assignment or the incorrect identification of the best objective value obtainable by fixing the discrete variables of the MINLP to the assignment. Therefore, in the first case, OA might not find an optimal point (or any feasible point) for the MINLP, and, in the second case, OA might fail to converge.
2. Obtaining linearizations of non-convex constraints via first-order Taylor approximation does not necessarily result in relaxations. This might provoke the MILP step of OA to wrongly rule out feasible points.

---

[2] Assuming also the use of a linearity-preserving numerical integration scheme (e.g. Explicit Euler, 4th-Order Runge Kutta, etc.)

In the case of a DC-OCP, the first concern does not apply. In fact, the problems that can be obtained from $\mathcal{P}_{\text{DC}}$ by fixing the discrete controls to a particular assignment are convex. Thus, in this case, the NLP step of OA works as intended. Contrarily, the second concern stands as the first-order Taylor approximations of the constraints of $\mathcal{P}_{\text{DC}}$ are not guaranteed to be relaxations of the constraints they stemmed from. As a consequence, in order to extend the OA framework to DC-OCPs, it is necessary to develop a strategy for obtaining suitable linear relaxations for disjunctive-convex constraints. More specifically, given the structure of OA, we are interested in building linearizations around integer-feasible points.

5.1.1 From Mixed-Integer Semi-Convex to Disjunctive-Convex

As said in the introduction of this chapter, any Mixed-Integer Semi-Convex Optimal Control Problem can be reformulated as a DC-OCP. This subsection briefly presents one strategy to do so.

To this end, define the standard form for a MISC-OCP as follows[3]:

$$
\begin{aligned}
(\tilde{x}^*, \tilde{u}^*, \tilde{v}^*) := \quad & \underset{\tilde{x},\tilde{u},\tilde{v}}{\operatorname{argmin}} \quad \int_{t=0}^{T} \mathbf{s}(\tilde{x}(t), \tilde{u}(t), \tilde{v}(t), t)\, \mathrm{d}t \\
& \text{s.t.} \quad \tilde{x}(0) = \bar{x}_0 \\
& \qquad \left.\begin{aligned} & \dot{\tilde{x}}(t) = \mathbf{d}(\tilde{x}(t), \tilde{u}(t), \tilde{v}(t), t) \\ & \mathbf{p}(\tilde{x}(t), \tilde{u}(t), \tilde{v}(t), t) \le 0 \\ & \tilde{v}(t) \in D_v \subset \mathbb{Z}^{n_v} \end{aligned}\right|_{t \in [0,T]}
\end{aligned}
\qquad (\mathcal{P}_{\text{MISC}})
$$

where the functions $\mathbf{s}$ and $\mathbf{p}$ are jointly convex in their entries: $x(t)$ and $u(t)$, and the function $\mathbf{d}$ is linear in the same entries.

The first step to take is to substitute the integer controls $\tilde{v}$ with a set of binary controls $\tilde{v}'$. This can be done by enumerating all the elements of D_v as follows: $D_v := \{a_i\}_{i=1}^{\#D_v}$, and by assigning a binary control $\tilde{v}'_i$ to each element a_i. Accordingly, the following equivalent problem formulation for $\mathcal{P}_{\text{MISC}}$ is

---

[3] As explained in the previous chapter, with an appropriate reformulation step, $\mathcal{P}_{\text{MISC}}$ can be used to represent also problems having a set of discrete states whose transition functions do not depend on the continuous states.

obtained:

$$
\begin{array}{rll}
(\tilde{x}^*, \tilde{u}^*, \tilde{v}'^*) := & \underset{\tilde{x},\tilde{u},\tilde{v}'}{\operatorname{argmin}} & \int\limits_{t=0}^{T} \mathbf{s}\Big(\tilde{x}(t), \tilde{u}(t), \sum_{i=1}^{\#D_v} \tilde{v}'(t)\, a_i, t\Big)\, \mathrm{d}t \\
& \text{s.t.} & \tilde{x}(0) = \bar{x}_0 \\
& & \left.\begin{array}{l}
\dot{\tilde{x}}(t) = \mathbf{d}\Big(\tilde{x}(t), \tilde{u}(t), \sum_{i=1}^{\#D_v} \tilde{v}'(t)\, a_i, t\Big) \\
\mathbf{p}\Big(\tilde{x}(t), \tilde{u}(t), \sum_{i=1}^{\#D_v} \tilde{v}'(t)\, a_i, t\Big) \leq 0 \\
\sum_{i=1}^{\#D_v} \tilde{v}'_i(t) = 1; \;\; \tilde{v}'(t) \in \{0,1\}^{\#D_v}
\end{array}\right|_{t\in[0,T]}
\end{array}
$$

Then considering the binary nature of the controls v' and their mutual exclusivity encoded in the constraint: $\sum_{i=1}^{\#D_v} \tilde{v}_i(t) = 1$, it is possible to perform an operation called “outer convexification” ([75]) which consists in “factorizing out” the binary controls as follows:

$$
\begin{array}{rll}
(\tilde{x}^*, \tilde{u}^*, \tilde{v}'^*) := & \underset{\tilde{x},\tilde{u},\tilde{v}'}{\operatorname{argmin}} & \sum_{i=1}^{\#D_v} \int\limits_{t=0}^{T} \tilde{v}'(t)\, \mathbf{s}(\tilde{x}(t), \tilde{u}(t), a_i, t)\, \mathrm{d}t \\
& \text{s.t.} & \tilde{x}(0) = \bar{x}_0 \\
& & \left.\begin{array}{l}
\sum_{i=1}^{\#D_v} \tilde{v}'(t)\, \big(\dot{\tilde{x}}(t) - \mathbf{d}(\tilde{x}(t), \tilde{u}(t), a_i, t)\big) = 0 \\
\sum_{i=1}^{\#D_v} \tilde{v}'(t)\, \mathbf{p}(\tilde{x}(t), \tilde{u}(t), a_i, t) \leq 0 \\
\sum_{i=1}^{\#D_v} \tilde{v}'_i(t) = 1; \;\; \tilde{v}'(t) \in \{0,1\}^{\#D_v}
\end{array}\right|_{t\in[0,T]}
\end{array}
$$

At this point, we have obtained an equivalent problem formulation for $\mathcal{P}_{\text{MISC}}$ which, after renaming the problem functions, is identical to $\mathcal{P}_{\text{DC-OCP}}$.

5.2 Preliminaries

This section focuses on the presentation of the two preexisting results that made possible the development and the refinement of the work under discussion. These results are the big-M relaxation method, around which the presented work revolves, and the technique named Linear Bound-Propagation that has given to the work a broader field of application and better performances.

5.2.1 Big-M reformulation

This subsection presents the big-M reformulation method, a well known strategy to decouple the discrete and the continuous part of a disjunctive constraint. The big-M method was introduced for the first time in by Nemhauser and Wolsey in 1988 ([66]) in the context of Integer Optimization and still to date it plays an important role in disjunctive programming and in mixed-integer linear optimization ([82]).

Consider the following mixed-binary constraint:

$$b\,\boldsymbol{\phi}(c) \leq 0 \tag{5.1}$$

where $b \in \{0,1\}$ and $c \in \mathbb{R}$. Clearly, if $b = 0$, the above constraint is automatically satisfied for any value of x, while, if $b = 1$, (5.1) is equivalent to: $\phi(c) \leq 0$. A big-M relaxation of (5.1) is the following constraint:

$$\boldsymbol{\phi}(c) \leq M(1-b) \tag{5.2}$$

where: $\forall c \in \mathbb{R}: \quad M \geq \boldsymbol{\phi}(c)$. It is easy to see that, if $b = 0$ the constraint (5.2) is always satisfied, while, if $b = 1$, (5.2) reduces itself to $\boldsymbol{\phi}(c) \leq 0$. Therefore, on binary-feasible assignments for b the two above constraints are equivalent. However, if we allow b to take value in the interval $[0,1]$, (5.2) results a relaxation of (5.1).

Using big-M reformulations we could in theory reformulate $\mathcal{P}_{\mathrm{DC}}$ as a convex problem. But unfortunately, this would require the computation of an upper bound for the value for each of the problem functions, and given the convexity of $\mathbf{p}_k$ and $\mathbf{s}_k$ this is not an easy problem. Moreover, such approach may require the generation of a great number of potentially useless reformulations. Therefore, at this stage, it is better to refrain from doing so.

5.2.2 Linear Bound-Propagation and Upper Bounds for Linear Expressions

Bound-Propagation techniques attempt to deduce new tight bounds on the value the variables of a problem using the already known bounds and the constraints the variables are involved in. These techniques are quite widespread as they are used in mixed integer programming ([69],[40],[4],[5]), as well as in machine learning ([43],[78]) and other fields.

In this subsection the simplest form of bound propagation will be introduced: Linear (Single-Constraint) Bound-Propagation (LBP). Consider an array of bounded variables: $\underline{x} \leq x \leq \bar{x}$ and a linear constraint: $\sum_{i=1}^{n_x} a_i x_i \leq b$. Then, for any $\hat{\imath}$ such that $a_{\hat{\imath}} \neq 0$, the following results stand:

$$\text{if } a_{\hat{\imath}} > 0: \begin{cases} x_{\hat{\imath}} \leq \dfrac{b - \sum_{i \neq \hat{\imath}} (\max(a_i, 0)\,\bar{x}_i + \min(a_i, 0)\,\underline{x}_i)}{a_{\hat{\imath}}} \\ x_{\hat{\imath}} \geq \dfrac{b - \sum_{i \neq \hat{\imath}} (\max(a_i, 0)\,\underline{x}_i + \min(a_i, 0)\,\bar{x}_i)}{a_{\hat{\imath}}} \end{cases} \tag{5.3}$$

$$\text{if } a_{\hat{i}} < 0 : \begin{cases} x_{\hat{i}} \geq \dfrac{b - \sum_{i \neq \hat{i}} (\max(a_i, 0)\, \bar{x}_i + \min(a_i, 0)\, \underline{x}_i)}{a_{\hat{i}}} \\ x_{\hat{i}} \leq \dfrac{b - \sum_{i \neq \hat{i}} (\max(a_i, 0)\, \underline{x}_i + \min(a_i, 0)\, \bar{x}_i)}{a_{\hat{i}}} \end{cases} \tag{5.4}$$

Moreover, it holds:

$$\sum_{i=1}^{n_x} a_i x_i \leq \sum_{i=1}^{n_x} (\max(a_i, 0)\, \bar{x}_i + \min(a_i, 0)\, \underline{x}_i) =: b' \tag{5.5}$$

so that $\sum_{i=1}^{n_x} a_i x_i \leq b'$ is a tighter-but-equivalent formulation for $\sum_{i=1}^{n_x} a_i x_i \leq b$.

The above results can be used to tighten the bounds for the variables and the constraint of an optimization problem in such a way that each bound update for a variable/constraint may trigger a cascade of updates for other variables/constraints. Such procedure is what we call LBP. In general, LBP does not terminate in a finite number of steps, but, properly sorting the list of the next variables and constraints to check, it can be stopped whenever the obtained bound updates become small enough to be considered irrelevant.

Now, LPB in combination with B&B provides two important advantages:

- Tightening the bounds of the variables and the constraints of the system provokes the B&B subproblems to yield tighter lower-bounds.
- Using bound propagation on the discrete variables of the problem might lead to the elimination of large part of the discrete assignment space.

Then, since the above advantages can help to drastically reduce the computational complexity of B&B, LBP is a popular technique in the context of mixed-integer programming and almost every MILP/MIQP solver implements its own version of linear bound propagation ([90],[91]). As a consequence, in the OA context, it is quite safe to assume the availability of a LBP procedure.

Finally, note that (5.5) gives an example of how easy it is to compute an upper-bound for a linear expression starting from the bounds of the involved variables. Such observation is crucial for the remainder of the current chapter.

