Linköping studies in science and technology. Licentiate Thesis No. 1967

On Complexity Certification of Branch-and-Bound Methods for MILP and MIQP with Applications to Hybrid MPC

Shamisa Shoja

LINKÖPING
UNIVERSITY

Linköping studies in science and technology. Licentiate Thesis
No. 1967

On Complexity Certification of Branch-and-Bound Methods for MILP and MIQP with Applications to Hybrid MPC

Shamisa Shoja

LINKÖPING
UNIVERSITY

This is a Swedish Licentiate's Thesis.

Swedish postgraduate education leads to a Doctor's degree and/or a Licentiate's degree.
A Doctor's Degree comprises 240 ECTS credits (4 years of full-time studies).
A Licentiate's degree comprises 120 ECTS credits,
of which at least 60 ECTS credits constitute a Licentiate's thesis.

Linköping studies in science and technology. Licentiate Thesis
No. 1967

On Complexity Certification of Branch-and-Bound Methods for MILP and MIQP with Applications to Hybrid MPC

Shamisa Shoja

`shamisa.shoja@liu.se`
`www.control.isy.liu.se`
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping
Sweden

ISBN 978-91-8075-224-4 (print)
ISBN 978-91-8075-271-8 (PDF)
ISSN 0280-7971

Printed by LiU-Tryck, Linköping, Sweden 2023

To my family!

Abstract

In model predictive control (MPC), an optimization problem is solved at each time step, in which the system dynamics and constraints can directly be taken into account. The MPC concept can be further extended to the control of hybrid systems, where a part of the state and control variables has a discrete set of values. When applying MPC to linear hybrid systems with performance measures based on the 1-norm or the ∞-norm, the resulting optimal control problem can be formulated as a mixed-integer linear program (MILP), while the optimal control problem with a quadratic performance measure can be cast as a mixed-integer quadratic program (MIQP). An efficient method to solve these non-convex MILP and MIQP problems is branch and bound (B&B) which relies on solving convex relaxations of the problem ordered in a binary search tree. For the safe and reliable real-time operation of hybrid MPC, it is desirable to have a priori guarantees on the worst-case complexity such that the computational requirements of the problem do not exceed the time and hardware capabilities.

Motivated by this need, this thesis aims to certify the computational complexity of standard B&B methods for solving MILPs and MIQPs in terms of, e.g., the size of the search tree or the number of linear systems of equations (iterations) that are needed to be solved online to compute optimal solution. In particular, this knowledge enables us to compute relevant worst-case complexity bounds for the B&B-based MILP and MIQP solvers, which has significant importance in, e.g., real-time hybrid MPC where hard real-time requirements have to be fulfilled. The applicability of the proposed certification method is further extended to suboptimal B&B methods for solving MILPs, where the computational effort is reduced by relaxing the requirement to find a globally optimal solution to instead finding a suboptimal solution, considering three different suboptimal strategies. Finally, the proposed framework is extended to the cases where the performance of B&B is enhanced by considering three common start heuristic methods that can help to find good feasible solutions early in the B&B search process.

Populärvetenskaplig sammanfattning

I modellprediktiv reglering (MPC) löses ett optimeringsproblem i varje tidssteg, där systemets dynamik och begränsningar direkt kan tas med i beräkningen. Detta ramverk kan utökas till hybrida system där en del av tillstånden och styrsignalerna har en diskret uppsättning värden. När man använder prestandamått baserade på 1-normen eller ∞-normen kan det resulterande optimala styrproblemet formuleras som ett linjärt program med både kontinuerliga och binära variabler (MILP), medan reglerproblem med kvadratisk kostnadsfunktion för hybrida system kan formuleras på formen mixed-integer quadratic programming (MIQP), där både MILP och MIQP är icke-konvexa. En effektiv metod för att lösa dessa problem är branch and bound (B&B) som bygger på att lösa en sekvens av konvexa relaxeringar av problemet ordnade i ett binärt sökträd. För säker och pålitlig realtidsoperation av hybrid MPC är det önskvärt att ha garantier för värsta tänkbara beräkningskomplexitet så att beräkningskraven för problemet inte överstiger tiden och hårdvarukapaciteten.

Denna avhandling syftar till att certifiera beräkningskomplexiteten hos B&B-algoritmer för att lösa problem av MILP- och MIQP-typ. Fokus i avhandlingen är komplexitet i termer av storleken på sökträdet eller antalet linjära ekvationssystem (LP/QP-iterationer) som måste lösas online för att hitta ett optimum som en funktion av parametern. Denna kunskap gör det möjligt för oss att beräkna relevanta komplexitetsgränser för det *värsta* fallet för B&B-baserade MILP- och MIQP-lösare, vilket är av stor betydelse för realtids-MPC för hybrida system. Vidare visas hur användbarheten av den föreslagna certifieringsmetoden kan utökas till suboptimala B&B-algoritmer där beräkningsbördan minskas genom att lätta på kravet att hitta en globalt optimal lösning för att istället hitta en suboptimal lösning, där tre olika suboptimala strategier studeras. Slutligen utvidgas det föreslagna ramverket ytterligare till de fall där prestanda för B&B-metoder förbättras genom att använda olika heuristiska metoder som hjälper till att t.ex. hitta bra tillåtna lösningar tidigt i sökträdet.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my supervisor, Daniel Axehill, for continuous support and encouragement during the last years. Thank you for all the feedback, encouragement, and for always being available for discussions. You have been a great source of ideas and inspiration.

I would like to thank the head of division, Martin Enqvist, for keeping up a nice and friendly working climate, and Ninna Stensgård for helping me with administrative tasks. A special thank is directed to Daniel Arnström for great collaborations during the past years. Further, I am grateful to my current and former colleagues at Automatic Control division for creating a work environment that is both stimulating and friendly. Special thanks to Carmela Bernardo, Daniel Arnström, Anton Kullberg, and Daniel Bossér for their help with proof-reading parts of the thesis.

This work was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. WASP did not only contribute with funding, but also an invaluable network with people from different universities, which I am grateful for that.

My sincere thanks go out to my friends: Hamed, Narges, Fatemeh, Carmela, Shervin, Roghayeh, Nasim, Mobin, Mina, and Farnaz for their unwavering support throughout my journey towards this Licentiate's degree. And to my colleagues at the Automatic Control division for all the after-work activities and gatherings we have had over the years. I look forward to many more memorable moments with you.

Last but not least, I would like to thank my family: Shahrokh, Shahnaz, Shila, Shina, and Karo. Your love, patience, and understanding have been a constant source of inspiration, and I am very much fortunate to have you in my life. I love you all!

Linköping, May 2023
Shamisa Shoja

Contents

II Publications

Notation

Some sets

Notation	Meaning
$\mathbb{N}_{1:m}$	Set of positive integers up to m
$\mathbb{R}$	Set of real numbers
$\mathbb{R}^{m \times n}$	Set of real $m \times n$ matrices
$\mathcal{S}_{+}^{n}$	Set of real $n \times n$ positive semi-definite matrices
$\mathcal{S}_{++}^{n}$	Set of real $n \times n$ positive definite matrices
$\{0, 1\}^n$	Set of vectors with n binary components

Operators

Notation	Meaning
a_i	The ith row of matrix A
$A_{\mathcal{W}}$	The rows of matrix $A \in \mathbb{R}^{m \times n}$ indexed by $\mathcal{W} \subset \mathbb{N}_{1:m}$
$\mathbf{0}$	A matrix, or a vector, with all elements equal to zero
$\{\mathcal{D}_i\}_{i=1}^{n}$	A sequence of n elements ($\{\mathcal{D}_i\}_{i=1}^{n} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$)

Abbreviations

Abbreviation	Meaning
B&B	Branch and Bound
BF	Best First
BrF	Breadth First
CFTOC	Constrained Finite-Time Optimal Control
DF	Depth First
EQP	Equality-constrained Quadratic Programming
FP	Feasibility Pump
KKT	Karush-Kuhn-Tucker
LICQ	Linear Independence Constraint Qualification
LP	Linear Programming
MILP	Mixed-Integer Linear Programming
MIQP	Mixed-Integer Quadratic Programming
mp	multi-parametric
MPC	Model Predictive Control
PWA	Piecewise Affine
PWC	Piecewise Constant
PWQ	Piecewise Quadratic
PPWA	Polyhedral Piecewise Affine
PPWC	Polyhedral Piecewise Constant
PPWQ	Polyhedral Piecewise Quadratic
QP	Quadratic Programming
RENS	Relaxation Enforced Neighborhood Search
RINS	Relaxation Induced Neighborhood Search

Part I

Background

1 Introduction

1.1 Background and motivation

Optimal control brings the fields of optimization and automatic control together to create a framework for computing optimal control inputs to the system under control. A popular variant of optimal control is model predictive control (MPC) which has had a significant influence on the industry. A reason that MPC has proven successful is that it can easily handle the control of multi-variable systems while considering constraints on both control inputs and system states. MPC involves solving an optimal control problem posed as an optimization problem at each sampling time to determine the next control input.

A model of the system under control is used in MPC to predict the future behavior of a system. The mathematical model of a dynamical system is usually stated using differential or difference equations, which typically arise from the physical laws governing the system's dynamics. Therefore, most control theories and tools deal with models that describe the evolution of real-valued signals. However, in many applications, the system to be controlled includes not only real-valued signals but also discrete-valued signals that derive from, e.g., on/off conditions and Boolean relations. Such systems are called hybrid systems, where continuous and discrete dynamics interact in a common framework.

There has been a growing interest in hybrid systems within the computer science and control communities. The interest is motivated by the recent technological development in, for example, the control of discrete event systems and embedded systems that also include a logical decision-making device within a physical process.

When MPC is applied to the control of hybrid systems, the optimization problem to be solved is mixed-integer, containing both real-valued and binary-valued decision variables. The discontinuous nature of these problems poses extra challenges in solving those compared to systems with just real-valued signals, highlighting the need for efficient optimization routines to solve such problems. Hybrid MPC can also be implemented in embedded hardware, where the optimal control problem is used in making real-time decisions over a limited time frame and/or memory. Hence, optimization problems need to be solved faster and with limited memory and computational resources. The complexity of solving such optimization problems may significantly vary at each time instance since these problems are dependent on the current state of the system that changes over time. It is, therefore, critical to ensure that the computing resources available are sufficient to solve the optimization problems within the time constraints before employing the solver. This motivates the idea behind the research presented in this thesis, which is to provide a priori guarantees that the solver used is capable of solving all possible (mixed-integer) optimization problems that may arise online within the limited time frame and memory for the application in question.

Based on the chosen performance measure, such optimization problem can take the form of a mixed-integer linear or quadratic program (MILP/MIQP). The most commonly used technique to solve these problems is branch and bound (B&B). The aim in this thesis is to provide worst-case bounds on the computational complexity of B&B methods for solving all possible optimization problems that can be encountered online. The complexity measure can be chosen as the total number of B&B nodes, iterations, or even the number of floating-point operations. The proposed certification methods additionally provide precise insights into the performance of the B&B method considered for the set of problems to be solved. A by-product of the proposed certification framework is solutions to multi-parametric MILPs and multi-parametric MIQPs that are computed offline for all parameters taken from a polyhedral parameter set.

1.2 Thesis outline

This thesis is divided into two parts, where Part I contains background material and Part II consists of the published papers.

Part I - Background

The first part of this thesis presents optimization preliminaries and relevant theoretical background material, including convex and mixed-integer optimization, mixed-integer linear and quadratic programming, B&B methods, and model predictive control for hybrid systems. The main purpose of this part is to provide theoretical foundations for the publications presented in Part II.

Part II - Publications

The second part of this thesis contains the publications listed below. A summary of the publications and the author's contribution to each one is given here.

Paper A: Overall Complexity Certification of a Standard Branch and Bound Method for Mixed-Integer Quadratic Programming

Paper A is an edited version of:

> Shamisa Shoja, Daniel Arnström, and Daniel Axehill. Overall complexity certification of a standard branch and bound method for mixed-integer quadratic programming. In *Proceedings of 2022 American Control Conference (ACC)*, pages 4957–4964, 2022. doi: 10.23919/ACC53348.2022.9867176.

Summary: Paper A presents a method to certify the computational complexity of a standard B&B method for solving MIQP problems defined as instances of a multi-parametric MIQP. Beyond previous work, not only the size of the binary search tree is considered, but also the exact complexity of solving the relaxations in the nodes by using recent results from exact complexity certification of active-set QP methods. With the algorithm proposed in this paper, a total worst-case number of QP iterations to be performed in order to solve the MIQP problem can be determined as a function of the parameter in the problem. An important application of the proposed method is MPC for hybrid systems, which can be formulated as an MIQP problem that has to be solved in real-time. The usefulness of the proposed method is successfully illustrated in numerical examples.

Background and contribution: The main idea of this work was conceived by Daniel Axehill. The author of this thesis contributed with the majority of the work including theoretical derivations, implementations, evaluations, and writing the manuscript. Daniel Ar. contributed in refining the idea and developing the theoretical derivations in the paper. Daniel Ax. and Daniel Ar. contributed with technical discussions, assisted in the development of the theoretical derivations, and reviewed the manuscript.

Paper B: Exact Complexity Certification of a Standard Branch and Bound Method for Mixed-Integer Linear Programming

Paper B is an edited version of:

> Shamisa Shoja, Daniel Arnström, and Daniel Axehill. Exact complexity certification of a standard branch and bound method for mixed-integer linear programming. In *Proceedings of the 61st IEEE Conference on Decision and Control (CDC)*, pages 6298–6305, 2022. doi: 10.1109/CDC51059.2022.9992451.

Summary: In Paper B, we present an algorithm to exactly certify the computational complexity of a standard B&B-based MILP solver. By the proposed method, guarantees on worst-case complexity bounds, e.g., the worst-case iterations or size of the B&B tree, are provided. This knowledge is a fundamental requirement for the implementation of MPC in a real-time system. Different node selection strategies, including best-first, are considered when certifying the complexity of the B&B method. Furthermore, the proposed certification algorithm is extended to consider the warm-starting of the inner solver in B&B.

Background and contribution: The idea of this work evolved after a discussion between the author of this thesis, Daniel Arnström, and Daniel Axehill. The author of this thesis contributed with the majority of the work including theoretical derivations, implementations, evaluations, and writing the manuscript. Daniel Ar. contributed in particular in the BASISRECOVERY procedure. Daniel Ax. and Daniel Ar. contributed with technical discussions, helped in refining the idea, and reviewed the manuscript.

Paper C: Exact Complexity Certification of Suboptimal Branch-and-Bound Algorithms for Mixed-Integer Linear Programming

Paper C is an edited version of:

> Shamisa Shoja and Daniel Axehill. Exact complexity certification of suboptimal branch-and-bound algorithms for mixed-integer linear programming. *Accepted at the 22nd IFAC World Congress*, 2023.

Summary: In Paper C, we extend the result in Paper B to exactly certify the computational complexity of standard suboptimal B&B algorithms for computing suboptimal solutions to MILP problems. Three well-known approaches for suboptimal B&B are considered. This work shows that it is possible to exactly certify the computational complexity also when these approaches are used. Moreover, it also enables to compute exact bounds on the level of suboptimality actually to be obtained online, also for methods previously without any such guarantees. It additionally provides a novel deeper insight into how these strategies affect the performance of the B&B algorithm in terms of the required computation time and memory storage. The exact bounds on the online worst-case computational complexity (e.g., the accumulated number of LP solver iterations or size of the B&B tree) and the worst-case suboptimality computed with the proposed method are very relevant for real-time applications such as MPC for hybrid systems.

Background and contribution: This work was initiated through discussions between the author of this thesis and Daniel Axehill. The idea was to extend and tailor the work in Paper B to also certify the B&B-based MILP solvers when finding suboptimal solutions. The author of this thesis contributed with the majority of the work including theoretical derivations, implementations, evaluations, and writing the manuscript. Daniel Ax. contributed with technical discussions and reviewed the manuscript.