5.3 Linear Relaxations for Disjunctive-Convex Constraints

As already mentioned, the missing block for successfully using OA on disjunctive-convex problems is a method for obtaining linear relaxations for disjunctive-convex constraints around arbitrary integer-feasible points. This section will focus on the presentation of a functional methodology in this direction.

Now, for simplicity, only one constraint of $\mathcal{P}_{\mathrm{DC}}$ (and therefore only one time-step) will be considered. Denote the form of the constraint as:

$$\sum_{i=1}^{n_v} v_{k,i}\phi_{k,i}(x_k, u_k, \eta_k) \leq 0 \tag{5.6}$$

The methodology to be presented is better explained in two steps: in the first step, only the "continuous parts": $\{\phi_{k,i}(x_k, u_k, \eta_k)\}_{i=1}^{n_v}$, of the above constraint will be linearized around a generic point $(\hat{x}_k, \hat{u}_k, \hat{\eta}_k)$ obtaining a bilinear relaxation. In the second step, the resulting bilinear relaxation will be reformulated using the big-M method around a number of discrete assignments $\hat{v}_k$.

5.3.1 Relaxing the Continuous Part of the Constraint

$\forall i \in \{1, \cdots, n_v\}$, the convexity of $\phi_{k,i}$ entails:

$$\phi_{k,i}(x_k, u_k, \eta_k) \geq \phi_{k,i}(\hat{x}_k, \hat{u}_k, \hat{\eta}_k) + \frac{\partial \phi_{k,i}}{\partial \cdot}(\hat{x}_k, \hat{u}_k, \hat{\eta}_k) \begin{bmatrix} x_k - \hat{x}_k \\ u_k - \hat{u}_k \\ \eta_k - \hat{\eta}_k \end{bmatrix} \tag{5.7}$$

for any given point: $(\hat{x}_k, \hat{u}_k, \hat{\eta}_k)$. Now, thanks to the binary nature of the $v_{k,i}$ it holds:

$$\sum_{i=1}^{n_v} v_{k,i}\phi_{k,i}(x_k, u_k, \eta_k) \geq \sum_{i=1}^{n_v} v_{k,i} \left(\phi_{k,i}(\hat{x}_k, \hat{u}_k, \hat{\eta}_k) + \frac{\partial \phi_{k,i}}{\partial \cdot}(\hat{x}_k, \hat{u}_k, \hat{\eta}_k) \begin{bmatrix} x_k - \hat{x}_k \\ u_k - \hat{u}_k \\ \eta_k - \hat{\eta}_k \end{bmatrix} \right)$$

Therefore, the constraint:

$$\sum_{i=1}^{n_v} v_{k,i} \left(\phi_{k,i}(\hat{x}_k, \hat{u}_k, \hat{\eta}_k) + \frac{\partial \phi_{k,i}}{\partial \cdot}(\hat{x}_k, \hat{u}_k, \hat{\eta}_k) \begin{bmatrix} x_k - \hat{x}_k \\ u_k - \hat{u}_k \\ \eta_k - \hat{\eta}_k \end{bmatrix} \right) \leq 0 \tag{5.8}$$

is a bilinear relaxation of (5.6).

5.3.2 Relaxing the Discrete Part of the Constraint

Before diving in the explaination of the this second step, let us postulate the knowledge of upper and lower bounds for x_k and u_k together with a lower bound for the slack variable η_k. In the later sections, such assumption will be discussed in more detail.

The next step, is to obtain a linear relaxation of (5.8), and therefore of (5.6), around any suitable binary assignment $\hat{v}_{k,i}$. To this end, consider the following definition:

$$\Delta(\hat{v}_k, v_k) := \sum_{i=1}^{n_v} v_{k,i} - 2\hat{v}_{k,i}v_{k,i} + \hat{v}_{k,i} \tag{5.9}$$

and observe that the value of $\Delta(\hat{v}_k, v_k)$ is zero if and only if $v_k = \hat{v}_k$ and greater than one otherwise. Now, denote with $\hat{v}_k^{(j)}$ the assignment where $\hat{v}_{k,j}^{(j)} = 1$ and $\hat{v}_{k,i}^{(j)} = 0$, $\forall i \neq j$. Then, assuming the knowledge of a suitable set of parameters $\{M^{(j)}\}_{j=1}^{n_v}$, it is possible to use the big-M reformulation technique on (5.8) to obtain the following set of constraints:

$$\left\{ \phi_{k,j}(\hat{x}_k, \hat{u}_k, \hat{\eta}_k) + \frac{\partial \phi_{k,j}}{\partial \cdot}(\hat{x}_k, \hat{u}_k, \hat{\eta}_k) \begin{bmatrix} x_k - \hat{x}_k \\ u_k - \hat{u}_k \\ \eta_k - \hat{\eta}_k \end{bmatrix} \leq M^{(j)} \Delta(\hat{v}_k^{(j)}, v_k) \right\}_{j=1}^{n_v} \tag{5.10}$$

where, each of the above constraints is a linear relaxation of (5.6) around a point: $(\hat{x}_k, \hat{u}_k, \hat{v}_k^{(j)}, \hat{\eta}_k)$.

Now, as shown in (5.5), thanks to the linearity of the continuous parts of (5.8) the computation of the parameters $M^{(j)}$ is trivial. However, it is not necessary to use all of the above constraints to ensure the convergence of the OA scheme to be built. In fact, the convergence can be obtained by just using the assignment $\hat{v}_k^{(j)}$ that matches the current discrete guess in OA. Nevertheless, it might be beneficial to add more than one linearization and the selection of which linearizations to consider is matter of compromise. In general, a good rule of thub is to generate linearizations only for those constraints where: $\phi_{k,j}(\hat{x}_k, \hat{u}_k, \hat{\eta}_k) > 0$.

5.4 Disjunctive Outer Approximation for Optimal Control

Now that a strategy to obtain suitable linearizations for disjunctive-convex constraints is in place, it is possible to build an OA-like scheme capable of efficiently solving $\mathcal{P}_{\mathrm{DC}}$. Denote such scheme with DOA-OC (Disjunctive Outer Approximation). The current section is subdivided in two parts: in the first part, the structure of DOA-OC is discussed, and the second part contains few comments on the convergence properties of the proposed algorithm.

5.4.1 The Algorithmic Structure

For the sake of clarity, the algorithm is hereby presented as consisting of three steps. The first two steps can be considered pre-processing routines as they serve the purpose of collecting data to be used in the third step, where the actual optimization takes place.

Before diving into the functioning of DOA-OC, it is necessary to make two additional assumptions on the structure of $\mathcal{P}_{\text{DC}}$:

A-2 The continuous controls are fully bounded, i.e. the path constraints entail:

$$\forall k : \ \exists \underline{u}_k, \bar{u}_k \text{ s.t. } \underline{u}_k \leq u_k \leq \bar{u}_k \text{ and } \left\|\underline{u}_k\right\|_\infty, \left\|\bar{u}_k\right\|_\infty < \infty$$

A-3 A lower bound for the value of each $\mathbf{s}_{k,i}(x_k, u_k)$ on the feasible space of $\mathcal{P}_{\text{DC}}$ is a priori known.

However, these assumptions are not very restrictive as, in most realistic scenarios, the control action is naturally bounded by the physical limits of the actuators (e.g. the maximum torque of an electric motor), and $\mathbf{s}_{k,i}(x_k, u_k)$ is convex.

The remainder of this section presents DOA-OC assuming some degree of familiarity of the reader with the classical OA approach. However, for ease of understanding, a schematic representation of DOA-OC can be found in Fig. 5.1.

Step 1: big-M relaxation for the dynamic constraints

DOA-OC starts by defining a complete set of big-M reformulations for the dynamic constraints of $\mathcal{P}_{\text{DC}}$ following a chronological order[4]: in the first dynamic equation (which is disjunctive-linear by construction) the state is fixed by the initial-state constraint. Thus, it is immediately possible to apply the big-M method to obtain a linear constraint. Next, a LPB step can be run on each of the continuous parts of the fist dynamic constraints to obtain a set of bounds: $\{(\underline{x}_{1,i}, \bar{x}_{1,i})\}_{i=1}^{n_v}$. Then the bounds for the first state variables can be deduced as follows:

$$\underline{x}_1 := \min_i \{\underline{x}_{1,i}\} \text{ and } \bar{x}_1 := \max_i \{\bar{x}_{1,i}\} \tag{5.11}$$

Finally, the process can be repeated for the second dynamic constraints, then the third, and so on until the $(N-1)$-th time-step.

---

[4] For employing the developed linearization technique it is necessary to have finite upper and lower bounds on the state variables x_k. This step serves principally the purpose of generating such bounds.

Remark 1: The above procedure does not take into account the constraints of the problem therefore the bounds on the state variables can grow unreasonably large with the time-steps. A solution to this issue is to define general bounds for the state variables if possible. In fact, in such case the bounds found by the LBP procedure stay at most as loose as the given general bounds.

Remark 2: The computational cost of this first step grows linearly with the number of time-steps considered and in practice it is quite modest. Moreover, if DOA-OC is used in an MPC fashion, in each iteration, it is only necessary to consider the state variables and the dynamic constraints relative to the new portion of the MPC subproblem.

Step 2: obtaining an initial binary assignment guess

Next, similarly to OA, DOA-OC necessitates an initial binary assignment to begin with. Such assignment can be arbitrary but the scheme may benefit from a good initial guess. In this regard, a fast and convenient strategy consists in solving the continuous relaxation of the problem at hand and then obtaining a binary assignment via a procedure called Sum Up Rounding ([75]).

Step 3: Outer Approximation

At this point it is possible to start solving $\mathcal{P}_{\mathrm{DC}}$ using our specialized OA-like scheme. The algorithm is almost identical to the classical OA with the sole difference being that the linearization for the constraints are obtained as explained in Section 5.3 rather than via simple first-order Taylor approximation.

Remark: The considered problem definition, $\mathcal{P}_{\mathrm{DC}}$, features only disjunctive-convex constraints. However, it is possible to imagine problems in which also generally convex constraints appear. In this case, the algorithm can easily be extended in such a way it uses Taylor approximations on the convex constraints and big-M relaxations on the disjunctive-convex ones.