Paper D: Exact Complexity Certification of Start Heuristics in Branch-and-Bound Methods for Mixed-Integer Linear Programming

Paper D is an edited version of:

> Shamisa Shoja and Daniel Axehill. Exact complexity certification of start heuristics in branch-and-bound methods for mixed-integer linear programming. *Submitted to the 62nd IEEE Conference on Decision and Control (CDC)*, 2023.

Summary: To enhance the performance of B&B, start heuristic methods are often applied to find good feasible solutions early in the B&B search tree, hence, reducing the overall effort in B&B to find optimal solutions. In Paper D, we extend the complexity certification framework for B&B-based MILP solvers in Paper B to also certify the computational complexity of the start heuristics and their integration into B&B. As a result, the exact worst-case computational complexity of the three considered start heuristics and, consequently, the B&B method when applying each one can be determined offline, which is of significant importance for real-time applications such as hybrid MPC.

Background and contribution: This work was initiated through discussions between the author of this thesis and Daniel Axehill. The idea was to tailor the proposed certification framework in Paper B for more efficient B&B-based MILP solvers. The author of this thesis contributed with the majority of the work including theoretical derivations, implementations, evaluations, and writing the manuscript. Daniel Ax. contributed with technical discussions and reviewed the manuscript.

2 Convex Optimization

Optimization algorithms are the foundation for computing solutions to optimal control problems. This chapter contains a brief overview of the field and introduces some of the most important concepts in the area. The content of the chapter is inspired by [21] and [49], which are comprehensive references to convex optimization.

2.1 Basics of optimization

A general optimization problem can be written as

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) && \text{(2.1a)} \\ \text{subject to} \quad & g_i(x) \leqslant 0, \quad i \in \mathcal{I}, && \text{(2.1b)} \\ & h_j(x) = 0, \quad j \in \mathcal{E}, && \text{(2.1c)} \end{aligned}$$

where $x \in \mathbb{R}^n$ is the optimization variable, $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is the objective function, $g_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $i \in \mathcal{I}$, represents the inequality constraint function, and $h_j : \mathbb{R}^n \rightarrow \mathbb{R}$, $j \in \mathcal{E}$, represents the equality constraint function. Here, $\mathcal{I} = \{1, \ldots, m\}$ and $\mathcal{E} = \{1, \ldots, p\}$ are the index sets of inequality and equality constraints, respectively. If $\mathcal{I} = \mathcal{E} = \emptyset$, the optimization problem (2.1) is unconstrained. The domain $\mathcal{D} \subset \mathbb{R}^n$ of the problem (2.1) is defined as the set of all points where the objective and constraint functions are defined, i.e.,

$$\mathcal{D} \triangleq \{x \in \mathbb{R}^n \ : \ x \in \operatorname{dom} f, \ \ x \in \operatorname{dom} g_i, \ i \in \mathcal{I}, \ \ x \in \operatorname{dom} h_j, \ j \in \mathcal{E}\}. \qquad (2.2)$$

The goal is to compute a solution $x \in \mathcal{D}$ to (2.1) that minimizes the cost function while belonging to the feasible set defined below.

Definition 2.1 (Feasible set). The *feasible set* Ω of (2.1) is defined as the set of points belonging to $\mathcal{D}$ that satisfies all the constraints, i.e.,

$$\Omega \triangleq \{x \in \mathcal{D} : \ \ g_i(x) \leqslant 0, \ i \in \mathcal{I}, \ \ h_j(x) = 0, \ j \in \mathcal{E}\}. \tag{2.3}$$

A solution $x^* \in \Omega$ to (2.1) with objective function value $f^* = f(x^*)$ is called *optimal* if there does not exist another solution with a smaller objective function value, that is, if $f(x^*) \leqslant f(x)$, $\forall x \in \Omega$. The problem (2.1) is said to be *infeasible* if $\Omega = \emptyset$, and is *unbounded* if $f^* = -\infty$. For the infeasible problem, $f^* = +\infty$.

2.1.1 Convex optimization problems

An important class of optimization problems is convex optimization problems that are useful in many areas, including automatic control. The following definitions are required to introduce this class of optimization problems.

Definition 2.2 (Convex set). A set Ω is *convex* if for any $x_1, x_2 \in \Omega$ and any $\alpha \in [0, 1]$, it holds that

$$\alpha x_1 + (1 - \alpha)x_2 \in \Omega. \tag{2.4}$$

Definition 2.3 (Convex function). A function $f : \mathbb{R}^n \to \mathbb{R}$ is *convex* if its domain (dom$f$) is a convex set and for any $x_1, x_2 \in \operatorname{dom} f$ and any $\alpha \in [0, 1]$, it holds that

$$f(\alpha x_1 + (1 - \alpha)x_2) \leqslant \alpha f(x_1) + (1 - \alpha)f(x_2). \tag{2.5}$$

Definition 2.4 (Concave function). A function $f : \mathbb{R}^n \to \mathbb{R}$ is concave if $-f$ is convex.

The optimization problem of (2.1) is called a *convex optimization problem* if the objective function $f(x)$ and the inequality constraint functions $g_i(x)$, $i \in \mathcal{I}$, are all convex and the equality constraint functions $h_j(x)$, $j \in \mathcal{E}$, are affine, that is, $h_j(x) = c_j^T x - d_j$, for some $c_j \in \mathbb{R}^n$ and $d_j \in \mathbb{R}$. A fundamental property of convex optimization problems is stated in the following theorem.

Theorem 2.5 (Optimizer of convex optimization problems). *Consider a convex optimization problem and let x^* be a local optimal solution. Then, x^* is also a global optimal solution.*

Proof: See [21], Section 4.2. □

2.1.2 Lagrange duality

The idea in Lagrange duality is to augment the objective function in (2.1) with the weighted sum of the constraint functions. The *Lagrangian* function $\mathcal{L} : \mathbb{R}^n \times$

$\mathbb{R}^m \times \mathbb{R}^p \rightarrow \mathbb{R}$ with $\mathrm{dom}\mathcal{L} = \mathcal{D} \times \mathbb{R}^m \times \mathbb{R}^p$ is defined as

$$\mathcal{L}(x, \lambda, \nu) = f(x) + \sum_{i \in \mathcal{I}} \lambda_i g_i(x) + \sum_{j \in \mathcal{E}} \nu_j h_j(x), \tag{2.6}$$

where $\lambda = [\lambda_1, \ldots, \lambda_m]^T \in \mathbb{R}^m$ and $\nu = [\nu_1, \ldots, \nu_p]^T \in \mathbb{R}^p$ are called Lagrange multipliers (or dual variables) associated with the inequality constraints (2.1b) and equality constraints (2.1c), respectively.

By minimizing the Lagrangian function with respect to the primal variables for a given λ and ν, the *Lagrange dual function* $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \rightarrow \mathbb{R}$ is obtained as

$$L(\lambda, \nu) = \inf_{x \in \mathcal{D}} \mathcal{L}(x, \lambda, \nu) = \inf_{x \in \mathcal{D}} \left(f(x) + \sum_{i \in \mathcal{I}} \lambda_i g_i(x) + \sum_{j \in \mathcal{E}} \nu_j h_j(x) \right). \tag{2.7}$$

The Lagrange dual function $L(\lambda, \nu)$ in (2.7) is the pointwise infimum of a family of affine functions of (λ, ν), and is, hence, a concave function. This property holds even if the primal problem (2.1) is not convex [21]. Another fundamental property of $L(\lambda, \nu)$ is that it provides a lower bound on the optimal objective function value f^* of (2.1). That is, for any $\lambda \geqslant 0$, the following inequality holds

$$L(\lambda, \nu) \leqslant f^*. \tag{2.8}$$

Therefore, the best possible lower bound on f^* can be achieved by maximizing the Lagrange dual function. This lower bound can be obtained by solving the following optimization problem

$$\underset{\lambda, \nu}{\text{maximize}} \quad L(\lambda, \nu) \tag{2.9a}$$

$$\text{subject to} \quad \lambda \geqslant 0. \tag{2.9b}$$

Problem (2.9) is called the Lagrange dual problem or simply the dual problem associated with the primal problem (2.1). Since $L(\lambda, \nu)$ is concave and constraints (2.9b) are convex, the dual problem (2.9) is a convex optimization problem, regardless of whether the primal problem (2.1) is convex or not [21].

Weak and strong duality

Let λ^* and ν^* denote the dual optimal solutions to (2.9) with the dual objective function value l^*. From (2.8), we have

$$l^* \leqslant f^*. \tag{2.10}$$

The relation (2.10) is called *weak duality*, and it holds even if l^* or f^* are infinite. For example, if the primal problem is unbounded from below ($f^* = -\infty$), then $l^* = -\infty$ from (2.10), meaning that the dual problem is infeasible. On the other hand, if the dual problem is unbounded from above ($l^* = \infty$), then $f^* = \infty$, which means that the primal problem is infeasible [49].

The difference $f^* - l^*$ is referred to as the optimal duality gap and is always non-negative. If this gap is zero, that is, if

$$l^* = f^* \tag{2.11}$$

we say that *strong duality* holds. Strong duality does not generally hold, even for convex optimization problems. Conditions that ensure strong duality holds are called constraint qualifications [21]. One well-known constraint qualification is given by Slater's theorem, which states that for the convex optimization problem (2.1), strong duality holds if Slater's condition holds. This condition states that for strong duality to hold, there must exist a strictly feasible point. Slater's condition and its refined version are defined in the following [21].

Definition 2.6 (Slater's condition). There exists an $x \in \operatorname{relint} \mathcal{D}$ such that

$$g_i(x) < 0,\ i \in \mathcal{I}, \qquad c_j^T x = d_j,\ j \in \mathcal{E}. \tag{2.12}$$

Here, $\operatorname{relint} \mathcal{D}$ is the relative interior of the domain $\mathcal{D}$.

Definition 2.7 (Slater's refined condition). There exists an $x \in \operatorname{relint} \mathcal{D}$ such that

$$g_i(x) \leqslant 0,\ i \in 1,\dots,k, \quad g_i(x) < 0,\ i \in k+1,\dots,m, \quad c_j^T x = d_j,\ j \in \mathcal{E}, \tag{2.13}$$

where $g_i(x)$, $i \in 1,\dots,k$, are all affine functions.

Strong duality implies that if there exists a primal optimal solution x^* with objective value f^* to the primal problem (2.1), then there also exists the dual optimal solution (λ^*, ν^*) with objective value $l^* = f^*$ to the dual problem (2.9), and vice versa [21].

2.1.3 Optimality conditions

Consider an optimization problem in the form of (2.1). Assume that the objective function $f(x)$ and inequality and equality constraint functions $g_i(x)$, $i \in \mathcal{I}$ and $h_j(x)$, $j \in \mathcal{E}$ are all differentiable and also that strong duality holds. Then the primal and dual optimal solutions to these problems satisfy the Karush-Kuhn-Tucker conditions (KKT) conditions defined below.

Definition 2.8 (KKT conditions). For problems of the form (2.1), the conditions

$$\nabla_x \mathcal{L}(x, \lambda, \nu) = \nabla f(x) + \sum_{i \in \mathcal{I}} \lambda_i \nabla g_i(x) + \sum_{j \in \mathcal{E}} \nu_j \nabla h_j(x) = 0, \tag{2.14a}$$

$$g_i(x) \leqslant 0, \quad i \in \mathcal{I}, \tag{2.14b}$$

$$h_j(x) = 0, \quad j \in \mathcal{E}, \tag{2.14c}$$

$$\lambda_i \geqslant 0, \quad i \in \mathcal{I}, \tag{2.14d}$$

$$\lambda_i g_i(x) = 0, \quad i \in \mathcal{I}, \tag{2.14e}$$

are called the KKT conditions [21].

Condition (2.14a) is called the *stationarity* condition, (2.14b) and (2.14c) are the *primal feasibility* conditions, (2.14d) is the *dual feasibility* condition, and (2.14e) is called the *complementary slackness* condition.

The KKT conditions (2.14) are necessary for optimality of the solution to the problem (2.1). If the problem (2.1) is convex, then the KKT conditions are also sufficient for optimality if some constraint qualification condition is satisfied, summarized in the following theorem.

Theorem 2.9. *Consider a convex optimization problem in the form* (2.1) *and assume $f(x)$, $g_i(x)$, $i \in \mathcal{I}$ and $h_j(x)$, $j \in \mathcal{E}$ are all differentiable and also Slater's condition holds. Then the KKT conditions* (2.14) *are necessary and sufficient conditions for optimality and any primal x^* and dual pair (λ^*, ν^*) points satisfying* (2.14) *are primal and dual optimal solutions, respectively.*

Proof: See [21], Section 5.5. □

2.2 Linear programming

An important class of convex optimization problems is *linear programs* (LPs). The optimization problem (2.1) is called an LP if the objective function and constraint functions are all affine. LPs can represent numerous problems directly. They can also appear as subproblems within approaches that solve other constrained optimization problems, including mixed-integer LPs as will be explained in Section 3.1.

In this section, first the LP problem formulation is introduced and then a method to solve these problems is presented. The following definition is required to what follows.

Definition 2.10 (Polyhedron). A polyhedron $\mathcal{R}$ in $\mathbb{R}^n$ is defined as an intersection of a finite set of closed halfspaces in $\mathbb{R}^n$ as

$$\mathcal{R} = \{x \in \mathbb{R}^n : \; Ax \leqslant b\},$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

2.2.1 Problem formulation

Consider LPs in the form

$$\underset{x}{\text{minimize}} \quad c^T x \tag{2.15a}$$

$$\text{subject to} \quad Ax \leqslant b, \tag{2.15b}$$

where $x \in \mathbb{R}^n$, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, and $b \in \mathbb{R}^m$. The feasible set Ω of (2.15) is determined by the affine inequality constraints (2.15b), therefore from Definition 2.10, Ω is a polyhedron.

In the problem (2.15), we have only considered inequality constraints. By standard simple manipulations, LPs that contain equality constraints could also be converted to the form (2.15) (and vice versa) [21] (Chapter 4).

The Lagrangian function defined in (2.6) for the LP problem (2.15) is given by

$$\mathcal{L}_{\mathrm{LP}}(x,\lambda) = c^T x + \lambda^T (Ax - b), \tag{2.16}$$

where $\lambda \in \mathbb{R}^m$. The Lagrange dual function defined in (2.7) for the LPs (2.15) is then obtained as

$$L_{\mathrm{LP}}(\lambda) \;=\; \inf_x \mathcal{L}_{\mathrm{LP}}(x,\lambda) \;=\; \begin{cases} -b^T\lambda & \text{if } \; -A^T\lambda = c, \\ -\infty & \text{if } \; -A^T\lambda \neq c. \end{cases} \tag{2.17}$$

Since we are interested in dual functions that are finite, the dual problem (2.9) for the LP then takes the form

$$\begin{aligned} \underset{\lambda}{\text{minimize}} \quad & b^T\lambda && (2.18a) \\ \text{subject to} \quad & A^T\lambda = -c, && (2.18b) \\ & \lambda \geqslant 0, && (2.18c) \end{aligned}$$

where the equivalent minimization problem in (2.18) has been considered by changing the sign of the objective function. Slater's refined condition is satisfied for the LP problems (2.15) containing affine constraints. Therefore, feasibility of (2.15) implies strong duality [21].

The necessary and sufficient KKT conditions (2.14) for the LP problem (2.15) become

$$\begin{aligned} & A^T\lambda + c = 0, && (2.19a) \\ & Ax - b \leqslant 0, && (2.19b) \\ & \lambda \geqslant 0, && (2.19c) \\ & \lambda_i(a_i^T x - b_i) = 0, \quad \forall i \in \mathbb{N}_{1:m}, && (2.19d) \end{aligned}$$

where a_i^T denotes the ith row of the matrix A and b_i denotes the ith component of the vector b. These conditions include stationarity (2.19a), primal feasibility (2.19b), dual feasibility (2.19c), and complementary slackness (2.19d) conditions.

A standard method to solve LPs is the *simplex method* [24, 48]. Before introducing this method, the following definitions are given which are fundamental for what follows.

Definition 2.11 (Active constraint). A constraint $a_i^T x \leqslant b_i$ is called *active* at a point $\hat{x} \in \mathbb{R}^n$ if it holds with equality, i.e., if $a_i^T \hat{x} = b_i$.

Definition 2.12 (Active set). The *active set* at a feasible point $x \in \mathbb{R}^n$, denoted $\mathcal{A}(x)$, is defined as the set containing the indices of all active constraints at x, i.e., $\mathcal{A}(x) \triangleq \{i \in \mathbb{N}_{1:m} \,:\, a_i^T x = b_i\}$.

Definition 2.13 (Linear independence constraint qualification). For the constraints with indices in the active set $\mathcal{A}(x)$, the linear independence constraint qualification (LICQ) holds if the constraint gradients are linearly independent.

If there exists an optimum point $x^* \in \Omega$ such that LICQ does not hold at x^*, then the LP problem (2.15) is said to be *primal degenerate* [49].

2.2.2 Simplex methods

Simplex methods rely on the observation that if the LP (2.15) is neither infeasible nor unbounded, then a solution is always obtained at a vertex of the feasible set (polyhedron) Ω [49]. A simple strategy would then be to enumerate all vertices of Ω and select the one that results in the smallest cost function as the solution. However, a better approach is to only explore vertices that improve the cost over the visited ones. This is the idea behind the simplex method.

The simplex method is in fact an active-set approach for linear programming that makes a guess of the optimal active set, denoted $\mathcal{A}^* = \mathcal{A}(x^*)$, and updates this estimated set until optimality is detected. This set is called a *basis* (or working set) and is denoted $\mathcal{W}$. A step in the solution sequence is in this thesis called an *iteration*, and the basis at iteration k is denoted $\mathcal{W}^k$. An important property of a basis for LP is that it always contains n indices of active constraints, such that the submatrix $A_{\mathcal{W}} \in \mathbb{R}^{n \times n}$ (containing the rows of A indexed by $\mathcal{W}$) has full row rank. In this way, LICQ holds.

We now study a primal variant of the simplex methods, where stationarity (2.19a), primal feasibility (2.19b), and complementary (2.19d) conditions are ensured throughout all iterations while dual feasibility (2.19c) is sought for. Dual feasibility will only be satisfied when an optimal solution x^* is found. The steps of the simplex method are described in the following, where at each iteration k, one index is removed and one is added to $\mathcal{W}^k$ so that $\mathcal{W}^k$ always includes n elements. The content of what follows is inspired by [20].

Initialization

Consider an iterate x^0 as a vertex of the feasible set. For the initial basis $\mathcal{W}^0$, select n indices of $\mathcal{A}(x^0)$, such that the submatrix $A_{\mathcal{W}^0}$ is invertible. Define the inactive set $\mathcal{W}_c^0 \triangleq \mathbb{N}_{1:m} \setminus \mathcal{W}^0$ as the complement of the basis, containing indices of all inequality constraints that are yet disregarded.

Main Loop

At iteration k, given the current basis $\mathcal{W}^k$, compute the dual variables λ^k from (2.19a) as

$$A_{\mathcal{W}^k}^T \lambda_{\mathcal{W}^k}^k = -c, \tag{2.20}$$

where $\lambda_{\mathcal{W}^k}^k$ is a subvector of λ^k indexed by $\mathcal{W}^k$. Note that from (2.19d), the dual variables for the constraints not included in $\mathcal{W}^k$ are zero, i.e., $\lambda_{\mathcal{W}_c^k}^k = \mathbf{0}$.

If $\lambda^k \geqslant 0$

The variable λ^k is dual feasible. The KKT conditions (2.19) are all satisfied and the *optimal solution* has been found.

Else (*Pivoting: remove an index and add one index to $\mathcal{W}^k$*)

The variable λ^k is not dual feasible. Select a *leaving* index $l \in \mathcal{W}^k$ corresponding to the negative component of the dual variable λ^k and remove it from $\mathcal{W}^k$. From Dantzig's selection rule, the most negative component of λ^k

is usually chosen. Moreover, select an *entering* index $e \in \mathcal{W}_c^k$ such that after the following update

$$\mathcal{W}^{k+1} = \mathcal{W}^k \setminus \{l\} \cup \{e\}, \qquad \mathcal{W}_c^{k+1} = \mathcal{W}_c^k \setminus \{e\} \cup \{l\},$$

the next iterate x^{k+1} provides a cost which is monotonically decreasing.

The simplex method needs an initial vertex x^0 and an initial basis $\mathcal{W}^0$ including an associated set of linearly independent constraints. Computing these two can in general be as hard as solving the original LP problem. An approach to determine these values is a so-called *Phase I*, in which an auxiliary linear optimization problem is solved. A detailed description of the Phase I method can be found in [49].

The simplex method described above terminates in a finite number of iterations if all visited vertices are non-degenerate. For comprehensive theoretical details on linear programming and the simplex methods see, e.g., [24, 49, 62].

2.3 Quadratic programming

Another important class of convex optimization problems is *quadratic programs* (QPs). The optimization problem (2.1) represents a QP if the objective function is quadratic and the constraint functions are affine. Many problems can be directly formulated as QPs. These problems can also emerge as subproblems in methods used to solve general constrained optimization problems such as mixed-integer QPs as will be discussed in Section 3.2. This section introduces the QP problem formulation followed by the presentation of active-set methods to solve such problems.

2.3.1 Problem formulation

Consider QPs in the form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{2}x^T H x + f^T x && (2.21a) \\ \text{subject to} \quad & Ax \leqslant b, && (2.21b) \end{aligned}$$

where $x \in \mathbb{R}^n$, $H \in \mathbb{S}^n_{++}$, $f \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, and $b \in \mathbb{R}^m$. The feasible set Ω for (2.21) containing affine inequality constraints is, again, in the form of a polyhedron. Since the Hessian matrix H is positive definite, the problem (2.21) is convex and has a unique optimal solution if the feasible set is not empty [21]. Note that the QP problem (2.21) is convex if $H \succeq 0$.

The Lagrangian function defined in (2.6) for the QP problem (2.21) is given by

$$\mathcal{L}_{\text{QP}}(x, \lambda) = \frac{1}{2}x^T H x + f^T x + \lambda^T (Ax - b), \tag{2.22}$$

where $\lambda \in \mathbb{R}^m$. From (2.14a), $\nabla_x \mathcal{L}_{\text{QP}}(x, \lambda) = Hx + f + A^T \lambda = 0$ results in $x = -H^{-1}\left(f + A^T \lambda\right)$. By using this, the Lagrange dual function (2.7) for the QP (2.21) is then obtained as

$$L_{\mathrm{QP}}(\lambda) = -\frac{1}{2}\lambda^T \left(AH^{-1}A^T\right)\lambda - \left(AH^{-1}f + b\right)^T \lambda - \frac{1}{2}f^T H^{-1} f. \tag{2.23}$$

The dual problem (2.9) for QPs using (2.23) can be rewritten as

$$\begin{aligned} &\underset{\lambda}{\text{minimize}} && \frac{1}{2}\lambda^T \left(AH^{-1}A^T\right)\lambda + \left(AH^{-1}f + b\right)^T \lambda && \text{(2.24a)} \\ &\text{subject to} && \lambda \geqslant 0. && \text{(2.24b)} \end{aligned}$$

In (2.24), the sign of the objective function has been changed to consider the equivalent minimization problem and the constant term in (2.23) has been ignored.

Since the constraints in (2.21) are affine, Slater's refined condition is satisfied for QPs, hence, feasibility for convex QPs implies strong duality. The KKT conditions (2.14) for the QP (2.21) become

$$\begin{aligned} &Hx + f + A^T\lambda = 0, && \text{(2.25a)} \\ &Ax - b \leqslant 0, && \text{(2.25b)} \\ &\lambda \geqslant 0, && \text{(2.25c)} \\ &\lambda_i(a_i^T x - b_i) = 0, \quad \forall i \in \mathbb{N}_{1:m}, && \text{(2.25d)} \end{aligned}$$

where a_i^T and b_i denote the ith row and element of A and b, respectively.

There are several methods to compute the solution to (2.21), such as active-set methods [14, 31, 36], interior-point methods [63, 65], and gradient projection methods [9, 51]. To solve QPs encountered in this thesis, we employ an active-set method which is outlined in the following section.

2.3.2 Active-set methods

Active-set methods for QPs are different from the simplex method in the sense that the iterates (and the solution x^*) do not necessarily lie on vertices of the feasible set [49]. To solve QPs that contain inequality constraints, these methods solve a sequence of equality-constrained QPs (EQP) where only the active constraints are considered relevant.

Let $\mathcal{A}^* = \mathcal{A}(x^*)$ denote the optimal active set at the optimum x^*. If $\mathcal{A}^*$ is known in advance, the optimal solution can be found by solving the EQP,

$$\begin{aligned} &\underset{x}{\text{minimize}} && \frac{1}{2}x^T Hx + f^T x && \text{(2.26a)} \\ &\text{subject to} && a_i^T x = b_i, \; i \in \mathcal{A}^*. && \text{(2.26b)} \end{aligned}$$

The optimizer of the EQP (2.26) is the solution to the system of linear equations

$$\begin{bmatrix} H & A_{\mathcal{A}^*}^T \\ A_{\mathcal{A}^*} & 0 \end{bmatrix} \begin{bmatrix} x^* \\ \lambda_{\mathcal{A}^*} \end{bmatrix} = \begin{bmatrix} -f \\ b_{\mathcal{A}^*} \end{bmatrix}, \tag{2.27}$$

which are the KKT conditions (2.25) for the EQP problem (2.26). Here, $A_{\mathcal{A}^*}$ is the submatrix of A and $b_{\mathcal{A}^*}$ is the subvector of and b, containing the rows indexed by

$\mathcal{A}^*$. If $\mathcal{A}^*$ is known, the optimal solution x^* to the QP (2.21) can thus be found by solving a single system of linear equations (2.27).

The optimal active set $\mathcal{A}^*$ is, however, usually unknown in advance. This motivates the idea behind the active-set methods: identifying $\mathcal{A}^*$. The identification of the active set is achieved by iteratively updating a so-called *working set*, denoted $\mathcal{W}$, which serves as an estimate of $\mathcal{A}^*$. To summarize, the solution to the QP problem (2.21) in the active-set methods is computed by solving a sequence of EQPs (system of linear equations) in the form (2.27), where each EQP is determined by the current working set.

These methods for solving QPs come in different types, including primal, dual, and primal-dual methods [4]. *Primal* methods generate iterates that remain primal feasible and dual feasibility is searched for [31, 49], while *dual* methods generate iterates that remain dual feasible and primal feasibility is searched for [12, 36]. In *primal-dual* methods, neither primal nor dual feasibility is maintained at each iteration until the optimal primal-dual pair is found [40]. A standard primal active-set method is presented in the following subsection.

A primal active-set algorithm

As mentioned above, the idea behind primal active-set methods is to select working sets such that the iterate at each iteration stays primal feasible while the objective function decreases monotonically. In particular, at each iteration of this method, the KKT conditions (2.25a), (2.25b), and (2.25d) are satisfied, whereas dual feasibility (2.25c) is only satisfied when the optimal solution is found.

Let x^k denote the kth iterate in the algorithm with the working set $\mathcal{W}^k$. At each iteration of the algorithm, one constraint is added to, or removed from, $\mathcal{W}^k$ until the optimal solution x^* is found and the algorithm terminates. The dual variable λ^k is only computed when the termination criterion is checked in the algorithm or when an index is removed from $\mathcal{W}^k$. In the following, the steps of the active-set algorithm are first explained, then the active-set algorithm is presented. The following content is inspired by [4, 20, 49].

Initialization

Consider an iterate x^0 and a working set $\mathcal{W}^0$, where $\mathcal{W}^0 \subseteq \mathcal{A}(x^0)$. Define the inactive set $\mathcal{W}_c^0 \triangleq \mathbb{N}_{1:m} \setminus \mathcal{W}^0$ as the complement of the working set. Note that, as primal feasibility needs to be maintained in the primal active-set method, x^0 should be a primal feasible point.

Main Loop

At iteration k, compute the solution $\hat{x}^{k+1}$ to the EQP determined by the current working set $\mathcal{W}^k$

$$\begin{aligned} \hat{x}^{k+1} = \underset{x}{\operatorname{argmin}} \quad & \frac{1}{2}x^T H x + f^T x && (2.28a) \\ \text{subject to} \quad & a_i^T x = b_i,\ i \in \mathcal{W}^k. && (2.28b) \end{aligned}$$

If $A\hat{x}^{k+1} \leqslant b$

The variable $\hat{x}^{k+1}$ is primal feasible. Compute the dual variables λ^k for the problem (2.28) from (2.25a) by solving,

$$A^T_{\mathcal{W}^k} \lambda^k_{\mathcal{W}^k} = -H\hat{x}^{k+1} - f. \tag{2.29}$$

Note that from (2.25d), the dual variables for the constraints not in $\mathcal{W}^k$ are zero, i.e., $\lambda^k_{\mathcal{W}^k_c} = \mathbf{0}$.

If $\lambda^k \geqslant 0$

The variable λ^k is dual feasible. Hence, the KKT conditions (2.25) are all satisfied and the optimal solution $x^* = \hat{x}^{k+1}$ has been found.

Else (*Remove an index from the working set*)

The variable λ^k is not dual feasible. Select an index $l \in \mathcal{W}^k$ corresponding to a negative component of λ^k and remove it from $\mathcal{W}^k$, and also set $\hat{x}^{k+1}$ as the next iterate. That is,

$$\mathcal{W}^{k+1} = \mathcal{W}^k \setminus \{l\}, \quad \mathcal{W}^{k+1}_c = \mathcal{W}^k_c \cup \{l\}, \quad x^{k+1} = \hat{x}^{k+1}.$$

From Dantzig's selection rule, the most negative component of λ^k is usually chosen. By choosing the leaving index l in this way, the next iterate x^{k+1} provides a cost that is decreasing monotonically.

Else $(A\hat{x}^{k+1} \nleqslant b)$

The variable $\hat{x}^{k+1}$ is not primal feasible, so we cannot set $x^{k+1} = \hat{x}^{k+1}$, as at least one of the constraints in $\mathcal{W}^k$ will be violated when moving to the next iterate, and the next iterate might become infeasible. Instead, we need to find the largest step size from $\alpha^k \in [0,1]$ such that next iterate $x^{k+1} = x^k + \alpha^k \left(\hat{x}^{k+1} - x^k\right)$ remains feasible. Let us first explicitly define the step length α^k_i that activates constraint i as

$$a_i^T \left(x^k + \alpha_i^k \left(\hat{x}^{k+1} - x^k\right)\right) = b_i \;\Leftrightarrow\; \alpha_i^k = \frac{b_i - a_i^T x^k}{a_i^T \left(\hat{x}^{k+1} - x^k\right)}. \tag{2.30}$$

The index of the first constraint $e \in \mathcal{W}^k_c$ that blocks the way towards $\hat{x}^{k+1}$ and the largest step size α^* can then be obtained

$$e = \underset{i \in \mathcal{W}^k_c :\, a_i^T \hat{x}^{k+1} > b_i}{\operatorname{argmin}} \alpha_i^k, \quad \alpha^* = \alpha^k_e. \tag{2.31}$$

The next iterate is then given by $x^{k+1} = x^k + \alpha^* \left(\hat{x}^{k+1} - x^k\right)$.