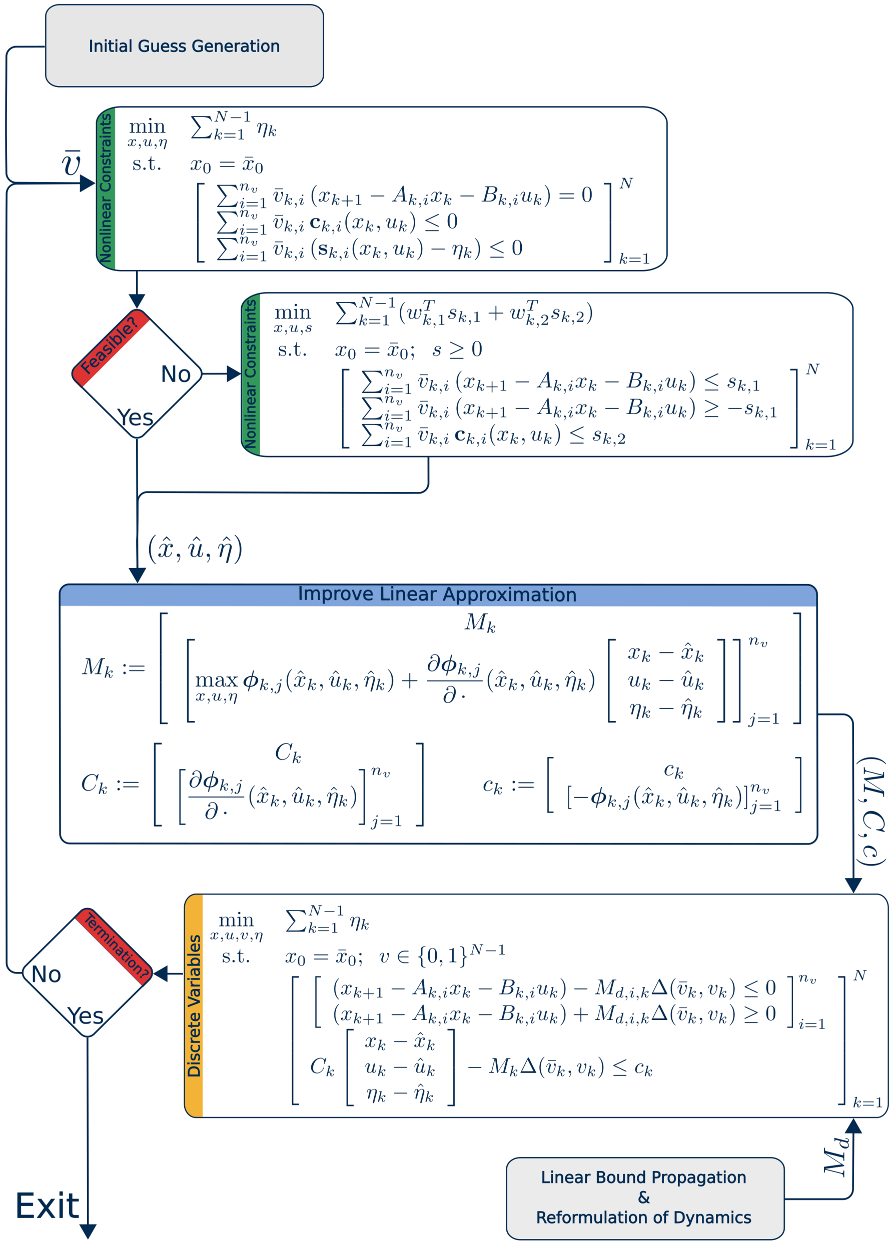

Initial Guess Generation
$\bar{v}$
Nonlinear Constraints
$\min_{x,u,\eta} \sum_{k=1}^{N-1} \eta_k$
s.t. $x_0 = \bar{x}_0$
$\begin{bmatrix} \sum_{i=1}^{n_v} \bar{v}_{k,i} \left(x_{k+1} - A_{k,i}x_k - B_{k,i}u_k\right) = 0 \\ \sum_{i=1}^{n_v} \bar{v}_{k,i}\, \mathbf{c}_{k,i}(x_k, u_k) \leq 0 \\ \sum_{i=1}^{n_v} \bar{v}_{k,i} \left(\mathbf{s}_{k,i}(x_k, u_k) - \eta_k\right) \leq 0 \end{bmatrix}_{k=1}^{N}$
Feasible?
No
Yes
Nonlinear Constraints
$\min_{x,u,s} \sum_{k=1}^{N-1} (w_{k,1}^T s_{k,1} + w_{k,2}^T s_{k,2})$
s.t. $x_0 = \bar{x}_0; \quad s \geq 0$
$\begin{bmatrix} \sum_{i=1}^{n_v} \bar{v}_{k,i} \left(x_{k+1} - A_{k,i}x_k - B_{k,i}u_k\right) \leq s_{k,1} \\ \sum_{i=1}^{n_v} \bar{v}_{k,i} \left(x_{k+1} - A_{k,i}x_k - B_{k,i}u_k\right) \geq -s_{k,1} \\ \sum_{i=1}^{n_v} \bar{v}_{k,i}\, \mathbf{c}_{k,i}(x_k, u_k) \leq s_{k,2} \end{bmatrix}_{k=1}^{N}$
$(\hat{x}, \hat{u}, \hat{\eta})$
Improve Linear Approximation
$M_k := \begin{bmatrix} \overset{M_k}{\left[\max_{x,u,\eta} \phi_{k,j}(\hat{x}_k, \hat{u}_k, \hat{\eta}_k) + \frac{\partial \phi_{k,j}}{\partial \cdot}(\hat{x}_k, \hat{u}_k, \hat{\eta}_k) \begin{bmatrix} x_k - \hat{x}_k \\ u_k - \hat{u}_k \\ \eta_k - \hat{\eta}_k \end{bmatrix} \right]_{j=1}^{n_v}} \end{bmatrix}$
$C_k := \begin{bmatrix} \overset{C_k}{\left[\frac{\partial \phi_{k,j}}{\partial \cdot}(\hat{x}_k, \hat{u}_k, \hat{\eta}_k) \right]_{j=1}^{n_v}} \end{bmatrix}$
$c_k := \begin{bmatrix} \overset{c_k}{\left[-\phi_{k,j}(\hat{x}_k, \hat{u}_k, \hat{\eta}_k) \right]_{j=1}^{n_v}} \end{bmatrix}$
(M, C, c)
Termination?
No
Yes
Discrete Variables
$\min_{x,u,v,\eta} \sum_{k=1}^{N-1} \eta_k$
s.t. $x_0 = \bar{x}_0; \quad v \in \{0,1\}^{N-1}$
$\begin{bmatrix} \begin{bmatrix} (x_{k+1} - A_{k,i}x_k - B_{k,i}u_k) - M_{d,i,k}\Delta(\bar{v}_k, v_k) \leq 0 \\ (x_{k+1} - A_{k,i}x_k - B_{k,i}u_k) + M_{d,i,k}\Delta(\bar{v}_k, v_k) \geq 0 \end{bmatrix}_{i=1}^{n_v} \\ C_k \begin{bmatrix} x_k - \hat{x}_k \\ u_k - \hat{u}_k \\ \eta_k - \hat{\eta}_k \end{bmatrix} - M_k\Delta(\bar{v}_k, v_k) \leq c_k \end{bmatrix}_{k=1}^{N}$
M_d
Linear Bound Propagation
&
Reformulation of Dynamics
Exit

Figure 5.1: Schematic representation of Disjunctive OA for Optimal Control

5.4.2 Convergence Properties

The third step of DOA-OC being almost identical to OA, the new algorithm inherits its convergence properties from the classical one. In fact, if the binary variables of the problem at hand are fixed to a specific assignment $\bar{v}$, the linearizations obtained via big-M relaxation around such assignment become identical to the linearizations that could be obtained via Taylor approximation around the same assignment. This makes possible to prove the finite convergence of DOA-OC using the same arguments used for OA. The interested reader may find in [59] a complete proof of the finite convergence of OA.

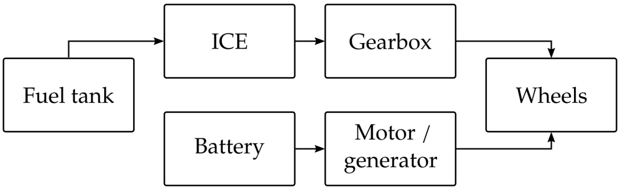
5.5 Numerical Validation

In this section a case study regarding the control of a hybrid electric vehicle is used to analyze the performances of Disjunctive OA. The current section is subdivided in three parts. In the first two, the chosen case study is introduced, and, in the last part, the results of a series of validating numerical experiments are presented.

5.5.1 The Case of Study

The model we will use for showcasing the performaces of the DOA-OC scheme is similar to the one used in the last section for MIRT-OC. In particular, we will again consider a Parallel Hybrid Vehicle driven along the FTP-75 drive cycle. However, the current model differs from the last as it features a different list of components and it is obtained using Torque instead of Power as main control signal. The latter choice has two consequences: firstly, it allows for a more accurate modeling of the start-and-stop phases of the vehicle, secondly, it generates a disjunctive convex model. In the remainder of this section, each of the components of the considered model will be presented in detail.

Figure 5.2: Parallel Hybrid Electric Vehicle Topology (Rep.)



Vehicle Body

The HEV considered in this case study medium-sized consumer car. The longitudinal dynamics of the car was modeled under the assumption of perfect knowledge of the vehicle speed profile: $\mathrm{v}^{\mathrm{cycle}}(t)$. The result reads as follows:

$$\begin{aligned}\mathrm{F}^{\mathrm{req}}(t)\ [\mathrm{N}] := \quad & \mathrm{M}\cdot\dot{\mathrm{v}}^{\mathrm{cycle}}(t) + \mathrm{F}_{\mathrm{friction}}\cdot\chi^{+}(\mathrm{v}^{\mathrm{cycle}}(t)) + \\ & 0.5\cdot\rho_{\mathrm{air}}\cdot\mathrm{C}_{\mathrm{drag}}\cdot\mathrm{A}_{\mathrm{frontal}}\cdot(\mathrm{v}^{\mathrm{cycle}}(t))^2\end{aligned} \tag{5.12}$$

where $\chi^{+}(x)$ is a function whose value is one if $x > 0$ and zero otherwise, and the following parameters were used:

Parameter	M	$\mathrm{F}_{\mathrm{friction}}$	ρ_{air}	$\mathrm{C}_{\mathrm{drag}}$	$\mathrm{A}_{\mathrm{frontal}}$
Value	1270.0	0.005	1.225	0.35	2
Units	kg	N	$\mathrm{kg/m^3}$	-	$\mathrm{m^2}$

Next, using (5.12) and assuming a wheel radius of $0.285m$, we obtain:

$$\begin{cases} \mathrm{T}^{\mathrm{req}}(t)\ [\mathrm{W}] := \mathrm{F}^{\mathrm{req}}(t)\cdot\mathrm{R}^{\mathrm{wheels}} \\ \omega^{\mathrm{cycle}}(t)\ [\mathrm{rad/s}] := \mathrm{v}^{\mathrm{cycle}}(t)/\mathrm{R}^{\mathrm{wheels}} \end{cases} \tag{5.13}$$

Battery

The battery pack selected for this new example is identical to the one used in the last chapter. Therefore, let us limit ourselves to recall its characteristics for ease of reference:

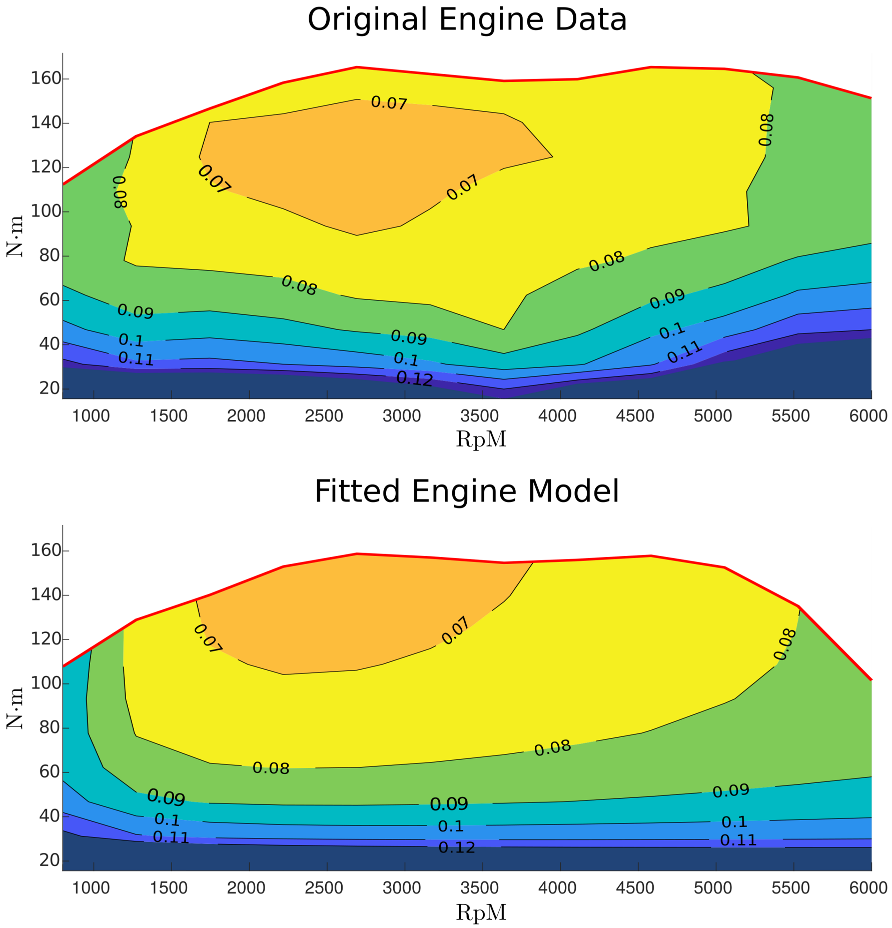
$$\begin{aligned}&\dot{\mathrm{E}}\ [\mathrm{W}] = \mathrm{P}^{\mathrm{BAT}} \\ &\boldsymbol{Pout}^{\mathrm{BAT}}(\mathrm{P}^{\mathrm{BAT}},\mathrm{E})\ [\mathrm{W}] := \mathrm{P}^{\mathrm{BAT}} - \frac{R\cdot C\cdot {\mathrm{P}^{\mathrm{BAT}}}^2}{2\cdot\mathrm{E} - 132\cdot\mathrm{U}_0^2\cdot C} \\ &132\cdot\mathrm{E}_{\mathrm{cell,max}} \le \mathrm{E}\ [\mathrm{J}] \le 132\cdot\mathrm{E}_{\mathrm{cell,max}}\end{aligned} \tag{5.14}$$

where:

Parameter	C	R	U_0	$\mathrm{E}_{\mathrm{cell,max}}$
Value	51782	0.01	3.3	27854
Units	Farad	Ω	V	J

Internal Combustion Engine

For the considered HEV a 80kW Internal Combustion Engine (ICE) was selected. The data were once again taken from a set of real measurements that can be found in [94]. A depiction of the measurements versus the fitted model can be found in Figure 5.3.



Original Engine Data

Fitted Engine Model

Figure 5.3: Engine: Fitted Model VS Original Data (Instant. Fuel Consumption)

The data were fitted using MATLAB curve fitting toolbox obtaining the following fuel consumption map:

$$\begin{aligned}\boldsymbol{dFin}^{\mathrm{ICE}}(\mathrm{T}^{\mathrm{ICE}},\omega^{\mathrm{ICE}}) := \quad & (a_2\omega\mathrm{n}^2 + a_1\omega\mathrm{n} + a_0)\cdot\mathrm{Tn}^2 + \\ & (b_2\omega\mathrm{n}^2 + b_1\omega\mathrm{n} + b_0)\cdot\mathrm{Tn} + \\ & (c_2\omega\mathrm{n}^2 + c_1\omega\mathrm{n} + c_0)\end{aligned}$$

where:

$$\mathrm{Tn} := \frac{\mathrm{T}^{\mathrm{ICE}}}{171.6} \quad \text{and} \quad \omega\mathrm{n} := \frac{\omega^{\mathrm{ICE}}}{628.32}$$

and:

l	a_l	b_l	c_l
0	0.7665	-0.1018	0.0348
1	-3.1445	5.3637	1.0270
2	3.7249	1.5618	-0.2544

Similarly, the measurements where used to model the engine speed-dependant maximum torque. The resulting fitted function is:

$$\begin{aligned}\boldsymbol{T}^{\mathrm{ICE}}_{\mathrm{max}}(\omega^{\mathrm{ICE}}) := \quad & -543.4\omega\mathrm{n}^8 + 2535\omega\mathrm{n}^7 - 4911\omega\mathrm{n}^6 + 5100\omega\mathrm{n}^5 - 3064\omega\mathrm{n}^4 + \\ & 1075\omega\mathrm{n}^3 - 213.5\omega\mathrm{n}^2 + 22.77\omega\mathrm{n} - 0.2537\end{aligned}$$

Electric Motor/Generator

For our HEV model a 35kW Electric Motor/Generator (EMG) was chosen. The original measured data, along with the results of the fitting, can be found in Figure 5.4. The obtained input-power map reads as follows:

$$\begin{aligned}\boldsymbol{Pin}^{\mathrm{EMG}}(\mathrm{T}^{\mathrm{EMG}},\omega^{\mathrm{EMG}}) := \quad & 35e3\cdot(a_3\omega\mathrm{n}^3 + a_2\omega\mathrm{n}^2 + a_1\omega\mathrm{n})\cdot\mathrm{Tn}^2 + \\ & 35e3\cdot(b_3\omega\mathrm{n}^3 + b_2\omega\mathrm{n}^2 + b_1\omega\mathrm{n})\cdot\mathrm{Tn} + \\ & 35e3\cdot(c_3\omega\mathrm{n}^3 + c_2\omega\mathrm{n}^2 + c_1\omega\mathrm{n})\end{aligned}$$

where:

$$\mathrm{T}_n := \frac{\mathrm{T}^{\mathrm{EMG}}}{126.5310} \quad \text{and} \quad \omega_n := \frac{\omega^{\mathrm{EMG}}}{1047.2}$$

and:

l	a_l	b_l	c_l
1	1.4060	3.786	0.1961
2	-5.6970	0.000	-0.3766
3	9.1400	0.000	0.1757

For the maximum-torque curve the following function was obtained:

$$\boldsymbol{T}^{\mathrm{EMG}}_{\mathrm{max}}(\omega^{\mathrm{EMG}})\ [\mathrm{W}] := \min(126.53, -65.66\omega\mathrm{n}^3 + 276.60\omega\mathrm{n}^3 - 369.34\omega\mathrm{n}^2 + 192.58)$$

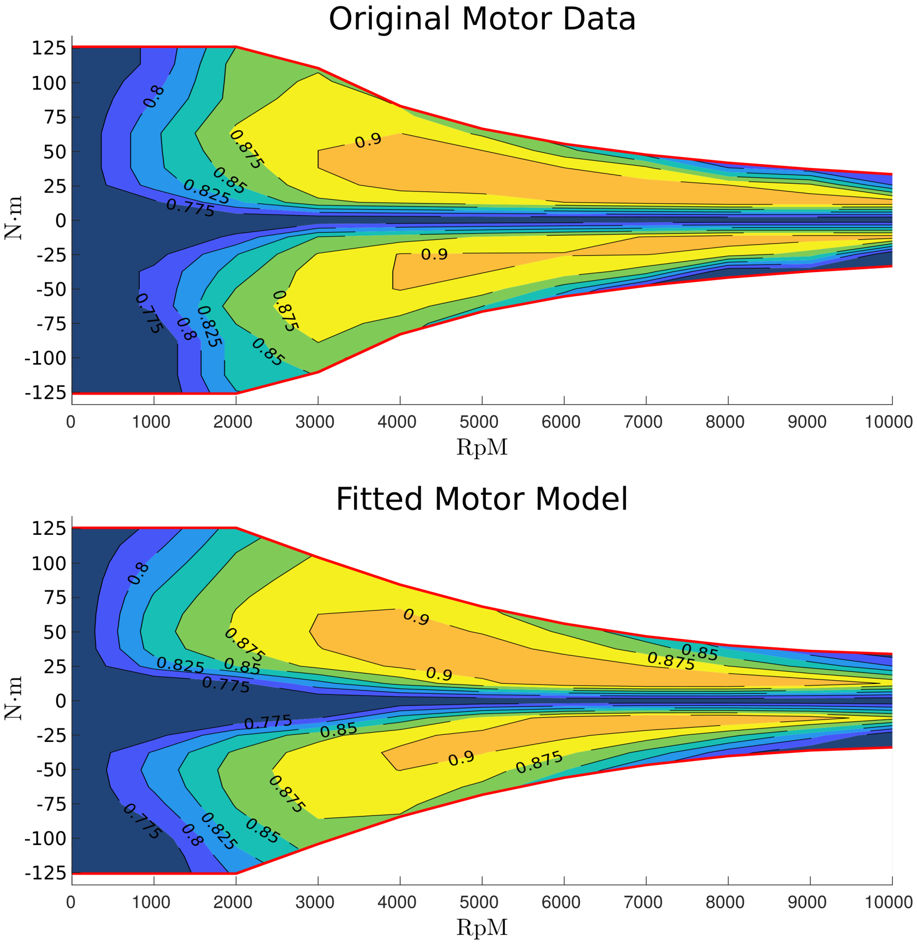


Figure 5.4: Electric Motor: Fitted Model VS Original Data (Efficiency)

Gearbox and Torque Coupling

Once again, the available ratios in the gearbox have been computed considering a reference engine speed of 2500RpM obtaining:

Gear	1st	2nd	3rd	4th	5th
Ref. Speed (km/h)	15	30	55	85	115
Ratio	17.72	8.86	4.83	3.13	2.31

Similarly, the fixed EMG-to-wheels ratio, was chosen as follows:

$$\mathrm{R}^{\mathrm{EMG}} := 3.544 \text{ (i.e. 2000RpM at 60km/h)}$$

Discrete Controls

The following binary controls were considered:

Off_k : $\mathrm{Off}_k = 1$ means that the engine is off and the gearbox is in neutral.

N_k : $\mathrm{N}_k = 1$ means that the engine is on and the gearbox is in neutral.

$[\mathrm{G}_{i,k}]_{i=1}^{5}$: $\mathrm{G}_{i,k} = 1$ means that the engine is on and the i-th gear is in use.

Further, the mutual exclusivity of the binary controls is imposed via:

$$\mathrm{Off}_k + \mathrm{N}_k + \sum_{i=1}^{5} \mathrm{G}_{i,k} = 1$$

Finally, one auxiliary state variable, pOff, was defined for keeping track of the past engine state.

5.5.2 The Resulting Model

According to the MPC structure, we will assume that, at any time, it is possible to predict with reasonable accuracy the future driver's inputs up to the next h seconds. Then, considering a time-discretization resolution of one second and h seconds of prediction window, the following problem formulation was obtained:

$$
\begin{array}{ll}
\min & \displaystyle\sum_{k=j}^{j+h-1}\left(\mathrm{dF}_k + K^{\mathrm{E2F}}(\bar{\mathrm{E}}_j)\cdot\mathrm{P}_k^{\mathrm{BAT}}\right)\\
\text{s.t.} & \text{\# Initial State}\\
& [\mathrm{F}_j,\mathrm{E}_j,\mathrm{pOff}_j]^T := [\bar{\mathrm{F}}_j,\bar{\mathrm{E}}_j,\mathrm{p\bar{O}ff}_j]^T\\
& \left[\begin{array}{l}
\text{\# Dynamic Constraints}\\
\mathrm{F}_{k+1} = -\mathrm{dF}_k + \mathrm{F}_k\\
\mathrm{E}_{k+1} = -\mathrm{P}^{\mathrm{BAT}} + \mathrm{E}_k\\
\mathrm{pOff}_{k+1} = \mathrm{Off}_k\\
\text{\# Integer Requirements and SoS1 Constraints}\\
\mathrm{Off}_k, \mathrm{N}_k \in \{0,1\};\quad [\mathrm{G}_{i,k} \in \{0,1\}]_{i=1}^{5}\\
\mathrm{Off}_k + \mathrm{N}_k + \sum_{i=1}^{5}\mathrm{G}_{i,k} = 1\\
\mathrm{pOff}_k - \mathrm{Off}_k \le \mathrm{N}_k \text{ (The engine can start only in neutral gear)}\\
\text{\#Battery and Electric Motor/Generator}\\
\mathrm{E}^{\min} \le \mathrm{E} \le \mathrm{E}^{\max}\\
\boldsymbol{Pin}^{\mathrm{EMG}}(\mathrm{T}_k^{\mathrm{EMG}}/\mathrm{R}^{\mathrm{EMG}}, \omega_k^{\mathrm{cycle}}\mathrm{R}^{\mathrm{EMG}}) \le \boldsymbol{Pout}^{\mathrm{BAT}}(\mathrm{P}_k^{\mathrm{BAT}}, \mathrm{E}_k)\\
\left|\mathrm{T}_k^{\mathrm{EMG}}\right| \le \boldsymbol{T}_{\max}^{\mathrm{EMG}}(\mathrm{R}^{\mathrm{EMG}}\omega_k^{\mathrm{cycle}})\cdot\mathrm{R}^{\mathrm{EMG}}\\
\text{\# Internal Combustion Engine}\\
\sum_{i=1}^{5}\mathrm{G}_{i,k}\cdot(\omega_{\min}^{\mathrm{ICE}} - \mathrm{R}_i^{\mathrm{GBX}}\omega_k^{\mathrm{cycle}}) \le 0\\
\sum_{i=1}^{5}\mathrm{G}_{i,k}\cdot(\mathrm{R}_i^{\mathrm{GBX}}\omega_k^{\mathrm{cycle}} - \omega_{\max}^{\mathrm{ICE}}) \le 0\\
\mathrm{T}^{\mathrm{ICE}} \le \sum_{i=1}^{5}\mathrm{G}_{i,k}\cdot\boldsymbol{T}_{\max}^{\mathrm{ICE}}(\mathrm{R}_i^{\mathrm{GBX}}\omega_k^{\mathrm{cycle}})\cdot\mathrm{R}_i^{\mathrm{GBX}}\\
\mathrm{dF} \ge \sum_{i=1}^{5}\mathrm{G}_{i,k}\cdot\boldsymbol{dFin}^{\mathrm{ICE}}(\mathrm{T}^{\mathrm{ICE}}/\mathrm{R}_i^{\mathrm{GBX}}, \omega_k^{\mathrm{cycle}}\mathrm{R}_i^{\mathrm{GBX}}) +\\
\qquad\qquad \sum_{i=1}^{5}\mathrm{G}_{i,k}\cdot(1 - prevG_{i,k})\cdot f_{\mathrm{shift}} +\\
\qquad\qquad\quad \mathrm{N}_k\cdot(f_{\mathrm{idle}} + \mathrm{pOff}_k\cdot f_{\mathrm{start}}/2) +\\
\qquad\qquad\quad \mathrm{Off}_k\cdot((1 - \mathrm{pOff}_k)\cdot f_{\mathrm{start}}/2)\\
\text{\# Torque Requirement}\\
\mathrm{T}_k^{\mathrm{req}} \le \mathrm{T}_k^{\mathrm{EMG}} + \mathrm{T}_k^{\mathrm{ICE}}
\end{array}\right]_{k=j}^{j+h-1}
\end{array}
\qquad (\mathcal{P}_{\mathrm{MPC}}^{(j)})
$$

where:

- $\bar{\mathrm{F}}_j$, $\bar{\mathrm{E}}_j$ and $\mathrm{p\bar{O}ff}_j$ are values extracted from the solution of $\mathcal{P}_{\mathrm{MPC}}^{(j-1)}$.
- The term: $K^{\mathrm{E2F}}(\bar{\mathrm{E}}_j)$ gives a different relative importance to the consumption of electric energy with respect to fuel consumption depending on the state of the vehicle and on the optimization goals. The considered value for $K^{\mathrm{E2F}}(\bar{\mathrm{E}}_j)$ was selected empirically and reads as follows:

$$K^{\mathrm{E2F}}(\bar{\mathrm{E}}_j) := 1.2\frac{\mathrm{E}_{\max} - \bar{\mathrm{E}}_j}{\mathrm{E}_{\max}}$$

- The constants f_{start}, f_{shift} and f_{idle} respectively represent the fuel cost of turning the ICE on, the fuel cost of changing gear and the engine idle fuel consumption.
- In the proposed model, the constraint: $\mathrm{Off}_k + \mathrm{N}_k \geq \mathrm{pOff}_k$, is used in order to force the optimization to assume a delay between the moment in which the engine starts to turn on and the moment in which it is actually available for use. Additionally, the fuel cost of turning the ICE on is split between the actual turning on phase: $\mathrm{N}_k \cdot (f_{\mathrm{idle}} + \mathrm{pOff}_k \cdot f_{\mathrm{start}}/2)$, and the turning off phase: $\mathrm{Off}_k \cdot ((1 - \mathrm{pOff}_k) \cdot f_{\mathrm{start}}/2)$. This two characteristics are meant to prevent the MPC strategy from turning the engine on and off unnecessarily often.

A sample solution, obtained considering 6 seconds of forward prediction, is depicted in Figure 5.5.

5.5.3 Numerical Results

This section gives a sample of the tests performed on the proposed algorithm. The remainder of the current subsection consists of two parts. In the first part, the performances of Disjunctive OA are evaluated at the variation of the size of the considered MPC subproblems then, briefly, the differences among the obtained solutions are explained. In the second part, Disjunctive OA will be tested against a state-of-the-art mixed-integer non-convex solver: Scip [41] in order to assess whether or not the developed method shows a performance edge over the preexisting techniques.

Remark: All the tests were performed on a personal computer equipped with a AMD Ryzen 2700 processor and 16gb of RAM. Only a single processor core was used in order to allow for fairer comparisons. Furter, in Disjunctive OA, the solvers Ipopt [83] and Gurobi [91] have been used for, respectively, the solution of non-linear and mixed-integer linear problems.

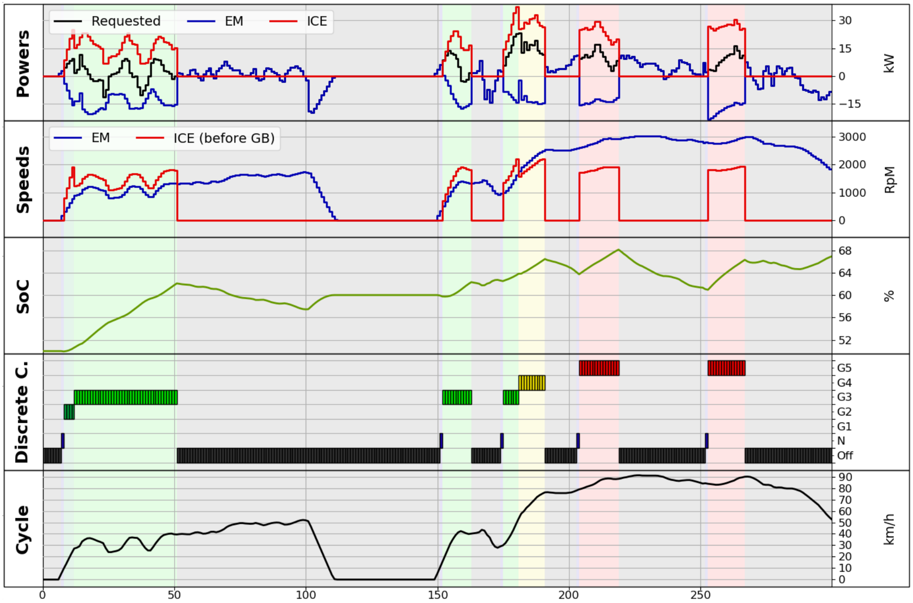


Figure 5.5: Result for 6[s] of Prediction Horizon

Tests on Disjunctive OA for Optimal Control

This subsection considers DOA-OC coupled with a relaxation-shift procedure. In other words, some the linearizations generated in one MPC iterations are reused in the next to reduce the number of OA iterations necessary to solve each single MPC subproblem. This setting is the best possible for the algorithm to date, however, as explained in the concluding part of this dissertation, in future additional improvements could be achieved.

Figure 5.5 shows the first 300 seconds of the solution obtained by setting the prediction horizon to 6 seconds ($h := 6\,\text{s}$). The vehicle is assumed to start with 30kg of fuel in the tank and a battery state-of-charge equal to 50%. Unsurprisingly, the control system lets the EMG handle the starts from standstill of the vehicle and generally tries to keep the ICE running around the 2000rmp mark. In fact, as visible in Figure 5.3, at such speed the ICE efficiency is maximal. On the battery management side, the control strategy tends to stabilize the state-of-charge between the 60% and the 70% mark.

As Figure 5.6 shows, the average time spent in each MPC iteration increasing with the length of the prediction horizon. Selecting $h := 3\,\text{s}$ or $h := 6\,\text{s}$ results in a time-per-iteration well below the real time requirements with peaks that never exceed the one second barrier. Contrarily, assuming $h := 9\,\mathrm{s}$ leads to optimization times which are often incompatible with the desired control frequency.

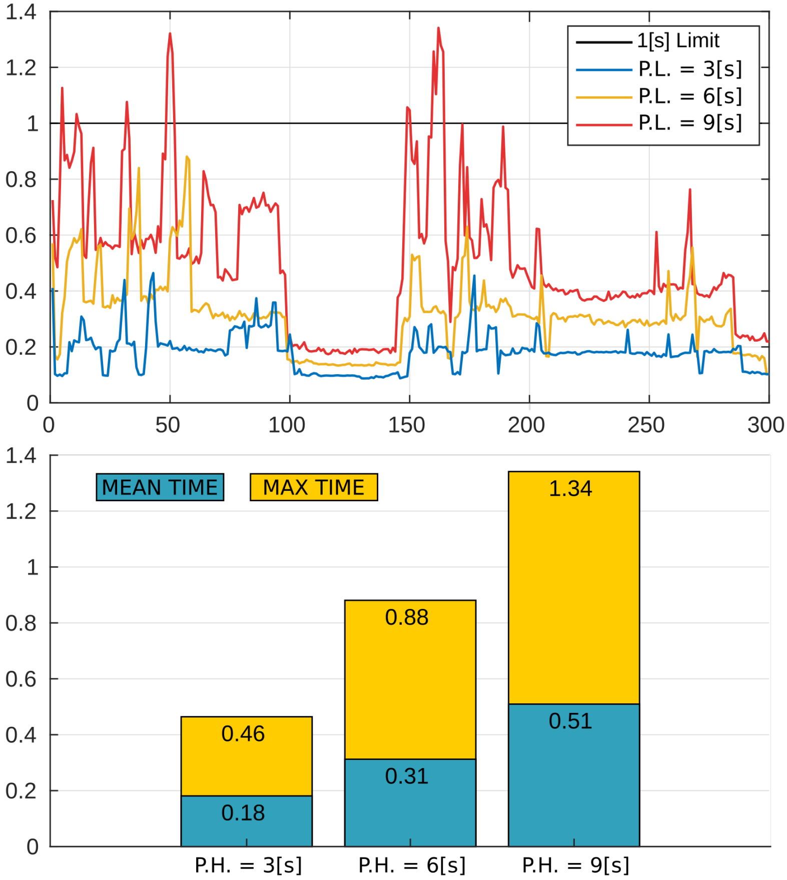


Figure 5.6: Time-per-Iteration for Disjunctive OA with Linearizations Shift at the Variation of the Prediction Horizon

Clearly, also the characteristics of the obtained control strategy depend on the length of the prediction horizon. We compare the discrete control actions relative to the three considered horizon lengths in two ways: first, in terms of objective value after 300 seconds of simulation in Table 5.1, then qualitatively in Figure 5.7.