If $\alpha^* = 1$ (*No change in the working set*)

$$\mathcal{W}^{k+1} = \mathcal{W}^k, \quad \mathcal{W}^{k+1}_c = \mathcal{W}^k_c.$$

Else (*Add an index to the working set*)

Add the index $e \in \mathcal{W}^k_c$ to $\mathcal{W}^k$, that is,

$$\mathcal{W}^{k+1} = \mathcal{W}^k \cup \{e\}, \quad \mathcal{W}^{k+1}_c = \mathcal{W}^k_c \setminus \{e\}.$$

The primal active-set method that has been described herein is summarized in Algorithm 1. As inputs, the algorithm takes an initial feasible point x^0 and an initial working set $\mathcal{W}^0$, as well as the maximum number of iterations $k^{\max}$. It then outputs the optimal pair (x^*, λ^*) if the problem is not infeasible and $k^{\max}$ has not been reached.

Algorithm 1 A primal active-set algorithm [49]

```
Input: $x^0, \mathcal{W}^0, k_{\max}$
Output: $x^*, \lambda^*$
$k \leftarrow 0, x^* \leftarrow \varnothing, \lambda^* \leftarrow \varnothing$
$\mathcal{W}_c^0 \leftarrow \mathbb{N}_{1:m} \setminus \mathcal{W}^0$
while $k \leqslant k^{\max}$ do
    Given $\mathcal{W}^k$, compute $\hat{x}^{k+1}$ from (2.28)
    if $A\hat{x}^{k+1} \leqslant b$ then                         ▷ $\hat{x}^{k+1}$ is primal feasible
        Compute $\lambda^k$ by solving (2.29)
        if $\lambda^k \geqslant 0$ then
            $x^* \leftarrow \hat{x}^{k+1}, \lambda^* \leftarrow \lambda^k$          ▷ Optimal solution found
            STOP
        else                                              ▷ Remove an index from $\mathcal{W}^k$
            $l \leftarrow \underset{i \in \mathcal{W}^k}{\operatorname{argmin}} \lambda_i^k$
            $x^{k+1} \leftarrow \hat{x}^{k+1}$
            $\mathcal{W}^{k+1} \leftarrow \mathcal{W}^k \setminus \{l\}, \mathcal{W}_c^{k+1} \leftarrow \mathcal{W}_c^k \cup \{l\}$
    else                                                  ▷ $\hat{x}^{k+1}$ is not primal feasible
        Compute $\alpha^*$ and the corresponding index $e \in \mathcal{W}_c^k$ from (2.31)
        $x^{k+1} = x^k + \alpha^* \left(\hat{x}^{k+1} - x^k\right)$
        if $\alpha^* = 1$ then
            $\mathcal{W}^{k+1} \leftarrow \mathcal{W}^k, \mathcal{W}_c^{k+1} \leftarrow \mathcal{W}_c^k$
        else                                              ▷ Add an index to $\mathcal{W}^k$
            $\mathcal{W}^{k+1} \leftarrow \mathcal{W}^k \cup \{e\}, \mathcal{W}_c^{k+1} \leftarrow \mathcal{W}_c^k \setminus \{e\}$
    $k \leftarrow k + 1$
return $x^*, \lambda^*$
```

Computing an initial feasible point x^0 for QPs can again be done using the Phase I method [49]. The initial working set $\mathcal{W}^0$ is chosen as a subset of the active constraints at x^0 ($W^0 \subseteq \mathcal{A}(x^0)$). Different choices of $\mathcal{W}^0$ result in different iteration sequences in Algorithm 1.

A dual active-set algorithm

As mentioned, the primal active-set method requires an initial feasible point x^0. Finding x^0 for the primal problem (2.21) is a nontrivial task. For example, the Phase I method used to find x^0 can on average take one-third or one-half of the solution processing time [36], making it a drawback of primal methods.

Dual active-set methods work with the dual problem (2.24) instead. For such problems, finding an initial feasible point is in fact trivial, as for instance, the origin is always a feasible point to (2.24). It can hence be advantageous to use a dual active-set solver.

A dual active-set algorithm starts with a feasible point λ^0 for the dual problem (2.24). At iteration k in a main loop, it computes x^k and updates working sets $\mathcal{W}^k$ by adding or removing one index from $\mathcal{W}^k$, until x^k becomes primal feasible, and the algorithm terminates. The optimal primal-dual pair (x^*, λ^*) is then returned if the problem is feasible. An efficient dual active-set solver can be found in, e.g., [6].

2.4 Multi-parametric programming

Parametric programming refers to an optimization problem that depends on a parameter. If the optimization problem involves multiple parameters, it is referred to as *multi-parametric programming*. In multi-parametric programming, the solution is characterized for a set of parameter values. Two special cases of parametric programming which are of interest in this thesis are introduced in this section. The following definitions are first given.

Definition 2.14 (Polyhedral partition). A collection of sets $\{\Theta^i\}_{i=1}^N$ is a *partition* of a set Θ if $\mathring{\Theta}^i \cap \mathring{\Theta}^j = \emptyset, i \neq j$, and $\cup_{i=1}^N \Theta^i = \Theta$, where $\mathring{\Theta}^i$ denotes the interior of the region Θ^i. Moreover, $\{\Theta^i\}_{i=1}^N$ is a *polyhedral partition* of a polyhedral set Θ if $\{\Theta^i\}_{i=1}^N$ is a partition of Θ, and Θ^i is a polyhedron, $\forall i$.

Definition 2.15 (Polyhedral piecewise quadratic functions). A function $h(\theta) : \Theta \rightarrow \mathbb{R}^n$, where $\Theta \subset \mathbb{R}^{n_\theta}$, is said *piecewise quadratic* (PWQ) if there exists a partition $\{\Theta^i\}_i$ of Θ and $h(\theta) = \theta^T R^i \theta + H^i \theta + K^i, \forall \theta \in \Theta^i$. The function is said *piecewise affine* (PWA) if $R^i = \mathbf{0}$, $\forall i$, and *piecewise constant* (PWC) if $R^i = \mathbf{0}$ and $H^i = \mathbf{0}$, $\forall i$.
If Θ is a polyhedral set and $\{\Theta^i\}_i$ is a polyhedral partition, then the PWQ function $h(\theta)$ is called polyhedral PWQ (PPWQ). Polyhedral PWA (PPWA) and polyhedral PWC (PPWC) functions are defined analogously.

2.4.1 Multi-parametric LP

Consider multi-parametric LPs (mp-LPs) in the form

$$\begin{aligned} &\underset{x}{\text{minimize}} && c^T x && (2.32a)\\ &\text{subject to} && Ax \leqslant b + W\theta, && (2.32b) \end{aligned}$$

with $x \in \mathbb{R}^n$ as the vector of decision variables and $\theta \in \Theta_0 \subseteq \mathbb{R}^{n_\theta}$ as a vector of parameters. Here, $\Theta_0 \subseteq \mathbb{R}^{n_\theta}$ is the parameter set and is assumed to be polyhedral. The objective function is given by $c \in \mathbb{R}^n$, and the inequality constraints are defined by $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $W \in \mathbb{R}^{m \times n_\theta}$. The set of feasible parameters for (2.32) is defined as

$$\Theta^* = \{\theta \in \Theta_0 : \Omega(\theta) \neq \emptyset\}, \tag{2.33}$$

where $\Omega(\theta) \triangleq \{x \in \mathbb{R}^n : \; Ax \leqslant b + W\theta\}$ is a point-to-set map assigning a feasible set to a parameter θ.

Let us denote the optimal value function or simply the value function to (2.32) $J^*(\theta)$, and the optimal solution function $x^*(\theta)$. The properties of the solution $x^*(\theta)$ to mp-LPs are discussed in detail in [11] and [20] (Chapters 5–6), summarized below.

Theorem 2.16. *Consider the mp-LP problem* (2.32) *and assume that there exists a $\bar{\theta}$ and $x^*(\bar{\theta})$ with a bounded cost $J^*(\bar{\theta})$. Then, Θ^* is a nonempty polyhedron, $J^*(\theta)$ is continuous, convex, and PPWA, and the optimizer $x^*(\theta)$ is continuous and PPWA.*

Proof: See [11]. □

2.4.2 Multi-parametric QP

Consider multi-parametric QPs (mp-QPs) in the form

$$\underset{x}{\text{minimize}} \quad \frac{1}{2}x^T H x + f^T x + \theta^T f_\theta^T x \tag{2.34a}$$

$$\text{subject to} \quad Ax \leqslant b + W\theta. \tag{2.34b}$$

The mp-QP problem is given by $H \in \mathbb{S}^n_{++}$, $f \in \mathbb{R}^n$, $f_\theta \in \mathbb{R}^{n \times n_\theta}$. The other variables are defined as in (2.32).

The feasible parameter set Θ^* for (2.34) is also defined as in (2.33). The properties of the solution $x^*(\theta)$ and the value function $J^*(\theta)$ of mp-QPs are discussed in [11] and [20] (Chapters 5–6), summarized in the following theorem.

Theorem 2.17. *Consider the mp-QP problem* (2.34) *and assume that there exists a $(\bar{x}, \bar{\theta})$ such that $A\bar{x} \leqslant b + W\bar{\theta}$. Then, Θ^* is a nonempty polyhedron, $J^*(\theta)$ is continuous and PPWQ, and the optimizer $x^*(\theta)$ is continuous and PPWA.*

Proof: See [11]. □

Over the past years, many researchers have made significant contributions to the field of multi-parametric programming, and in particular of mp-LPs and mp-QPs, both in theory and applications. These contributions have led to the development of new algorithms, methodologies, and software tools such as the Multi-Parametric Toolbox [39]. For detailed description on how to compute parametric solutions to the mp-LPs (2.32) and the mp-QPs (2.34) see [1, 16, 17, 27, 43].

3

Mixed-Integer Optimization

In mixed-integer optimization problems, optimization variables are not allowed to be only real-valued, but also integer-valued. This chapter is intended to introduce two special classes of such optimization problems which are the main focus in the thesis. Common tools to solve these problems: branch-and-bound (B&B) methods, are presented next. Finally, multi-parametric programming for the problems under consideration is introduced.

3.1 Mixed-integer linear programming

Consider the LP problem (2.15) and let decision variables include both real and binary variables. The problem is in the form of a mixed-integer linear program (MILP) written as

$$\mathcal{P}_{\text{MILP}}: \quad \begin{aligned} &\underset{x}{\text{minimize}} && c^T x && \text{(3.1a)} \\ &\text{subject to} && Ax \leqslant b, && \text{(3.1b)} \\ &&& x_i \in \{0,1\}, \quad \forall i \in \mathcal{B}, && \text{(3.1c)} \end{aligned}$$

where $x = [x_c^T, x_b^T]^T \in \mathbb{R}^{n_c} \times \{0,1\}^{n_b}$ denotes the vector of $n = n_c + n_b$ continuous and binary decision variables. The objective function is given by $c \in \mathbb{R}^n$, and the feasible set is determined by $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. Since the n_b optimization variables indexed by the set $\mathcal{B}$ are binary-valued, the MILP problem is non-convex and is known to be $\mathcal{NP}$-hard [64].

Definition 3.1. Let $\hat{x} \in \mathbb{R}^n$. We call $\hat{x}$

- infeasible for (3.1) if it does not satisfy constraints (3.1b)–(3.1c),
- LP-feasible for (3.1) if it satisfies (3.1b) and $0 \leqslant \hat{x}_i \leqslant 1, \forall i \in \mathcal{B}$,

- integer-feasible for (3.1) if it satisfies (3.1b) and (3.1c).

A simplified version of the problem (3.1) is defined by relaxing the integrality constraints (3.1c) into interval constraints. This problem is a so-called LP relaxation in the form

$$\mathcal{P}_{\mathrm{LP}}: \quad \begin{aligned} \underset{x}{\text{minimize}} \quad & c^T x && (3.2a)\\ \text{subject to} \quad & Ax \leqslant b, && (3.2b)\\ & 0 \leqslant x_i \leqslant 1, \quad \forall i \in \mathcal{B}, && (3.2c)\\ & x_i = 0,\ \forall i \in \mathcal{B}_0,\ x_i = 1,\ \forall i \in \mathcal{B}_1, && (3.2d) \end{aligned}$$

where $\mathcal{B}_0, \mathcal{B}_1 \subseteq \mathcal{B}$ and $\mathcal{B}_0 \cap \mathcal{B}_1 = \emptyset$ are the index sets of binary variables fixed to 0 and 1, respectively. Problem (3.2) is in the form of (2.15), where some (binary) decision variables indexed by $\mathcal{B}_0$ and $\mathcal{B}_1$ have been fixed.

There are several methods presented in the literature for solving MILPs (3.1), including ([15]) cutting plane methods, decomposition methods, logic-based methods, and the methods which are the main focus of this thesis, B&B methods, that will be presented in Section 3.3.

3.2 Mixed-integer quadratic programming

Consider the QP problem (2.21) and let decision variables include both real and binary variables. The problem is then in a form of mixed-integer quadratic program (MIQP) formulated as

$$\mathcal{P}_{\mathrm{MIQP}}: \quad \begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{2} x^T H x + f^T x && (3.3a)\\ \text{subject to} \quad & Ax \leqslant b, && (3.3b)\\ & x_i \in \{0, 1\}, \quad \forall i \in \mathcal{B}, && (3.3c) \end{aligned}$$

where $x = [x_c^T, x_b^T]^T \in \mathbb{R}^{n_c} \times \{0, 1\}^{n_b}$ is the decision variable. The objective function is defined by $H \in \mathbb{S}_{++}^n$, and $f \in \mathbb{R}^n$, and the feasible set is defined by $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. The MIQP problem (3.3) is non-convex due to the presence of integrality constraints (3.3c).

A QP relaxation of the problem (3.3) is defined where the integrality constraints (3.3c) are relaxed to interval constraints, resulting in a problem in the form

$$\mathcal{P}_{\mathrm{QP}}: \quad \begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{2} x^T H x + f^T x && (3.4a)\\ \text{subject to} \quad & Ax \leqslant b, && (3.4b)\\ & 0 \leqslant x_i \leqslant 1, \quad \forall i \in \mathcal{B}, && (3.4c)\\ & x_i = 0,\ \forall i \in \mathcal{B}_0,\ x_i = 1,\ \forall i \in \mathcal{B}_1, && (3.4d) \end{aligned}$$

where $\mathcal{B}_0, \mathcal{B}_1$ are defined as in (3.2). This problem is in the form of (2.21) with some fixed decision variables (indexed by $\mathcal{B}_0$ and $\mathcal{B}_1$).

The methods outlined in Section 3.1 for solving MILPs can also be applied to solve MIQPs. In [33], the authors compared outer approximation methods [28, 32] and generalized Benders decomposition methods [34] with B&B methods, and the conclusion was that B&B is superior to the others for solving MIQPs.

3.3 Branch-and-bound methods

The MILP and MIQP problems are combinatorial, meaning that their optimal solutions correspond to subsets chosen from a finite set [64]. As a result, they are considerably more challenging to solve than LPs and QPs. Due to the combinatorial nature, these problems can essentially be solved by enumeration, in which all possible combinations of the binary variables are generated. For each combination, the corresponding LP/QP problem is solved to compute real variables. To find the optimal solution, the objective function values are compared and the combination providing the smallest cost is chosen as the optimal solution. However, when problems involve many binary variables, the computational burden can become overwhelming due to the exponential growth of the number of possible combinations, which is 2^{n_b} combinations. Therefore, a method is required to find the optimal solution to these problems without explicit enumeration. In this section, the B&B method which is the most commonly used and efficient method to solve MILPs and MIQPs is presented in detail.

3.3.1 Introduction

B&B is often referred to as an algorithm, it might however be more appropriate to consider it as a family of algorithms that share a common core solution process: implicitly enumerating all possible combinations of the problem under consideration. The worst case complexity of B&B, however, grows exponentially with the problem size, and the number of combinations needed for implicit enumeration in B&B is problem dependent. Reference [44] offers an initial overview of the primary B&B algorithm. More in-depth coverage of the B&B solution process can be found in, e.g., [50, 64].

B&B is a general framework to obtain exact solutions of optimization problems that are $\mathcal{NP}$-hard. To handle the combinatorial part of the problem, a binary search tree is used. Each *node* in the search tree corresponds to a relaxation of the problem. An example of a binary search tree is shown in Figure 3.1. At the level l in the tree, one of the relaxed binary-constrained variables is fixed to 0 and 1, resulting into two new subproblems that form two new nodes in the B&B tree. The newly generated nodes are referred to as the *child nodes* of the *parent node*. This procedure is called *branching*. At the top of the tree (level $l = 0$), the *root node* is found where all the binary constraints have been relaxed. At the bottom of the tree (level $l = n_b$), the *leaf nodes* are found in which all binary constraints have been fixed.

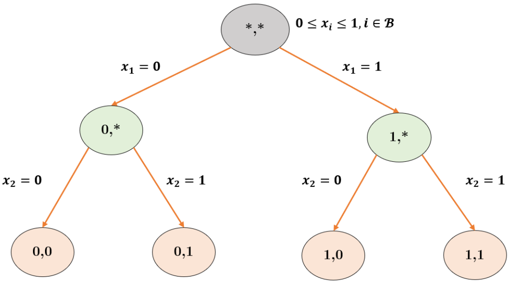


Figure 3.1: *A binary search tree example for two binary variables* x_1 *and* x_2*. Each node is here depicted as an ellipse that contains the values of the binary variables. The symbol* $*$ *denotes that the relaxed binary variable is free to be in* $[0, 1]$.

An important property of the B&B search tree is that the relaxation in a node gives a *lower bound* on the optimal objective function value for the subtree below that node, whereas an integer-feasible solution provides an *upper bound* on the optimal objective function value that is valid in the entire tree. The idea of B&B is to use these lower and upper bounds to possibly limit the size of the search tree and avoid explicitly enumerating all the nodes. This procedure is called *bounding*. The name of the method reflects these two motivating procedures: *branch and bound*.

3.3.2 Branch-and-bound algorithm

In order to describe the B&B algorithm, we first need to define the following notations:

- $\eta \triangleq (\mathcal{B}_0, \mathcal{B}_1)$: A node containing a relaxation, defined in (3.2) for MILPs and in (3.4) for MIQPs. The $\mathcal{B}_0$ and $\mathcal{B}_1$ sets are defined in (3.2d) and are node dependent.
- x^*: The optimal solution to the MILP problem (3.1) or the MIQP problem (3.3).
- J^*: The optimal objective function value.
- $\bar{x}$: The best known integer-feasible solution so far.
- $\bar{J}$: The objective function value of the best known integer-feasible solution so far (the upper bound).
- $\underline{x}$: The optimal solution of a relaxation in a node.
- $\underline{J}$: The objective function value of a relaxation in a node (the lower bound).
- $\mathcal{A}$: The active set at an optimal solution $\underline{x}$.

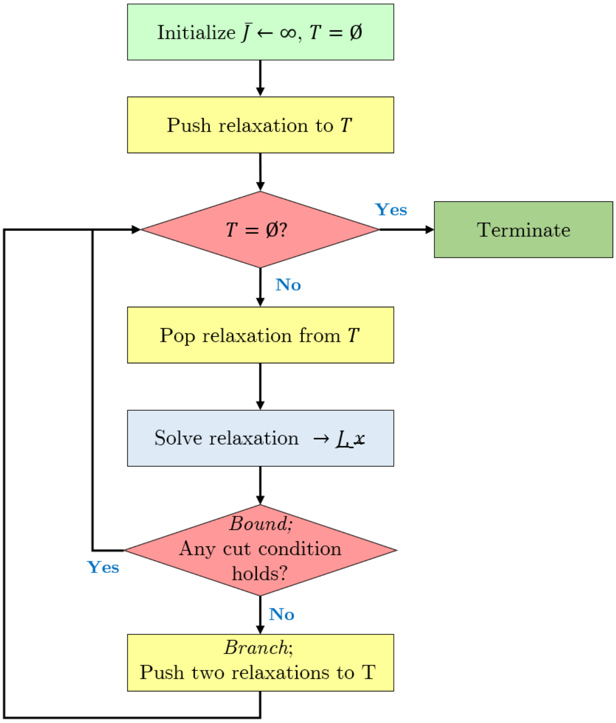

Initialize $\bar{J} \leftarrow \infty$, $T = \emptyset$
Push relaxation to T
$T = \emptyset$?
Yes
Terminate
No
Pop relaxation from T
Solve relaxation $\rightarrow \underline{J}, \underline{x}$
Bound;
Any cut condition
holds?
Yes
No
Branch;
Push two relaxations to T

Figure 3.2: *The flowchart of the B&B algorithm.*

- $\mathcal{T}$: A sorted list implementing a priority queue to store nodes that are generated but not yet explored in the B&B tree.

The flowchart of the B&B algorithm is shown in Figure 3.2. Starting at the root node, all the binary constraints (3.1c) are relaxed to (3.2c) (hence $\mathcal{B}_0 = \mathcal{B}_1 = \emptyset$), resulting in a fully relaxed LP/QP problem that is stored in the list $\mathcal{T}$. At each iteration in the B&B process, a node is selected from $\mathcal{T}$ and the relaxation is solved by an LP/QP solver. In order to solve the LP and QP relaxations, the simplex and dual active-set methods described in Sections 2.2.2 and 2.3.2, respectively, are employed in this thesis. After obtaining the optimal solution $\underline{x}$ and $\underline{J}$ of the relaxation, some conditions are evaluated to possibly bound the tree. If any such condition holds, the tree is cut (pruned) in this node. Otherwise, branching is performed in which one of the relaxed binary variables is selected and fixed to generate two child nodes. New nodes are appended to $\mathcal{T}$ to be processed in the tree at next iterations. The algorithm terminates when there is no unprocessed node left. Algorithm 2 summarizes the B&B algorithm.

The performance of B&B is significantly influenced by three key choices (see Figure 3.3): the *cut conditions*, which are rules that restrict exploration of all nodes in the tree; the *node selection strategy*, which determines the order in which nodes (subproblems) in the tree are explored; and the *branching strategy*, which

Algorithm 2 Branch and bound

```
Input: MILP problem (3.1) or MIQP problem (3.3)
Output: $J^*, x^*$
 1: $\bar{x} \leftarrow \varnothing, \bar{J} \leftarrow \infty, \mathcal{T} \leftarrow \emptyset$
 2: Push $\eta(\emptyset, \emptyset)$ to $\mathcal{T}$
 3: while $\mathcal{T} \neq \emptyset$ do
 4:     Pop $\eta(\mathcal{B}_0, \mathcal{B}_1)$ from $\mathcal{T}$
 5:     $\underline{J}, \underline{x}, \mathcal{A} \leftarrow$ Solve $\eta(\mathcal{B}_0, \mathcal{B}_1)$
 6:     if $\underline{J} \geq \bar{J}$ then
 7:         There exists no feasible solution to the relaxation which is better than $\bar{x}$
 8:     else if all binary variables are active then
 9:         Better integer-feasible solution has been found
10:         $\bar{J} \leftarrow \underline{J}, \bar{x} \leftarrow \underline{x}$
11:     else
12:         Select $k : k \in \mathcal{B}, k \notin (\mathcal{B}_0 \cup \mathcal{B}_1)$
13:         Push $\eta(\mathcal{B}_0 \cup \{k\}, \mathcal{B}_1)$ and $\eta(\mathcal{B}_0, \mathcal{B}_1 \cup \{k\})$ to $\mathcal{T}$
14: return $J^* \leftarrow \bar{J}, x^* \leftarrow \bar{x}$
```

specifies how to select the next relaxed binary variable to fix and generate new subproblems in the tree. More details on each choice is given in what follows.

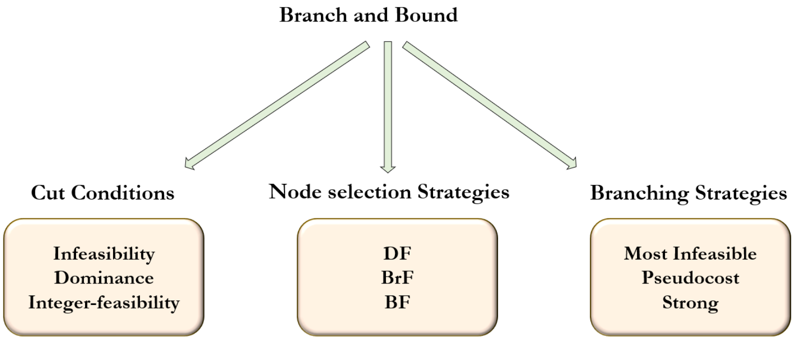


Figure 3.3: *A diagram of the three main B&B components.*

Cut conditions in B&B

The three cut conditions that result in bounding the B&B tree are as follows [64]:

1. *Infeasibility* cut condition;

   This cut condition is satisfied at a node if the relaxation is infeasible, in which $\underline{J} = \infty$. The entire subtree below that node is therefore infeasible, as adding more constraints would not render the problem feasible. Step 6 in Algorithm 2 evaluates this cut condition.

2. *Dominance* cut condition;

   This cut condition is satisfied at a node if the lower bound $\underline{J}$ is *greater* than the upper bound $\bar{J}$. Adding more constraints would not decrease the objective function value, hence, a better objective function value cannot be found in the subtree below that node. Step 6 in Algorithm 2 also examines this cut condition.

3. *Integer-feasibility* cut condition;

   This cut condition is satisfied at a node if the solution to the relaxation is integer feasible. Adding constraints would not decrease the objective function value, hence, an optimal solution for the entire subtree below that node is found. Step 8 in Algorithm 2 evaluates this cut condition.

When the solution to a relaxation in a node satisfies one of these cut conditions, then all nodes in the subtree below that node can be shown to be of no use and can therefore be disregarded, that is, the tree will be cut (pruned) in that node.

Node selection strategies in B&B

The order in which new nodes are inserted into the sorted list $\mathcal{T}$ depends on the choice of node selection strategy. The most common node selection strategies are listed below [64].

- *Depth first* (DF);

  In DF, the next node to be processed is selected as one of the child nodes of the current node. This process is continued until a cut condition for a relaxation holds, and the node is pruned. After pruning, the backtracking starts, which is the procedure of going back toward the root node in searching for a node with an unprocessed child node. As it is possible to prune the tree if an integer feasible solution is found, and as they are more likely to be found deep down in the tree, this search strategy is encouraged.

  Let ρ denote the priority order of a node to be explored in the tree. Then this value for each node in DF is computed as

  $$\rho_{\mathrm{DF}} = \frac{1}{l+1}, \tag{3.5}$$

  where l denotes the level of the node in the search tree.

- *Breadth first* (BrF);

  In BrF, the unprocessed node that was created first is chosen. That is, the priority is the same for all nodes at the same level in the search tree. The priority order of each node is thus assigned to

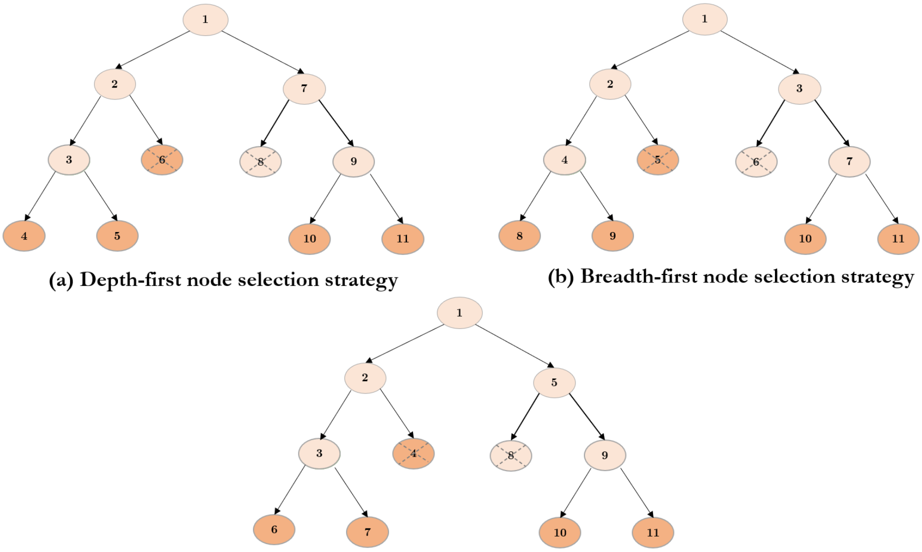
  $$\rho_{\mathrm{BrF}} = l + 1. \tag{3.6}$$

- *Best first* (BF);

  In BF, always the node with the best bound, i.e., the one with the lowest lower bound, is chosen. The advantage of this strategy is that the total number of nodes evaluated in the tree could be minimized. The priority order is thus determined by the current value function that needs to be stored for each node as follows

$$\rho_{\mathrm{BF}} = \underline{J}. \tag{3.7}$$

The generated nodes are ordered and stored in $\mathcal{T}$ at Step 13 in Algorithm 2, based on the value of ρ, such that the node with the highest priority (the lowest value of ρ) is stored in the beginning of the list $\mathcal{T}$. Figure 3.4 demonstrates these node selection strategies, depicting the exploration order of nodes in a small search tree.



(a) Depth-first node selection strategy

(b) Breadth-first node selection strategy

(c) Best-first node selection strategy

Figure 3.4: *B&B search tree with different node selection strategies. Numbers inside nodes indicate exploration order. Nodes shown in the darker color are integer feasible, and the crossed nodes have been pruned.*

Branching strategies in B&B

How to select the next binary variable to branch on at Step 12 in Algorithm 2 depends on the chosen branching strategy. By selecting the most promising variable for branching, the selected branching strategy can efficiently guide the B&B search towards the optimal solution.

Let $\bar{\mathcal{B}} = \{i \in \mathcal{B} : \underline{x}_i \neq \{0, 1\}\}$ denote the candidate list of relaxed binary variables (determined after obtaining the solution $\underline{x}$), and let $s_i \in \mathbb{R}$ denote a score value for the binary variable x_i, $i \in \bar{\mathcal{B}}$, which depends on the chosen branching strategy. After calculating s_i for all $i \in \bar{\mathcal{B}}$, an index k that provides the highest score value,

$$k \leftarrow \text{argmax}_{i \in \bar{\mathcal{B}}} s_i \tag{3.8}$$

is then chosen as the branching variable index at Step 12 of Algorithm 2.

A common branching strategy is the *most infeasible* approach. The idea of this method is to choose the branching variable with the lowest tendency to take binary values, hence, to reduce the infeasibility of the problem as quickly as possible. Therefore, a relaxed binary variable that is closest to 0.5 is always chosen. The assigned score value s_i for each relaxed binary variable in this strategy is formulated as

$$s_i = 0.5 - |x_i - 0.5|, \ i \in \bar{\mathcal{B}}. \tag{3.9}$$

Pseudocost and *Strong* branching are other branching strategies which are more sophisticated. Pseudocost branching determines the variable to branch on by estimating the expected improvement in the objective function value, based on pseudocost values that keep track of the difference in the objective function value when a variable is fixed at different values. Strong branching on the hand, evaluates the impact of fixing each variable in turn and branching on the variable that produces the most significant reduction in the bound. For further details and a comprehensive overview of different branching strategies see [2, 46].