Horizon Length	3s	6s	9s
Objective Value	230.468	205.719	203.900
$\sum_{k=0}^{300} \mathrm{dF}_k$	173.321	152.496	151.064
$\sum_{k=0}^{300} \left(\mathrm{dF}_k + K^{\mathrm{E2F}}(\bar{\mathrm{E}}_j)\cdot \mathrm{P}_k^{\mathrm{BAT}}\right)$	57.1469	53.223	52.837

Table 5.1: Disjunctive OA: Overall Objective at the Variation of the Prediction Horizon

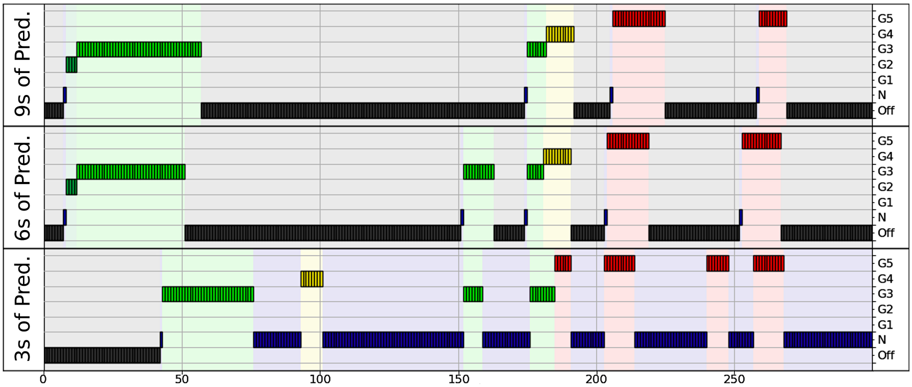


Figure 5.7: Disjunctive OA: Obtained Discrete Control Actions at the

From Figure 5.7 it is evident that the control actions resulting from setting $h := 6\mathrm{s}$ or $h := 9\mathrm{s}$ are quite similar in structure and, according to the above table, there is only a small difference in terms of fuel consumption and objective value between the two. However, setting $h := 3\mathrm{s}$, results in a totally different control action with a much higher objective value. Notably, considering only three seconds of prediction makes less appealing for the optimization to shift gear or change the state of the engine. This is due to the fact that the cost of shifting or turning the engine on/off is rarely dominated by the objective gain resulting from having the vehicle in the optimal state for the next three seconds.

In conclusion, for the current setup, setting a prediction horizion length of 6 seconds appears to be the best compromise between the time required for each MPC iteration and the quality of the resulting control strategy.

Comparison with SCIP a State-of-the-Art Mixed-Integer Non-Convex Solver

This subsection answers to the question whether or not the developed scheme, which exploits the particular structure of the considered problems, does outperform a generic mixed-integer non-convex solver. In the performed tests. SCIP ([41]), being one of the fastest non-commercial solvers of its type, was chosen as a representative of the mixed-integer non-convex solvers class. In order to level up the comparison, the relaxation-shift procedure of Disjunctive OA was disabled (as no similar support for SCIP was available). The experiments consisted of running MPC 120 MPC iterations with both solvers and using 3, 6 and 9 seconds of prediction horizon. The results are summarized in Figure 5.8.

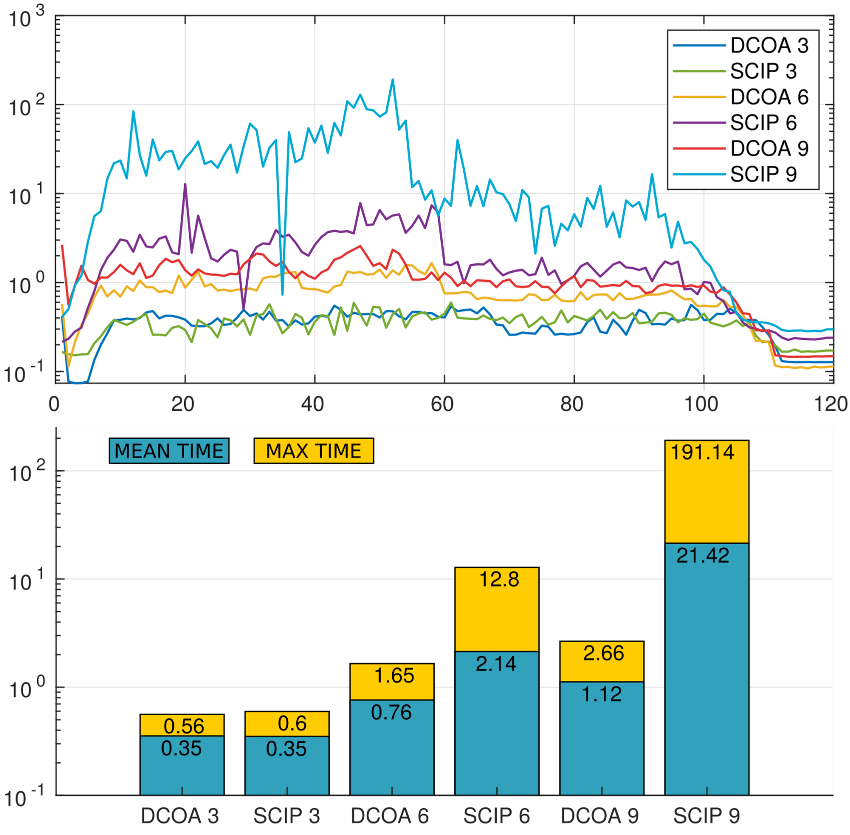


Figure 5.8: Comparison Disjunctive OA versus SCIP

Remark 1: Differently from the last subsection, for the representation of the results of the tests currently under exam a logarithmic scale was used. This is due to the fact that the hereby depicted quantities exhibit large reciprocal differences: up to three orders of magnitude.

Regarding the 3 seconds of prediction horizon tests, Figure 5.8 shows no noteworthy difference between the tested solvers. This is probably due to the fact that our implementation of the DOA-OC algorithm is not really optimized and its higher start-up time overshadows any possible performance edge over SCIP. However, looking at the 6 and 9 seconds of prediction horizon tests we get a profoundly different picture. In fact, as the prediction horizon length increases the solution time with SCIP grows very rapidly until reaching almost 200 seconds of maximum time per single iteration with only 9 seconds of prediction horizon. This is not surprising as SCIP needs to branch on the continuous variables in addition to the binary ones in order to cope with the non-covexity of the problem. In meanwhile, DOA-OC is able to avoid any kind of spatial Branch&Bound. Hence, it shows a much tamer complexity growth as its performance is similar to the one of the classical (convex) OA scheme.

Finally, comparing the data collected for DOA-OC between Figure 5.6 and Figure 5.8 we can draw some conclusions on the effects of the relaxation-shift procedure. In fact, the reduction in average time-per-iteration due to the forward propagation of the obtained linear approximations appears to be about 50%. Simultaneously, the time-per-iteration obtained using of relaxation-propagation technique appears be much more variable than the one obtained without it. This is probably due to variability in the degree of similarity between the solution of one MPC iteration and the solution of the next. Nevertheless, it is clear that the relaxation-propagation procedure is capable of providing DOA-OC with a sizeable performance boost.

Remark 2: The collected results are not a measure of the absolute performances of the involved solvers but, rather, they demonstrate the usefulness of exploiting the particular structure of $\mathcal{P}_{\mathrm{DC}}$. In fact, in general, SCIP is a much better optimized solver with respect to the proof-of-concept level Disjunctive OA solver used for the presented tests.

5.6 Summary and Final Comments

In this chapter we presented an original approach to mixed-integer semi-convex optimal control, Disjunctive Outer Approximation for optimal control, that consists of transforming a problem into a disjunctive-convex form and using a specialized OA-like solution strategy. By means of the performed experiments, the characteristics of the new algorithm have been explained and it was shown how, in this setting, Disjunctive Outer Approximation is capable of largely out-perform a state-of-the-art non-convex solver.

Nevertheless, there is still a large margin of improvement for the proposed algorithm. In particular, merging DOA-OC with MIRT-OC would yield a scheme capable to obtain better linearizations updating the big-M coefficients whenever new bounds on the state variables are discovered while allowing for the use of a relaxation-propagation technique also in case of model mismatch.

Chapter 6

Summary, Conclusions and Outlook

This last chapter of the dissertation collects the author's final comments and remarks. The remainder of the chapter is subdivided in three parts. The first part will summarize the presented strategies and the obtained results. In the second part, in line with to the collected evidence, some conclusions will be drawn. Finally, the third part will present the principal directions of further development for the presented technique according to the author's opinion.

6.1 Summary

The first algorithm, Proximal Outer Approximation (POA), is a mixed-integer convex programming scheme having an Outer Approximation (OA) like structure. Similarly to OA, POA uses the solutions of a sequence of NLPs to iteratively build a linearly constrained relaxation of a given MICP while using the solutions of a sequence of MILPs for generating informed guesses on the globally optimal discrete assignment for the problem at hand. However, POA features different NLP and MILP steps with respect to the original OA algorithm. The NLP steps of POA are meant to generate linear relaxations of the constraints of the MICP from within the feasible space of its continuous relaxation. While, the MILP steps of POA consider an additional adaptive distance term into the objective of the MILP subproblems. The different NLP steps allow POA to generate tighter linear relaxations, and therefore, to construct a sufficiently accurate linearly constrained relaxation of the original MICP faster. The modified MILP steps keep the generated discrete assignment guesses close to the feasible space of the original MICP during the phases in which the algorithm has not obtained an accurate relaxation yet. The performances of POA have been compared against the ones of OA over a known public library of benchmarks. The results suggested that POA is capable of yielding faster and more robust convergence with respect to the classical OA algorithm. In particular, the results show that POA makes possible to save more than 40% of computation time in the solution process of highly non-linear problems.

The second technique, namely the Mixed-Integer Real-Time Optimal Control (MIRT-OC) algorithm, represents an innovative approach to mixed-integer Model Predictive Control. MIRT-OC takes inspiration from single-tree mixed-integer convex programming approaches, such as LP/NLP B&B, and extends the single-tree concept to mixed-integer MPC. In MIRT-OC the some of the optimization data generated during one MPC iteration is propagated to the next iteration in the form of an initial linear relaxation and a partially explored B&B-tree for the new MPC subproblem. The most characteristic elements in the proposed algorithm are the two specialized routines that make it possible to adapt the information collected for one MPC subproblem into suitable information for the next subproblem. Such routines make use of the structure of the B&B algorithm as well as the theory of duality for linear/quadratic solvers. In order to evaluate the advantages in using MIRT-OC rather than a more immediate approach to MPC, a case of study involving the control of an Hybrid Vehicle was devised. A number numerical experiments, for varying levels of sensors/model noise and different MPC prediction horizon lengths, have been performed. As a result, MIRT-OC was found capable of providing large reductions in the average computational cost of the MPC iterations even in presence of substantial model inaccuracies and sensors noise. Specifically, on the performed experiments MIRT-OC required the solution of less NLPs and between the 30%-45% of the LPs with respect to the common approach.

The third proposed approach, Disjunctive Outer Approximation for Optimal Control (DOA-OC), is an adaptation of Outer Approximation to disjunctive-convex optimal control problems. The algorithm exploits the big-M reformulation technique and the particular structure of the constraints of the suitable problems to obtain valid OA-compatible linear relaxations. Doing so, DOA-OC allows mixed-integer optimal control problems resulting convex solely in their continuous variables to be solved as efficiently as jointly convex problems are, rather than how their non-convex nature would require. Consequently, in order to asses the performances of the presented methodology a series of numerical experiments involving the control of a hybrid electric vehicle have been performed. Aside from showing the existence of a large computational advantage in using DOA-OC rather than a general non-convex solver for the considered class of problems, the experiments have proven the viability of using the developed technique in the context of MPC.

6.2 Conclusions

This subsection discusses the general findings that can be deduced by observing the results obtained during the devolpment of the present work.