3.3.3 Suboptimal branch-and-bound methods

B&B can solve MILPs and MIQPs efficiently in many cases. When the size of the problem increases, however, the computation time and/or memory space required to find the optimal solution in B&B can sometimes become intractable, not the least for embedded applications on simple hardware. One way to reduce the computation burden is by relaxing the requirement to find a globally optimal solution to instead find a suboptimal solution. In this section, we review three suboptimal strategies that can be used in B&B to find the suboptimal solutions. For a detailed description of these methods, see, e.g., [10, 41, 42].

The ϵ-allowance method

In the ϵ-allowance method, the dominance-cut condition (at Step 6 in Algorithm 2) is relaxed such that the tree is cut in a node if the difference between the relaxation's objective function value $\underline{J}$ and the upper bound $\bar{J}$ is small enough, that is, if $\underline{J} + \epsilon \geq \bar{J}$, where ϵ is a user-defined allowance function. The allowance function can in general take the following form

$$\epsilon = \varepsilon + \varepsilon_r \underline{J}, \tag{3.10}$$

where ε and ε_r are user-defined positive constants. The allowance function provides bounds on the optimal solution as follows [41]

$$\hat{J} - \epsilon \ \leq J^* \leq \ \hat{J}, \tag{3.11}$$

where $\hat{J}$ denotes the (potentially) suboptimal objective function value and J^* denotes the optimal objective function value. The following suboptimality bounds can be obtained for this approach:

$$\varepsilon_r = 0 \rightarrow \hat{J} - J^* \leq \varepsilon, \quad \text{(absolute suboptimality bound)} \tag{3.12a}$$

$$\varepsilon = 0 \ \rightarrow \frac{\hat{J} - J^*}{\hat{J}} \leq \varepsilon_r, \quad \text{(relative suboptimality bound)} \tag{3.12b}$$

where in (3.12b), it is assumed that $\hat{J} > 0$. The properties of this approach are discussed in detail in [41].

The T-cut method

In the T-cut method, the B&B tree is terminated as soon as a predetermined number of nodes (T_0) are decomposed through branching. A motivation to this strategy is that it can be used to, in a rather rough way, limit the computation time. Let T denote the number of node decomposition in B&B. Then, $2T + 1$ nodes are in total explored in the search tree (2 subproblems generated through branching and 1 referring to the root node). The best integer-feasible solution that has been found so far is then outputted as the suboptimal solution. To implement this method in the B&B algorithm, the number of node decomposition is counted and the algorithm terminates once this value reaches the bound T_0. See [42] for more details.

The M-cut method

Define *active nodes* as the nodes stored in the candidate list $\mathcal{T}$ that have not yet been explored in the B&B tree. In the M-cut method, the number of active nodes is limited to a prespecified number (M_0). As a result, the computer memory is restricted to store at most M_0 active nodes in $\mathcal{T}$. To apply this method, when branching the B&B tree, some generated nodes are excluded from $\mathcal{T}$ to guarantee that always $M \leq M_0$, where M denotes the number of elements in $\mathcal{T}$. Which node(s) to be excluded from the list $\mathcal{T}$ is determined by the value of its assigned priority order defined in Section 3.3.2, such that the one(s) with the least priority will be excluded. The properties of this method have been discussed in [42].

3.3.4 Heuristics in branch and bound

Heuristics are procedures that aim to compute feasible solutions computationally inexpensively by utilizing some relevant information. Although these methods do not guarantee finding a feasible solution, they have shown to be relevant supplementary procedures to, e.g., find good feasible solutions at early stages in B&B. There is a wide variety of heuristics proposed in the literature for the B&B methods. Primal heuristics are a category of such techniques that can be used as both start and improvement heuristics. We give a brief overview of these methods in the following, while we refer to [18] for an in-depth discussion of primal heuristics for MILPs.

Start heuristics

The goal in the start heuristic methods is to find an integer-feasible solution, denoted $\hat{x}$, early in the search process. Finding integer-feasible solutions early can result in a useful upper bound that can help to prune nodes in the search tree, therefore, reducing the size of the tree and the overall effort. Let $\underline{x}$ denote an optimal solution obtained after solving a relaxation, e.g., at the root node, and let $\bar{\mathcal{B}}$ denote a candidate list of relaxed binary variables. Most of the start heuristics are applied in the root node, but they might also be called before or after the root node is solved. Some commonly used start heuristics are listed below.

- *Rounding*;

  Rounding heuristics, such as *simple rounding* and *relaxation enforced neighborhood search (RENS)*, try to round relaxed binary variables $\underline{x}_i$, $i \in \bar{\mathcal{B}}$, up or down such that the resulting point ($\hat{x}$) becomes integer feasible. In particular, the RENS method presented in [18] solves a sub mixed-integer problem created from the original problem, where variables, for which the solution $\underline{x}_i$ are binary, are fixed.

- *Diving*;

  Diving method iteratively fixes relaxed binary variables $\underline{x}_i$, $i \in \bar{\mathcal{B}}$ to 0 or 1, and solves the obtained relaxations. The procedure is terminated once infeasibility is detected or an integer-feasible solution is found. Thereby, this method resembles a depth-first node selection strategy of a promising root-leaf path by "diving" in the B&B tree until a hopefully good integer-feasible solution is found. See [18] for more details.

- *Feasibility pump (FP)*;

  The feasibility pump is particularly effective for MILPs where finding a feasible solution is difficult, but where a feasible solution is likely to exist within a relatively small region of the solution space. The idea of the FP method proposed in [30] is to construct two sequences of points that hopefully converge to a feasible solution. One sequence contains LP-feasible solutions that are not necessarily integer-feasible, and the other contains solutions that are not necessarily LP-feasible.

Improvement heuristics

The aim in improvement heuristics is, given one or more integer-feasible solutions $\bar{x}$, to find an improved integer-feasible solution $\hat{x}$ that provides a better objective function value. Some common improvement heuristics include:

- *Local branching*;

  Local branching is a refinement heuristic based on the observation that feasible solutions often have additional solutions in their neighborhood (in terms of the Manhattan distance) [29]. Having an integer-feasible solution $\bar{x}$, the idea is then to find another integer-feasible solution $\hat{x}$ in its neighborhood by applying *soft* *fixing* that requires some of the variables of $\hat{x}$ to take the same value as $\bar{x}$, but not concretely fix any of those.

- *Relaxation induced neighborhood search (RINS)*;

  RINS, proposed in [23], relies on the observation that the best-known integer-feasible so far solution $\bar{x}$ and a solution $\underline{x}$ of a relaxation often have some variables with identical values. To find a solution $\hat{x}$ that provides a better objective function value than $\bar{x}$ in RINS, a sub-MILP is created and solved. This subproblem is generated from the original problem by fixing variables with identical values in $\bar{x}$ and $\underline{x}$ corresponding to the current node. This is different from RENS where it only takes a solution $\underline{x}$ to the relaxation.

3.4 Multi-parametric mixed-integer programming

In this section, mixed-integer optimization that depends on a parameter vector is discussed. Such a problem forms a multi-parametric program as discussed in Section 2.4. An interest in multi-parametric mixed-integer programming stems from its applications in systems theory and optimal control for hybrid systems. For instance, constrained finite-time optimal control problems for hybrid systems can be represented as mathematical programs with the initial state of the system serving as a parameter. Further discussion on this topic can be found in Section 4.3.

We focus on two specific cases of such problems that are particularly relevant to our research in what follows.

3.4.1 Multi-parametric MILP

A mathematical formulation of a multi-parametric MILP (mp-MILP) is given by

$$\begin{aligned} & & \underset{x}{\text{minimize}} \quad & c^T x & (3.13a)\\ & \mathcal{P}_{\text{mpMILP}}(\theta): & \text{subject to} \quad & Ax \leqslant b + W\theta, & (3.13b)\\ & & & x_i \in \{0,1\}, \;\; \forall i \in \mathcal{B}, & (3.13c)\end{aligned}$$

where $x \in \mathbb{R}^{n_c} \times \{0,1\}^{n_b}$, $c \in \mathbb{R}^n$ $A \in \mathbb{R}^{m\times n}$, and $b \in \mathbb{R}^m$. Here, $\theta \in \Theta_0$ is the vector of parameters, and $\Theta_0 \subseteq \mathbb{R}^{n_\theta}$ is the parameter set which is assumed to be polyhedral. By changing the parameter θ, the right-hand side of the constraints (3.13b) are perturbed that results in different MILP problems in the form of (3.1).

The set of feasible parameters for (3.13) is defined as

$$\Theta^* = \{\theta \in \Theta_0 : \Omega(\theta) \neq \emptyset\}, \tag{3.14}$$

where $\Omega(\theta) \triangleq \{x \in \mathbb{R}^n : \; Ax \leqslant b + W\theta, \;\; x_i \in \{0,1\}, \; \forall i \in \mathcal{B}\}$. Denote the (optimal) value function and the optimal solution function to (3.13) by $J^*(\theta)$ and $x^*(\theta)$, respectively. The properties of the mp-MILP solution $x^*(\theta)$ are discussed in detail in, e.g., [19], summarized below.

Theorem 3.2. *Consider the mp-MILP problem* (3.13). *Then, the feasible parameter set Θ^* is not necessarily convex, $J^*(\theta)$ is PWA, but in general, neither convex nor continuous, and the optimizer $x^*(\theta)$ is PWA, but not necessarily continuous.*

Proof: See [19]. □

A convex relaxation of the mp-MILP (3.13) problem is obtained by relaxing the integrality constraints (3.13c) into constraints of the form in (3.2c). This relaxation is in the form of a convex mp-LP problem given by (2.32), where some (binary) decision variables have been fixed.

For a specific $\bar{\theta} \in \Theta_0$, the right-hand side inequality constraint coefficient vector in (3.1) becomes $\bar{b} = b + W\bar{\theta}$, and the problem will be in the form of a (non-parametric) MILP problem (3.1). The mp-MILP problem (3.13) can be solved offline for all $\theta \in \Theta_0$ using a parametric MILP solver, e.g., the one in [26], or online for a specific parameter $\bar{\theta} \in \Theta_0$ using an ordinary MILP solver, e.g., the one presented in Section 3.3.

3.4.2 Multi-parametric MIQP

A mathematical formulation of an mp-MIQP problem is

$$\mathcal{P}_{\text{mpMIQP}}(\theta): \quad \begin{array}{ll} \underset{x}{\text{minimize}} & \frac{1}{2}x^T H x + f^T x + \theta^T f_\theta^T x \qquad (3.15\text{a}) \\ \text{subject to} & Ax \leqslant b + W\theta, \qquad (3.15\text{b}) \\ & x_i \in \{0,1\}, \quad \forall i \in \mathcal{B}. \qquad (3.15\text{c}) \end{array}$$

The mp-MIQP problem is given by $H \in \mathbb{S}_{++}^n$, $f \in \mathbb{R}^n$, $f_\theta \in \mathbb{R}^{n \times n_\theta}$, and the other variables are defined as in (3.13). Changing the parameter $\theta \in \Theta_0$, where $\Theta_0 \subseteq \mathbb{R}^{n_\theta}$ is a polyhedral set, perturbs the linear term in the objective function and the right-hand side of the constraints (3.15b), leading to different MIQPs in the form of (3.3).

The feasible parameter set Θ^* of (3.15b) is defined in (3.14). The properties of the mp-MIQP solution are discussed in detail in, e.g., [19], summarized below.

Theorem 3.3. *Consider the mp-MIQP problem* (3.15). *Then, the feasible parameter set Θ^* is not necessarily convex, $J^*(\theta)$ is PWQ, but in general not continuous, and the optimizer $x^*(\theta)$ is PWA, but not necessarily continuous.*

Proof: See [19]. □

A convex relaxation of mp-MIQP (3.15) is obtained by relaxing the integrality constraints (3.15c) into constraints in the form of (3.4c). This convex relaxation is in a form of an mp-QP problem given in (2.34), where some of the binary decision variables have been fixed.

The mp-MIQPs (3.15) can be solved offline for all $\theta \in \Theta_0$ using a parametric MIQP solver, e.g., as in [10, 27], or online for a specific parameter $\bar{\theta}$ in the parameter set. For a single $\bar{\theta}$, the problem (3.15) simplifies to a (non-parametric) MIQP problem in the form of (3.3) with the linear function $\bar{f} = f + f_{\theta}\bar{\theta}$ in the objective function value and the right-hand side of inequality constraints $\bar{b} = b + W\bar{\theta}$. These problems can be solved online using an ordinary MIQP solver, e.g., the one explained in Section 3.3.

To compute solutions to mp-MILPs (3.13) and mp-MIQPs (3.15), two main approaches have been proposed in the literature: B&B and decomposition methods. In the former method, B&B methods presented in Section 3.3 are employed, in which at each node of the B&B search tree, a relaxation of the form of an mp-LP/mp-QP is solved. In the later method, the MILP/MIQP problem is alternatively decomposed into an mp-LP/mp-QP and an MILP/MIQP subproblem. By iteratively combining the solutions between these two subproblems, the optimal solution to the original problem is found. The references [1, 10] and [26] provide comprehensive details on how to compute parametric solutions to such problems using the B&B and decomposition methods, respectively.

4

Model Predictive Control for Hybrid Systems

Hybrid systems appear in many application areas where both real-valued and logic variables interact in one common framework. This chapter is intended to present an optimal control strategy, namely model predictive control (MPC), for hybrid systems. The chapter starts with an introduction to MPC. A class of hybrid system models that is suitable for solving optimal control problems is then described in Section 4.2. In Section 4.3, optimal control problems for these models are presented. In particular, hybrid dynamical optimization problems can be recast into mixed-integer linear or quadratic programs that are discussed in Section 4.3.

4.1 Model predictive control

One of the primary objectives of control theory is to determine values for the control inputs $u \in \mathbb{R}^{n_u}$ such that the system states $z \in \mathbb{R}^{n_z}$ take the most desirable values. Often, the input and state variables are constrained in various ways, for instance, physical and practical limits on the control inputs and/or system states. An advanced technique to find control inputs while satisfying constraints is MPC.

The concept of MPC dates back to the 1960s [52, 54], and it has appeared in industrial applications from the 1980s [13, 53]. Since then, it has been developed considerably both in academia and in industry. The basic idea behind MPC is to use a mathematical model of the system under control to predict its future behavior over a predefined prediction horizon, thereby also taking into account constraints on states and control inputs [47]. Based on this prediction, MPC then optimizes the control inputs over the finite-time horizon to achieve a desired outcome. A schematic illustration of MPC is shown in Figure 4.1 [4]. The fact that MPC predicts the future behavior of a system and that it can adequately

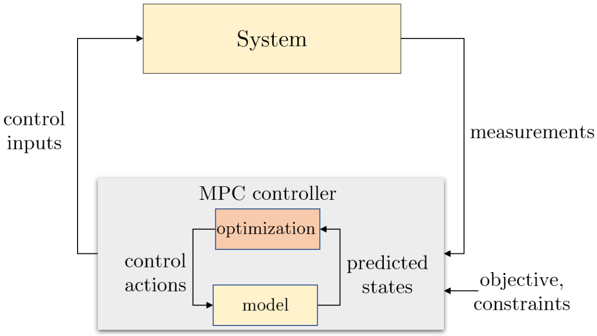


Figure 4.1: *Illustration of model predictive control.*

incorporate different types of constraints make this method an attractive control technique. A reference book and a summary for MPC can be found in [22] and [55], respectively.

MPC consists of some common components, and different approaches to choose each one can result in different formulations. Three main components are:

- A prediction model; to predict future states.
- An objective function; to reflect the desired behavior of the system.
- Constraints; to restrict the system states and control inputs.

A *prediction model* is often described in the following time-invariant form

$$z_{k+1} = f(z_k, u_k), \tag{4.1}$$

where z_k and u_k denote the state and control vectors at time step k. The mapping $f \in \mathbb{R}^{n_z} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_z}$ is often called the dynamics of the system. In order to apply the prediction model in practical settings where controllers are typically implemented, it is necessary to use discrete-time models rather than continuous-time models. Thus, the system in (4.1) is specified in discrete time. A discrete-time model can be obtained by approximating the continuous-time model, for example, through zero-order hold.

The *performance measure* or *objective* from time instant 0 to the time horizon N is defined as follows

$$J(z_0, Z, U) = V(z_N) + \sum_{k=0}^{N-1} l(z_k, u_k), \tag{4.2}$$

where N is the prediction horizon, z_0 is the current measured (estimated) state, and $Z = \{z_1, \ldots, z_N\}$ and $U = \{u_0, \ldots, u_{N-1}\}$ denote the sequence of predicted states and control actions, respectively. The function $V : \mathbb{R}^{n_z} \rightarrow \mathbb{R}$ is called the terminal cost and the function $l : \mathbb{R}^{n_z} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}$ is called the stage cost.

The *constrained finite-time optimal control* (CFTOC) problem to be solved at each time step in MPC is given by

$$
\begin{aligned}
J^*(z_0) = \underset{U}{\text{minimize}} \quad & J(z_0, Z, U) && (4.3a)\\
\text{subject to} \quad & z_{k+1} = f(z_k, u_k), \quad \forall k = 0, \dots, N-1, && (4.3b)\\
& h(z_k, u_k) \leq 0, \qquad \forall k = 0, \dots, N-1, && (4.3c)\\
& z_N \in \mathcal{Z}_f, && (4.3d)\\
& z_0 = z, && (4.3e)
\end{aligned}
$$

where the objective function in (4.3a) is given by (4.2), constraint (4.3b) is the dynamical model of the system, (4.3c) handles constraints on states and control actions, (4.3d) restricts the final states into the terminal set $\mathcal{Z}_f \subseteq \mathbb{R}^{n_z}$, e.g., the origin, and finally (4.3e) defines the initial state z_0 as the current state z. The optimal control sequence U that minimizes (4.3a) is denoted U^*, and the resulting optimal cost $J^*(z_0)$ is called the *value function* [20]. The values of U^* and $J^*(z_0)$ depend on the initial state $z_0 = z$.

An overview of MPC is shown in Algorithm 3. From Step 4, only the first control action u_0^* is applied to the system and the other control actions $u_1^*, \dots, u_{N-1}^*$ are disregarded. At the subsequent time step, a new sequence is computed to replace the previous one. This online replanning provides the desired feedback control feature.

Algorithm 3 Model predictive control

```
1: while true do
2:     Measure (estimate) the current state $z$
3:     Obtain $U^*$ by solving (4.3) with initial state $z_0 = z$
4:     Apply $u = u_0^*$ to the controlled system
```

Lots of the research on MPC has focused on *linear MPC*, where the system dynamics (the prediction model) are assumed to be linear, i.e.,

$$z_{k+1} = Fz_k + Gu_k, \qquad (4.4)$$

where $F \in \mathbb{R}^{n_z \times n_z}$ and $G \in \mathbb{R}^{n_z \times n_u}$. In addition, the stage and terminal cost functions in (4.2) for linear MPC take the form of the 1-, 2-, or ∞-norm. The CFTOC problem (4.3) to be solved is then rewritten as

$$
\begin{aligned}
J^*(z_0) = \underset{U}{\text{minimize}} \quad & J(z_0, Z, U) && (4.5a)\\
\text{subject to} \quad & z_{k+1} = Fz_k + Gu_k, \quad \forall k = 0, \dots, N-1, && (4.5b)\\
& A_z z_k + A_u u_k \leq b, \quad \forall k = 0, \dots, N-1, && (4.5c)\\
& A_f z_N \leq b_f, && (4.5d)\\
& z_0 = z, && (4.5e)
\end{aligned}
$$

with the objective function $J(z_0, U)$ given in the form of (4.6) or (4.7). Algorithm 3 with the optimization problem (4.5) to be solved at Step 3 is known as Linear MPC.

4.1.1 Optimal control, $1/\infty$-norm case

If the 1-norm or ∞-norm is used in the objective function (4.5a), then $V(z_N) = \|Pz_N\|_p$ and $l(z_k, u_k) = \|Qz_k\|_p + \|Ru_k\|_p$, where $p = 1$ or ∞. Here, $P \in \mathbb{S}_+^{n_z}$, $Q \in \mathbb{S}_+^{n_z}$, and $R \in \mathbb{S}_{++}^{n_u}$ are weight matrices. The performance measure (4.2) is then given by

$$J(z_0, Z, U) = \|Pz_N\|_p + \sum_{k=0}^{N-1} \Big(\|Qz_k\|_p + \|Ru_k\|_p\Big). \tag{4.6}$$

The optimal control problem (4.5) with the performance measure (4.6) can be expressed as an mp-LP problem in the form of (2.32). The idea behind reformulating (4.5) into an mp-LP is to consider the current state as the parameter vector, i.e., $z = \theta$, and solve the resulting mp-LP for all possible parameters. This allows to make the use of a wide variety of optimization methods and software available to solve LPs and mp-LPs. This approach is known as explicit MPC [17], and from Theorem 2.16, results in a solution $x^*(\theta)$ which is a PPWA function, and a value function $J^*(\theta)$ which is convex PPWA over a polyhedral partition of the parameter space. See, e.g., [16, 17, 61] for extensive details.

4.1.2 Optimal control, 2-norm case

If the Euclidean (2-) norm is used in the objective function (4.5a), then $V(z_N) = z_N^T P z_N$ and $l(z_k, u_k) = z_k^T Q z_k + u_k^T R u_k$, where $P \in \mathbb{S}_+^{n_z}$, $Q \in \mathbb{S}_+^{n_z}$, and $R \in \mathbb{S}_{++}^{n_u}$. The performance measure (4.2) is then rewritten as

$$J(z_0, Z, U) = z_N^T P z_N + \sum_{k=0}^{N-1} \Big(z_k^T Q z_k + u_k^T R u_k\Big). \tag{4.7}$$

Common approaches such as sparse and condensed formulations can be used to cast the optimal control problem (4.5) for linear MPC with the quadratic performance measure (4.7) into an mp-QP in the form of (2.34) [8, 45]. Such reformulation, again, enables us to use a broad range of optimization solvers and software available to solve QPs and mp-QPs. From Theorem 2.17, the PPWA solution $x^*(\theta)$ and PPWQ value function $J^*(\theta)$ over the polyhedral partition of the parameter space are obtained from solving the resulting mp-QP.

4.2 Models of hybrid systems

MPC can be used to handle a special class of systems known as hybrid systems. Hybrid systems arise in a large number of applications where discrete-valued signals, such as logic decision-making, are combined with real-valued signals that originate from physical processes. Different types of models can be used to describe these systems. In [3, 35], some modeling frameworks for these systems are provided.

A popular modeling class of hybrid systems is mixed logical dynamical (MLD) systems. MLD is a general model and can describe a wide range of systems. The focus in this thesis is on discrete-time systems, which will be introduced in the following subsection.

4.2.1 Mixed logical dynamical systems

MLD systems proposed in [15] is a common model representation of hybrid systems in the MPC scheme. These systems consist of a collection of linear difference equations involving both real and binary variables and a set of linear inequality constraints given by the following relations

$$z_{k+1} = Fz_k + G_1 u_k + G_2 \delta_k + G_3 w_k, \tag{4.8a}$$

$$E_1 \delta_k + E_2 w_k \leq E_3 u_k + E_4 z_k + e_5, \tag{4.8b}$$

where the system state $z_k \in \mathbb{R}^{n_{z_c}} \times \{0,1\}^{n_{z_b}}$ and the control signal $u_k \in \mathbb{R}^{n_{u_c}} \times \{0,1\}^{n_{u_b}}$ at time step k collect both continuous and binary values. Here, $\delta_k \in \{0,1\}^{n_{\delta_b}}$ and $w_k \in \mathbb{R}^{n_w}$ are the binary and continuous auxiliary variables at time step k, respectively, and $F, G_1, G_2, G_3, E_1, E_2, E_3, E_4$ and e_5 are matrices of appropriate dimensions. Other constraints on continuous and logical variables of states and control signals, such as $z_k \in \mathcal{Z}$ and $u_k \in \mathcal{U}$, can also be incorporated into (4.8b), demonstrating the generality of the MLD framework.

In [37], it is shown that under some assumptions, the MLD system is equivalent to other model classes of hybrid systems such as linear complementarity systems [38], extended linear complementarity systems [25], piecewise affine systems [60], and max-min-plus-scaling systems.

4.3 Optimal control of hybrid systems

Most of the methods used to control hybrid systems rely on optimal control principles, which involve finding the control inputs that minimize a cost function subject to system constraints [20]. To solve such optimization problems, various techniques such as dynamic programming, MPC, and hybrid control theory can be used. In particular, MPC has been successfully applied to a variety of hybrid systems, including automotive systems, power systems, and chemical processes. In this section, we focus on applying MPC for discrete-time hybrid systems with linear or quadratic performance measures.

Consider the MLD models (4.8) of hybrid systems. Define the performance measure

$$J_{\mathrm{MLD}}(z_0, Z, U, \Delta, W) = V_h(z_N) + \sum_{k=0}^{N-1} l_h(z_k, u_k, \delta_k, w_k), \tag{4.9}$$

where N is the prediction horizon, $V_h : \mathbb{R}^{n_z} \to \mathbb{R}$ is the terminal cost function, and $l_h : \mathbb{R}^{n_z} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_\delta} \times \mathbb{R}^{n_w} \to \mathbb{R}$ is the stage cost function. Here, $Z = \{z_1, \ldots, z_N\}$, $U = \{u_0, \ldots, u_{N-1}\}$, $\Delta = \{\delta_0, \ldots, \delta_{N-1}\}$ and $W = \{w_0, \ldots, w_{N-1}\}$ are the sequence of predicted states, control actions, and binary and continuous auxiliary variables, respectively. The CFTOC problem to be solved in MPC for these systems is then written as [20]

$$
\begin{aligned}
J^*_{\mathrm{MLD}}(z_0) = \underset{U,\Delta,W}{\text{minimize}} \quad & J_{\mathrm{MLD}}(z_0, Z, U, \Delta, W) && \text{(4.10a)}\\
\text{subject to} \quad & z_{k+1} = Fz_k + G_1 u_k + G_2 \delta_k + G_3 w_k, \ \forall k = 0, \dots, N-1, && \text{(4.10b)}\\
& E_1 \delta_k + E_2 w_k \leq E_3 u_k + E_4 z_k + e_5, \quad \forall k = 0, \dots, N-1, && \text{(4.10c)}\\
& z_N \in \mathcal{Z}_f, && \text{(4.10d)}\\
& z_0 = z, && \text{(4.10e)}
\end{aligned}
$$

where constraints (4.10b) and (4.10c) are the MLD system (4.8), (4.10d) limits the system state at the end of the horizon, and (4.10e) defines the initial state. The terminal set $\mathcal{Z}_f \subseteq \mathbb{R}^{n_z}$ in (4.10d) is a compact polyhedral set that we want the system state to reach at the end of horizon of length N. Two common forms of the cost function $J_{\mathrm{MLD}}(z_0, Z, U, \Delta, W)$ in (4.9) are presented in the following subsections.

The algorithm shown in Algorithm 3, in which the optimization problem (4.10) is solved at Step 3, is called hybrid MPC. The discontinuous nature of the solution space due to the presence of binary variables poses an extra challenge to solve such problems.

4.3.1 Hybrid optimal control, $1/\infty$-norm case

If the 1-norm or ∞-norm is used in (4.9), the performance measure is in the form

$$
J_{\mathrm{MLD}}(z_0, Z, U, \Delta, W) = \|P_h z_N\|_p + \sum_{k=0}^{N-1} \Big(\|Q_z z_k\|_p + \|R_h u_k\|_p + \|Q_\delta \delta_k\|_p + \|Q_w w_k\|_p \Big), \tag{4.11}
$$

where $p = 1$ or ∞. Here, $P_h \in \mathbb{S}_+^{n_z}$, $Q_z \in \mathbb{S}_+^{n_z}$, $R_h \in \mathbb{S}_{++}^{n_u}$, $Q_\delta \in \mathbb{S}_+^{n_\delta}$, and $Q_w \in \mathbb{S}_+^{n_w}$ are weight matrices.

The optimal control problem (4.10) with the performance measure (4.11) can be recast as an mp-MILP problem in the form of (3.13) [20]. The idea (as mentioned in Section 4.1.1) is to consider the current state z as the parameter vector θ. The solution can then be computed offline by employing an mp-MILP solver. See Section 3.4.1. Thereby, a control law with the properties stated in Theorem 3.2 is obtained that describes the control inputs as a function of the measured (estimated) states.

4.3.2 Hybrid optimal control, 2-norm case

If the Euclidean (2–) norm is used in (4.9), the performance measure can be obtained as

$$
J_{\mathrm{MLD}}(z_0, Z, U, \Delta, W) = z_N^T P_h z_N + \sum_{k=0}^{N-1} \Big(z_k^T Q_z z_k + u_k^T R_h u_k + \delta_k^T Q_\delta \delta_k + w_k^T Q_w w_k \Big), \tag{4.12}
$$

with the weight matrices defined as in (4.11).

The optimal control problem (4.10) with quadratic performance measure (4.12) can be recast as an mp-MIQP problem in the form of (3.15) [20]. Solving the resulting mp-MIQP by an mp-MIQP solver (outlined in Section 3.4.2) results in a control input as a function of the states satisfying the characteristics stated in Theorem 3.3.

5 Concluding Remarks

This chapter concludes the first section of the thesis by providing a summary of the key contributions presented in Part II, along with outlining some potential areas for future research to extend or improve the results herein.

5.1 Summary of contributions

In Paper A, the problem of analyzing the computational complexity of B&B-based MIQP solvers is addressed. In particular, an algorithm is presented to compute a useful upper bound on the worst-case computational complexity for solving any possible MIQP from a particular mp-MIQP. Recent methods for exact complexity certification of active-set QP methods, e.g., [5], can be employed in the proposed certification algorithm, enabling also taking into account the complexity originating from solving the relaxations in B&B nodes. Even though the main focus in this work has been a complexity measure in terms of the accumulated number of QP iterations and the number of nodes in the B&B tree, there is freedom in this choice and alternatives such as the number of floating-point operations (flops) could also be considered. This knowledge enables us to compute relevant complexity bounds on the worst-case complexity of MIQP solvers based on B&B, which is of significant importance, e.g., in the context of real-time MPC for hybrid systems.

In Paper B, we extend the complexity certification framework presented in Paper A to also certify the computational complexity of B&B-based MILP solvers. The proposed framework can exactly determine the complexity measure in terms of, e.g., the total number of nodes and linear systems of equations (LP iterations). Furthermore, we develop the proposed algorithm to consider different node exploration strategies, including best-first. As the best node selection strategy is generally problem dependent, the certification algorithm can, hence, be used to rigorously identify the strategy that leads to the least number of iterations and ultimately the lowest computation time for solving the problem. This could help in improving the speed of execution of, e.g., embedded real-time MPC. Additionally, the proposed complexity certification framework is extended to consider warm-starting of the inner (LP) solvers in the B&B process.

When the size of the MILP problems increases, the computation time and/or memory space required to find the optimal solution in B&B can sometimes become intractable. To reduce the computational burden, the requirement to find a globally optimal solution can be relaxed to instead find a suboptimal solution. In Paper C, we extend the result from Paper B to exactly certify the computational complexity of suboptimal B&B methods. To that end, three standard suboptimal strategies to reduce the computational complexity of B&B for the price of suboptimality are considered. The proposed method in this work not only introduces the possibility of computing exact bounds on the computational complexity, but also exactly how suboptimal the computed solution will be. Therefore, it provides a tool to rigorously analyze the trade-off between complexity and suboptimality, which has high relevance in hybrid MPC.