The experience with Proximal Outer Approximation has shown that there exist margins of improvement also in algorithms as well established as Outer Approximation. In particular, the results suggest that, in Outer Approximation based algorithms, it is possible to obtain performance gains by adaptively trading off the chances of generating discrete guesses with low objective values and the chances of generating feasible assignments. Arguably, this is particularly true in an MPC context where the solution of the previous MPC subproblem might give a good insight on the location of a globally optimal assignment for the current subproblem. Additionally, the experiments have also confirmed the beneficial impact that choosing linearization points inside the feasible space has on the quality of the obtained linearizations.

Thanks to the development of MIRT-OC it was proven that it is possible, although not straightforward, to exploit the similarity existing between to subsequent mixed-integer MPC iterations to reduce the computational cost of each iteration. The concept of single-tree methods has already been proven very efficient in mixed-integer optimization, thus, it is reasonable to expect an even greater success in mixed-integer MPC (where the idea can be applied on two distinct levels). Although MIRT-OC is a proof of concept solver and it lacks many of the components that usually make mixed-integer solvers competitive, it seems to contain all the basic ingredients for a very effective approach to mixed-integer MPC.

With DOA-OC it was presented an easy way of extending all the results obtained using Outer Approximation the class of disjunctive-convex optimal control problems. The algorithm entails the possibility of solving in a fairly efficient way non-convex problems that were formerly difficult to tackle. Additionally, the scheme can benefit from advancements in OA-based techniques as well as from more refined preprocessing routines for mixed-integer linear programming. Finally, given the fact that the disjunctive structure is well suited for representing problems in which piece-wise convex constraints/costs appear, DOA-OC might also provide the means for better handling a number of challenging non-convex optimal control problems either via reformulation or via approximation techniques.

6.3 Outlook

The presented algorithms have been developed and presented independently in order to ease the assessment of their performances with respect to the state-of-the art. However, despite the fact that each of the methodologies can still be improved on its own, the first and foremost source of further improvements for them might derive from combining their characteristics. As a consequence, the current section focuses in proposing ideas in this direction.

6.3.1 Combining POA with MIRT-OC

The proximity considerations present in POA could prove particularly beneficial for MIRT-OC where the solution of the lastly solved MPC subproblem may give a good indication on the location of a close-to-optimal points for the current subproblem. As a consequence, it is quite reasonable to consider combining POA with MIRT-OC. Even more so considering that two algorithms have quite compatible structures. However, such confluence is not straightforward to obtain as the objective function used in the MILP steps of POA varies during the optimization. In fact, it is necessary to develop a strategy for being able to rapidly conform the B&B-tree already built to any appropriate change in the MILP objective. In other words, it is necessary to develop a single-tree version of POA.

6.3.2 Combining POA with DOA-OC

Unfortunately, the NLP steps of POA it are not compatible with disjunctive convex problems as in such steps the discrete variables are not fixed, and thus, they would require the solution of challenging non-convex problems. Nevertheless, assuming that the appropriate linearizations generation routine is used, the MILP steps on POA can be applied also to disjunctive problems. Actually, the fact that the MILP steps of POA tend to keep two subsequent discrete assignment guesses close to each other is likely to improve the performances of DOA-OC as the linearizations generated via big-M reformulations are more "local" than the ones generated via Taylor approximations.

6.3.3 Combining DOA-OC with MIRT-OC

In the current status, DOA-OC makes use only of the initial bounds on the problem variables to compute the big-M constants. However, as the optimization goes further new tighter bounds might be discovered. Especially in MPC, where in every new subproblem the initial state is fixed to a certain value, the variable bounds can change dramatically during the process. Additionally, in case of model mismatch it is necessary to be in possession of a revising strategy for the big-M constants for preventing the erroneous removal of feasible discrete assignments from the search space. Computing revised values for the big-M constants is by itself cheap and easy, however, in order to perform such update in a single tree-scheme it is necessary to rapidly obtain a new feasible dual point for each of the nodes of the current B&B-tree. Conveniently, MIRT-OC is well equipped for the task and an extension in this direction would be fairly easy to obtain.

Bibliography

[1] Achterberg T. and Berthold T.; “Improving the feasibility pump”; Discrete Optimization, 4(1):77-86, 2007.

[2] Bayrak A.; “Topology Considerations in Hybrid Electric Vehicle Powertrain Architecture Design”; PhD Thesis, 2015.

[3] Bellman R.E.; “Dynamic Programming”; Princeton University Press, Princeton, 1957.

[4] Belotti P.; “Bound reduction using pairs of linear inequalities”; Journal of Global Optimization, 2013.

[5] Belotti P., Cafieri S., Lee J. and Liberti L.; “Feasibility-based bounds tightening via fixed points”; Lect. Notes Computer Science, 2010

[6] Belotti P., Kirches C., Leyffer S., Linderoth J. and Luedtke J.J.; “Mixed-Integer Nonlinear Optimization”; Acta Numerica, 22:1-131, 2012.

[7] Belotti P., Lee J., Liberti L., Margot F., and Waechter A.; “Branching and Bounds Tightening Techniques for Non-Convex MINLP”; Optimization Methods and Software, 24(4-5):597-634, 2009.

[8] Ben-Chaim M., Shmerlingand E. and Kuperman A.; “Analytic modeling of vehicle fuel consumption”; Energies, 6(1):117-127, 2013.

[9] Bernal D.E., Vigerske S., Trespalacios F. and Grossmann I.E.; “Improving the performance of DICOPT in convex MINLP problems using a feasibility pump”; Optimization Methods and Software, 1:171-190, 2017.

[10] Berthold T.; “Primal MINLP Heuristics in a Nutshell”; Operations Research Proceedings, -:481-486, 2013.

[11] Berthold T.; “RENS: The optimal rounding”; Mathematical Programming Computation,6(1):33-54, 2014.

[12] Berthold T. and Sullivan J.M.; “Heuristic algorithms in global MINLP solvers”; PhD thesis, 2014.

[13] Bertsekas D.; “Dynamic Programming and Optimal Control, Vol. IandII, 4th Edition”; Athena Scientific, 2017.

[14] Bertsekas D.P.; “Nonlinear Programming” (2nd edition); Athena Scientific, 1999.

[15] Bock H.G. and Plitt K.J.; “A multiple shooting algorithm for direct solution of optimal control problems”; In Proceedings 9th IFAC World Congress Budapest, 1984.

[16] Bock H.G. and Plitt K.J.; “A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems”; IFAC Proceedings, 1984.

[17] Bonami P. and Gonçalves J.P.M.; “Heuristics for convex mixed integer nonlinear programs”; Computational Optimization and Applications, 51(2):729-747, 2012.

[18] " Bonami P., Biegler L.T., Conn A.R., Cornuéjols G., Grossmann I.E., Laird C.D., Wächter A. et al."; “An algorithmic framework for convex mixed integer nonlinear programs”; Discrete Optimization, 5(2):186-204, 2008.

[19] Bonami P., Cornuéjols G., Lodi A., and Margot F.; “A feasibility pump for mixed integer nonlinear programming”; Mathematical Programming, 119(2):331-352, 2009.

[20] Buşoniu L., de Bruin T., Tolić D., Kober J. and Palunko I.; “Reinforcement learning for control: Performance, stability, and deep approximators”; Annual Reviews in Control, 46:8-28, 2018.

[21] Bussieck M.R. and Vigerske S.; “MINLP Solver Software”; Wiley Encyclopedia of Operations Research and Management Science, -:1-17, 2011.

[22] C. Kirches; “Fast Numerical Methods for Mixed-Integer Nonlinear Model-Predictive Control”; Springer, 2011.

[23] D’Ambrosio C.; “Application-oriented Mixed Integer Non-Linear Programming”; PhD thesis, 2009.

[24] D’Ambrosio C., Frangioni A., Liberti L. and Lodi A.; “A storm of feasibility pumps for nonconvex MINLP”; Mathematical Programming, 136(2):375-402, 2012.

[25] Dao T., Seamanand A. and Mcphee J.; “Mathematics-Based Modeling of a Series-Hybrid Electric Vehicle”; 5th Asian Conference on Multibody Dynamics, 2010.

[26] De Mauri M., Gillis J., Swevers J.and Pipeleers G.; “Real-Time Model Predictive Control for a Parallel Hybrid Electric Vehicle using Outer Approximation and Semi-Convex Cut Generation”; Proceedings of the IEEE 16th International Workshop in Advanced Motion Control, 2020.

[27] De Mauri M., Gillis J., Swevers J., Pipeleers G.; “A Proximal-Point Outer Approximation Algorithm”; Computational Optimization and Applications, 2020.

[28] De Mauri M., Van Roy W., Gillis J., Swevers J. and Pipeleers G.; “Real Time Iterations for Mixed-Integer Model Predictive Control”; 2020 European Control Conference (ECC), 2020.

[29] De Santis M., Lucidi S. and Rinaldi F.; “Feasibility Pump-like heuristics for mixed integer problems”; Discrete Applied Mathematics, 165:152-167, 2014.

[30] De Santis M., Lucidi S. and Rinaldi F.; “A New Class of Functions for Measuring Solution Integrality in the Feasibility Pump Approach”; SIAM Journal on Optimization, 23(3):1575-1606, 2013.

[31] Dihel M. and Gros S.; “Manuscript of Numerical Optimal Control”;Class Handouts, 2017.

[32] Dihel M., Bock H.G. and Schlöder J.P.; “A real-time iteration scheme for Nonlinear optimization in optimal feedback control”; SIAM J. Control and Optimization, 43:1714-1736, 2005.

[33] Duran M.A. and Grossmann I.E.; “An outer-approximation algorithm for a class of mixed-integer nonlinear programs”; Mathematical Programming, 36(3):307-339, 1986.

[34] E.D. Tate and S. P. Boyd; “Finding Ultimate Limits of Performance for Hybrid Electric Vehicles”; SAE Transactions, 109(6), 1998.

[35] Eifert M.; “Fuel Consumption Reduction with a Starter-Alternator using an MPC-based Optimisation”; 2003 European Control Conference (ECC), -:2469-2474, 2003.

[36] Fischetti M. and Monaci M.; “Proximity search for 0-1 mixed-integer convex programming”; Journal of Heuristics, 20(6):709-731, 2014.

[37] Fischetti M. and Salvagnin D.; “Feasibility pump 2.0”; Mathematical Programming Computation, 1(2-3):201-222, 2009.

[38] Fischetti M., Glover F., and Lodi A.; “The feasibility pump”; Mathematical Programming, 104(1):91-104, 2005.

[39] Fletcher R., Ley S., and Leyffer S.; “Solving mixed integer nonlinear programs by outer approximation”; Mathematical Programming, 66(1-3):327-349, 1994.

[40] Gamrath G.; “Improving strong branching by domain propagation”; EURO J. Comput. Optim., 2:99-122, 2014.

[41] " Gamrath G., Anderson D., Bestuzheva K., Chen W.K., Eifler L., Gasse M., Gemander P., Gleixner A., Gottwald L., Halbig K., Hendel G., Hojny C., Koch T., Bodic P.L., Maher S.J., Matter F., Miltenberger M., Mühmer E., Müller B., Pfetsch M.E., Schlösser F., Serrano F., Shinano Y., Tawfik C., Vigerske S., Wegscheider F., Weninger D. and Witzig J."; “The SCIP Optimization Suite 7.0 Technical Report”; Optimization Online, 2020.

[42] Geißler B., Morsi A., Schewe L. and Schmidt M.; “Penalty Alternating Direction Methods for Mixed-Integer Optimization: A New View on Feasibility Pumps”; SIAM Journal on Optimization, 27(3):1611-1636, 2017.

[43] Gowal S. et al.; “On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models”; Preprint, 2018.

[44] Gros S., Zanon M., Quirynen R., A. Bemporad and M. Diehl; “From linear to nonlinear MPC: bridging the gap via the real-time iteration”; International Journal of Control, 09:1-19, 2016.

[45] Grossman I.E.; “Review of nonlinear mixed-integer and disjunctive programming techniques”; Optimization and Engineering, 3(3):227-252, 2002.

[46] Grossmann I.; “Review of Nonlinear Mixed-Integer and Disjunctive Programming Techniques”; Optimization Engineering, 3(3):227-252, 2002.

[47] Guzzella L. and Sciarretta A.; “Vehicle Propulsion Systems: Introduction to Modeling and Optimization”; Springer, 2013.

[48] Hespanhol P., Quirynen R. and Di Cairano S.; “A Structure Exploiting Branch-and-Bound Algorithm for Mixed-Integer Model Predictive Control”; European Control Conference, 2019.

[49] Hijazi H., Bonami P., and Ouorou A.; “An outer-inner approximation for separable mixed-integer nonlinear programs”; INFORMS Journal on Computing, 26(1):31-44, 2014.

[50] Houska B., Ferreau H.J. and Diehl M.; “An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range”; Automatica 47(10):2279-2285, 2011.

[51] Hu X. et.al.; “Integrated Optimization of Battery Sizing, Charging, and Power Management in Plug-In Hybrid Electric Vehicles”; IEEE Trans. Control Syst. Technol, 2015.

[52] Hunting M.; “The AIMMS outer approximation algorithm for MINLP”; Technical Report AIMMS B.V., 2011.

[53] Jonasson K.; “Analysing Hybrid Drive System Topologies”; Licentiate Thesis.

[54] Kronqvist J., Bernal D.E., and Grossmann I.E.; “Using regularization and second order information in outer approximation for convex MINLP”; Mathematical Programming, 12:-, 2018.

[55] Kronqvist J., Bernal D.E., Lundell A. and Westerlund T.; “A center-cut algorithm for quickly obtaining feasible solutions and solving convex MINLP problems”; Computers and Chemical Engineering, 122:105-113, 2019.

[56] Kronqvist J., Bernal D.E., Lundell A., and Grossmann I.E.; “A review and comparison of solvers for convex MINLP”; Optimization and Engineering, 20(2):397-455, 2019.

[57] Kronqvist J., Lundell A., and Westerlund T.; “The extended supporting hyperplane algorithm for convex mixed-integer nonlinear programming”; Journal of Global Optimization, 64(2):249-272, 2016.

[58] Land A.H. and Doig A.G.; “An automatic method of solving discrete programming problems”; Econometrica, 28(3):497-520, 1960.

[59] Leyffer S.; “Generalized Outer Approximation”; Encyclopedia of Optimization, 2(1):247-254, 2001.

[60] Leyffer S.; “Deterministic Methods for Mixed Integer Nonlinear Programming”; PhD thesis, 1993.

[61] Little J.D.C., Murty K.G., Sweeney D.W. and Karel C.; “An algorithm for the traveling salesman problem”; Operations Research, 11(6):972-989, 1963.

[62] Liu J. and Peng H.; “Modeling and Control of a Power-Split Hybird Vehicle”; Ieee Trans. Control Syst. Technol., 16(6):1242-1251, 2008.

[63] Lundell A., Kronqvist J., and Westerlund T.; “The Supporting Hyperplane Optimization Toolkit A Polyhedral Outer Approximation Based Convex MINLP Solver Utilizing a Single Branching Tree Approach”;2018

[64] Marcucci T. and Tedrake R.; “Warm Start of Mixed-Integer Programs for Model Predictive Control of Hybrid Systems”; IEEE Transactions on Automatic Control, 2020.

[65] Mura R., Member S., Utkinand V. and Onori S.; “Recasting the HEV Energy management problem into an Infinite-Time Optimization Problem including Stability”; 52nd IEEE Conference on Decision and Control, -:, 2013.

[66] Nemhauser G.L. and Wolsey L.A.; “Integer and Combinatorial Optimization”; Wiley-Interscience, Wiley, 1988.

[67] Ngo V., Hofman T., Steinbuchand M. and Serrarens A.; “Optimal Control of the gearshift command for hybrid electric vehicles”; IEEE Trans. Veh. Technol., 61(8):3531-3543, 2012.

[68] Ogata K.; “Modern Control Engineering” (fifth edition); Prentice Hall, 2010.

[69] Puranik Y. and Sahinidis N. V.; “Domain reduction techniques for global NLP and MINLP optimization”; Constraints, 22(3):338-376, 2017.

[70] Quesada I., and Grossmann I.E.; “An LP/NLP based branch and bound algorithm for convex MINLP optimization problems”; Computers and Chemical Engineering, 16(10-11):937-947, 1992.

[71] Quirynen R. and Hespanhol P.; “A Real-Time Iteration Scheme with Quasi-Newton Jacobian Updates for Nonlinear Model Predictive Control”; European Control Conference, 2018.

[72] Rawlings J.B., Mayne D.Q. and Diehl M.; “Model Predictive Control: Theory, Computation, and Design” (2nd ed.); Nob Hill Publishing, 2018.

[73] Sager S.; “A Benchmark Library of Mixed-Integer Optimal Control Problems”;2012

[74] Sager S. (2006); “Numerical methods for mixed-integer optimal control problems”; PhD Thesis, 2006.

[75] Sager S., Bock H. and Diehl M.; “Solving mixed-integer control problems by sum up rounding with guaranteed integer gap”; Mathematical Programming, 2008.

[76] Sager S., Bock H.G. and Diehl M.; “The Integer Approximation Error in Mixed-Integer Optimal Control”; Mathematical Programming, 133:1-23, 2012.

[77] Sawaya N.W. and Grossmann I.E.; “A cutting plane method for solving linear generalized disjunctive programming problems”; Computer Aided Chemical Engineering, 15(C):1032-1037, 2003.

[78] Sen Huang P. et al.; “Achieving verified robustness to symbol substitutions via interval bound propagation”; 9th Int. Jt. Conf. Nat. Lang. Process. Proc. Conf., 4083-4093, 2020.

[79] Sharma S., Knudsen B.R. and Grimstad B.; “Towards an objective feasibility pump for convex MINLPs”; Computational Optimization and Applications, 63(3):737-753, 2016.

[80] Sontag E.; “Mathematical Control Theory: Deterministic Finite-Dimensional Systems”; Springer, 1998.

[81] Sundström O., Guzzella L. and Soltic P.; “Torque-assist hybrid electric powertrain sizing: From optimal control towards a sizing law”; IEEE Trans. Control Syst. Technol., 18(4):837-849, 2010.

[82] Trespalacios F. and Grossmann I.E.; “Improved Big-M reformulation for generalized disjunctive programs”; Comput. Chem. Eng., 76:98-103, 2015.

[83] " Wächter A. and Biegler L.T."; “On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming”; Mathematical Programming, 106(1):25-57, 2006.

[84] Yu H., Castelli-Dezza F. and Cheli F. ; “Optimal Powertrain Design and Control of a 2-IWD Electric Race Car”; 2017 International Conference of Electrical and Electronic Technologies for Automotive, -:1-7, 2017.

[85] Zhang X., C. T. Li, D. Kumand H. Peng; “Prius + and volt -: Configuration analysis of power-split hybrid vehicles with a single planetary gear”; IEEE Transactions on Vehicle Technology, 61(8):3544-3552, 2012.

[86] CasADi (Automatic Differentiation and Symbolic Programming), “https://web.casadi.org/publications”.

[87] OpenBB (MIQP solver), “https://github.com/MassimoDeMauri/OpenBB.jl”.

[88] CLP (QP/LP solver), “https://projects.coin-or.org/Clp”.

[89] Ipopt (NLP solver), “https://projects.coin-or.org/Ipopt”.

[90] Cplex ((MI)QP solver),“https://www.ibm.com/analytics/cplex-optimizer”.

[91] Gurobi((MI)QP solver),“https://www.gurobi.com/”.

[92] Bonmin (MICP solver), “https://projects.coin-or.org/Bonmin”.

[93] MINLPLib (collection of problem instances), “http://www.minlplib.org”.

[94] Advisor, Data Files “FC_INSIGHT”, “FC_SI95” and “MC_AC75” from “https://sourceforge.net/projects/adv-vehicle-sim/”.

Curriculum Vitae

Personal Data

Massimo De Mauri

Date and Place of Birth: October the 26th 1989, Roma, Italia.

Nationality: Italian.

Email: massimo.demauri@gmail.com

Education

2015-2020 - Ph.D. student at the department of Mechanical Engineering, KU Leuven, Leuven, Belgium.

2012-2015 - M.Sc. in Artificial Intelligence and Robotics (Computer Engineering), Università degli Studi La Sapienza, Roma, Italia

2008-2012 - B.Sc. in Mathematics, Università degli Studi di Roma Tre, Roma, Italia.

Pubilcations

Articles in Peer-Reviewed Journals

M. De Mauri, J. Gillis, J. Swevers and G.Pipeleers. “A Proximal-Point Outer Approximation Algorithm”. Computational Optimization and Applications, 2020.

Articles in International Conferences Proceedings

M. De Mauri, J. Gillis, J. Swevers and G. Pipeleers. “Real-Time Model Predictive Control for a Parallel Hybrid Electric Vehicle using Outer Approximation and Semi-Convex Cut Generation”. Proceedings of the IEEE 16th International Workshop in Advanced Motion Control (AMC2020), Kristiansand (Norway), 20-22 April 2020.

M. De Mauri, W. Van Roy, J. Gillis, J. Swevers and G. Pipeleers. “Real Time Iterations for Mixed-Integer Model Predictive Control”. Proceedings of the European Control Conference (ECC2020), Saint Petersburg (Russia), 12-15 May 2020.

Abstracts in International Conferences

A Toolbox for Control and Design Co-optimization of Hybrid Powertrain, 13 Sep. 2016. 4th European Conference on Computational Optimization (EUCCO 2016), Leuven (Belgium), 12-14 Sep. 2016.

Optimal design for hybrid vehicles with switching dynamics, 29 Mar. 2017. Benelux Meeting on Systems and Control, Spa (Belgium), 28-39 Mar. 2017.

A time transformation approach in hybrid vehicles optimal design, 06 Sep 2017. Optimization 2017, Lisbon (Portugal), 06-08 Sep. 2017.

Proximal Outer Approximation, 28 Mar. 2018. Benelux Meeting on Systems and Control, Soesterberg (The Netherlands), 27-29 Mar. 2018.

A Fast Heuristics for Optimal Control of Hybrid Electric Vehicles, 10 Sep. 2018. 5th European Conference on Computational Optimization (EUCCO 2018), Trier (Germany), 09-11 Sep. 2018.

Towards a real-time implementation of Approximate Dynamic Programming for Mixed-Integer Optimal Control, 20 Mar 2019. Benelux Meeting on Systems and Control, Lommel (Belgium) 19-21 Mar. 2019.

Real Time Iterations for Mixed-Integer Model Predictive Control, 10 Mar. 2020. Benelux Meeting on Systems and Control, Elspeet (The Netherlands), 10-12 Mar. 2020.

Collaborations as Secondary Author

T. Singh, M. De Mauri, W. Decré, J. Swevers and G. Pipeleers. "Feedback control of linear systems with optimal sensor and actuator selection". Journal Of Vibration And Control, July 2020.

Combined H∞ linear parameter varying control design and optimal sensor/actuator selection. T. Singh, Z. Dong, M. De Mauri, W. Decré, J. Swevers, G. Pipelers.IFAC 17th European Control Conference 2019, Naples, Italy, 25-28 Jun 2019.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF MECHANICAL ENGINEERING
MECO RESEARCH TEAM
Celestijnenlaan 300
B-3001 Leuven
https://www.mech.kuleuven.be/en/pma/research/meco