In Paper D, we extend the complexity certification framework in Paper B to tailor the proposed method also for the B&B-based MILP solvers that include heuristics. With that aim, we present and analyze parametric versions of commonly used start heuristics known from high-performing MILP solvers, with the purpose of certifying the computational complexity of their online counterparts. In particular, three start heuristics: RENS, diving, and (objective) feasibility pump are considered. The proposed certification algorithms for the considered heuristics are then integrated into the B&B certification framework to be able to determine the worst-case accumulated number of LP iterations, nodes, and/or flops of B&B schemes for MILP. The result from this work can additionally be used to determine whether a considered start heuristic method helps in speeding up the B&B process, and exactly to what extent.

5.2 Future work

In this section, some ideas for future work and possible extensions of the contributions that are presented in this thesis are summarized.

A unifying complexity certification framework

A possible extension of the results in Papers A and B is to unify the presented complexity certification frameworks of the B&B methods for MILPs and MIQPs. To that end, the proposed method in Paper A needs to be extended so that it could either handle quadratic regions to obtain exact complexity measures or provide tighter complexity measures by applying, e.g., over- and under-approximation of the resulting quadratic regions. Different branching strategies such as most infeasible branching could also be integrated into the proposed certification framework.

Improvement heuristics

In Paper D, start heuristics were integrated into the certification framework. It is also interesting to investigate if and how the improvement heuristic methods such as local branching and RINS described in Section 3.3.4 can be integrated into the B&B certification scheme, which will be considered in future work.

Real-time certification

An interesting future research direction is to try to extend the proposed framework to generate exact or very good bounds on the worst-case execution time for hybrid MPC considered in this research. This was recently introduced for linear MPC in [7]. As described in [7], in order to move from iteration certificates to execution-time certificates, it will be necessary to shift from an algorithmic viewpoint to a programmatic viewpoint. That is, to take into consideration the implementation of the algorithm in code, how this code is compiled, and on which hardware the compiled machine code is executed.

High-performance computing

An advantage of the certification framework is that the complexity measures can be computed solely offline. In addition, the proposed algorithm is highly suitable for parallelization, since different parts of the parameter space are independent from each other. Therefore, the algorithm can be parallelized through, e.g., domain decomposition. This involves splitting the parameter space into distinct regions and applying the certification method to each one. Thus, it is possible to employ high-performance computing to certify more challenging problems with larger problem sizes within a reasonable time.

Optimizing the optimizer

There are several key choices and degrees of freedom in the B&B methods, as highlighted in Section 3.3. The best selections among these options are usually problem dependent. An interesting research direction is, hence, to use the results from the certification framework to investigate which particular choices provide the best worst-case performance when MILP and MIQP instances from particular mp-MILP and mp-MIQP are solved.

Bibliography

[1] Joaquin Acevedo and Efstratios N Pistikopoulos. A multiparametric programming approach for linear process engineering problems under uncertainty. *Industrial & Engineering Chemistry Research*, 36(3):717–728, 1997.

[2] Tobias Achterberg, Thorsten Koch, and Alexander Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, 2005.

[3] P.J. Antsaklis. Special issue on hybrid systems: Theory and applications a brief introduction to the theory and applications of hybrid systems. *Proceedings of the IEEE*, 88(7):879–887, 2000. doi: 10.1109/JPROC.2000.871299.

[4] Daniel Arnström. *Real-time certified MPC: Reliable active-set QP solvers.* PhD thesis, Linköping University Electronic Press, 2023.

[5] Daniel Arnström and Daniel Axehill. A unifying complexity certification framework for active-set methods for convex quadratic programming. *IEEE Transactions on Automatic Control*, 67(6):2758–2770, 2022.

[6] Daniel Arnström, Alberto Bemporad, and Daniel Axehill. A dual active-set solver for embedded quadratic programming using recursive LDL^T updates. *IEEE Transactions on Automatic Control*, 67(8):4362–4369, 2022.

[7] Daniel Arnström, David Broman, and Daniel Axehill. Exact worst-case execution-time analysis for implicit model predictive control. *arXiv preprint arXiv:2304.11576*, 2023.

[8] Daniel Axehill. Controlling the level of sparsity in MPC. *Systems & Control Letters*, 76:1–7, 2015.

[9] Daniel Axehill and Anders Hansson. A dual gradient projection quadratic programming algorithm tailored for model predictive control. In *2008 47th IEEE Conference on Decision and Control*, pages 3057–3064. IEEE, 2008.

[10] Daniel Axehill, Thomas Besselmann, Davide Martino Raimondo, and Manfred Morari. A parametric branch and bound approach to suboptimal explicit hybrid MPC. *Automatica*, 50(1):240–246, 2014.

[11] Bernd Bank, Jürgen Guddat, Diethard Klatte, Bernd Kummer, and Klaus Tammer. *Non-linear parametric optimization*. Springer, 1983.

[12] Roscoe A Bartlett and Lorenz T Biegler. QPSchur: A dual, active-set, schur-complement method for large-scale and structured convex quadratic programming. *Optimization and Engineering*, 7:5–32, 2006.

[13] Margret Bauer and Ian K Craig. Economic assessment of advanced process control–a survey and framework. *Journal of Process Control*, 18(1):2–18, 2008.

[14] Alberto Bemporad. A quadratic programming algorithm based on nonnegative least squares with applications to embedded model predictive control. *IEEE Transactions on Automatic Control*, 61(4):1111–1116, 2015.

[15] Alberto Bemporad and Manfred Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[16] Alberto Bemporad, Francesco Borrelli, Manfred Morari, et al. Model predictive control based on linear programming – the explicit solution. *IEEE Transactions on Automatic Control*, 47(12):1974–1985, 2002.

[17] Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

[18] Timo Berthold. *Primal heuristics for mixed integer programs*. PhD thesis, Zuse Institute Berlin (ZIB), 2006.

[19] Francesco Borrelli. *Constrained optimal control of linear and hybrid systems*, volume 290. Springer, 2003.

[20] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[21] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[22] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer Science & Business Media, 2013.

[23] Emilie Danna, Edward Rothberg, and Claude Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, 102:71–90, 2005.

[24] George Dantzig. *Linear programming and extensions*. Princeton university press, 1963.

[25] Bart De Schutter and Bart De Moor. The extended linear complementarity problem and the modeling and analysis of hybrid systems. In *Hybrid Systems V*, pages 70–85. Springer, 1999.

[26] Vivek Dua and Efstratios N Pistikopoulos. An algorithm for the solution of multiparametric mixed integer linear programming problems. *Annals of Operations Research*, 99(1):123–139, 2000.

[27] Vivek Dua, Nikolaos A Bozinis, and Efstratios N Pistikopoulos. A multiparametric programming approach for mixed-integer quadratic engineering problems. *Computers & Chemical Engineering*, 26(4-5):715–733, 2002.

[28] Marco A Duran and Ignacio E Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36:307–339, 1986.

[29] Matteo Fischetti and Andrea Lodi. Repairing MIP infeasibility through local branching. *Computers & operations research*, 35(5):1436–1445, 2008.

[30] Matteo Fischetti, Fred Glover, and Andrea Lodi. The feasibility pump. *Mathematical Programming*, 104:91–104, 2005.

[31] Roger Fletcher. A general quadratic programming algorithm. *IMA Journal of Applied Mathematics*, 7(1):76–91, 1971.

[32] Roger Fletcher and Sven Leyffer. Solving mixed integer nonlinear programs by outer approximation. *Mathematical Programming*, 66:327–349, 1994.

[33] Roger Fletcher and Sven Leyffer. Numerical experience with lower bounds for MIQP branch-and-bound. *SIAM Journal on Optimization*, 8(2):604–616, 1998.

[34] Arthur M Geoffrion. Generalized benders decomposition. *Journal of Optimization Theory and Applications*, 10:237–260, 1972.

[35] Rafal Goebel, Ricardo G Sanfelice, and Andrew R Teel. Hybrid dynamical systems. *IEEE Control Systems Magazine*, 29(2):28–93, 2009.

[36] Donald Goldfarb and Ashok Idnani. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming*, 27 (1):1–33, 1983.

[37] Wilhemus PMH Heemels, Bart De Schutter, and Alberto Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091, 2001.

[38] WPMH Heemels, Johannes M Schumacher, and S Weiland. Linear complementarity systems. *SIAM Journal on Applied Mathematics*, 60(4):1234–1269, 2000.

[39] Martin Herceg, Michal Kvasnica, Colin N Jones, and Manfred Morari. Multiparametric toolbox 3.0. In *2013 European control conference (ECC)*, pages 502–510. IEEE, 2013.

[40] Michael Hintermüller, Kazufumi Ito, and Karl Kunisch. The primal-dual active set strategy as a semismooth Newton method. *SIAM Journal on Optimization*, 13(3):865–888, 2002.

[41] Toshihide Ibaraki. Computational efficiency of approximate branch-and-bound algorithms. *Mathematics of Operations Research*, 1(3):287–298, 1976.

[42] Toshihide Ibaraki, Shojiro Muro, Takeshi Murakami, and Toshiharu Hasegawa. Using branch-and-bound algorithms to obtain suboptimal solutions. *Zeitschrift für Operations-Research*, 27(1):177–202, 1983.

[43] Colin N Jones, M Barić, and Manfred Morari. Multiparametric linear programming with applications to control. *European Journal of Control*, 13 (2-3):152–170, 2007.

[44] Eugene L Lawler and David E Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719, 1966.

[45] Bernt Lie, Marta D Díez, and Tor A Hauge. A comparison of implementation strategies for MPC. 2005.

[46] Jeff T Linderoth and Martin WP Savelsbergh. A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing*, 11(2):173–187, 1999.

[47] David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[48] John A Nelder and Roger Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

[49] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[50] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: Algorithms and complexity*. Courier Corporation, 1998.

[51] Panagiotis Patrinos and Alberto Bemporad. An accelerated dual gradient-projection algorithm for embedded linear model predictive control. *IEEE Transactions on Automatic Control*, 59(1):18–33, 2013.

[52] AI Propoi. Application of linear programming methods for the synthesis of automatic sampled-data systems. *Avtomat. i Telemekh*, 24(7):912–920, 1963.

[53] S Joe Qin and Thomas A Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764, 2003.

[54] Marshall D Rafal and William F Stevens. Discrete dynamic optimization applied to on-line optimal control. *AIChE Journal*, 14(1):85–91, 1968.

[55] James B Rawlings. Tutorial overview of model predictive control. *IEEE Control Systems Magazine*, 20(3):38–52, 2000.

[56] Shamisa Shoja and Daniel Axehill. Exact complexity certification of start heuristics in branch-and-bound methods for mixed-integer linear programming. *Submitted to the 62nd IEEE Conference on Decision and Control (CDC)*, 2023.

[57] Shamisa Shoja and Daniel Axehill. Exact complexity certification of suboptimal branch-and-bound algorithms for mixed-integer linear programming. *Accepted at the 22nd IFAC World Congress*, 2023.

[58] Shamisa Shoja, Daniel Arnström, and Daniel Axehill. Exact complexity certification of a standard branch and bound method for mixed-integer linear programming. In *Proceedings of the 61st IEEE Conference on Decision and Control (CDC)*, pages 6298–6305, 2022. doi: 10.1109/CDC51059.2022.9992451.

[59] Shamisa Shoja, Daniel Arnström, and Daniel Axehill. Overall complexity certification of a standard branch and bound method for mixed-integer quadratic programming. In *Proceedings of 2022 American Control Conference (ACC)*, pages 4957–4964, 2022. doi: 10.23919/ACC53348.2022.9867176.

[60] Eduardo Sontag. Nonlinear regulation: The piecewise linear approach. *IEEE Transactions on Automatic Control*, 26(2):346–358, 1981.

[61] Petter Tøndel, Tor Arne Johansen, and Alberto Bemporad. An algorithm for multi-parametric quadratic programming and explicit MPC solutions. *Automatica*, 39(3):489–497, 2003.

[62] Robert J Vanderbei et al. *Linear programming*. Springer, 2020.

[63] Yang Wang and Stephen Boyd. Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology*, 18(2):267–278, 2009.

[64] Laurence A Wolsey. *Integer programming*. John Wiley & Sons, 2020.

[65] Stephen J Wright. *Primal-dual interior-point methods*. SIAM, 1997.

Part II

Publications

Papers

The papers associated with this thesis have been removed for copyright reasons. For more details about these see:

https://doi.org/10.3384/9789180752718

Licentiate Theses
Division of Automatic Control
Linköping University

P. Andersson: Adaptive Forgetting through Multiple Models and Adaptive Control of Car Dynamics. Thesis No. 15, 1983.

B. Wahlberg: On Model Simplification in System Identification. Thesis No. 47, 1985.

A. Isaksson: Identification of Time Varying Systems and Applications of System Identification to Signal Processing. Thesis No. 75, 1986.

G. Malmberg: A Study of Adaptive Control Missiles. Thesis No. 76, 1986.

S. Gunnarsson: On the Mean Square Error of Transfer Function Estimates with Applications to Control. Thesis No. 90, 1986.

M. Viberg: On the Adaptive Array Problem. Thesis No. 117, 1987.

K. Ståhl: On the Frequency Domain Analysis of Nonlinear Systems. Thesis No. 137, 1988.

A. Skeppstedt: Construction of Composite Models from Large Data-Sets. Thesis No. 149, 1988.

P. A. J. Nagy: MaMiS: A Programming Environment for Numeric/Symbolic Data Processing. Thesis No. 153, 1988.

K. Forsman: Applications of Constructive Algebra to Control Problems. Thesis No. 231, 1990.

I. Klein: Planning for a Class of Sequential Control Problems. Thesis No. 234, 1990.

F. Gustafsson: Optimal Segmentation of Linear Regression Parameters. Thesis No. 246, 1990.

H. Hjalmarsson: On Estimation of Model Quality in System Identification. Thesis No. 251, 1990.

S. Andersson: Sensor Array Processing; Application to Mobile Communication Systems and Dimension Reduction. Thesis No. 255, 1990.

K. Wang Chen: Observability and Invertibility of Nonlinear Systems: A Differential Algebraic Approach. Thesis No. 282, 1991.

J. Sjöberg: Regularization Issues in Neural Network Models of Dynamical Systems. Thesis No. 366, 1993.

P. Pucar: Segmentation of Laser Range Radar Images Using Hidden Markov Field Models. Thesis No. 403, 1993.

H. Fortell: Volterra and Algebraic Approaches to the Zero Dynamics. Thesis No. 438, 1994.

T. McKelvey: On State-Space Models in System Identification. Thesis No. 447, 1994.

T. Andersson: Concepts and Algorithms for Non-Linear System Identifiability. Thesis No. 448, 1994.

P. Lindskog: Algorithms and Tools for System Identification Using Prior Knowledge. Thesis No. 456, 1994.

J. Plantin: Algebraic Methods for Verification and Control of Discrete Event Dynamic Systems. Thesis No. 501, 1995.

J. Gunnarsson: On Modeling of Discrete Event Dynamic Systems, Using Symbolic Algebraic Methods. Thesis No. 502, 1995.

A. Ericsson: Fast Power Control to Counteract Rayleigh Fading in Cellular Radio Systems. Thesis No. 527, 1995.

M. Jirstrand: Algebraic Methods for Modeling and Design in Control. Thesis No. 540, 1996.

K. Edström: Simulation of Mode Switching Systems Using Switched Bond Graphs. Thesis No. 586, 1996.

J. Palmqvist: On Integrity Monitoring of Integrated Navigation Systems. Thesis No. 600, 1997.

A. Stenman: Just-in-Time Models with Applications to Dynamical Systems. Thesis No. 601, 1997.

M. Andersson: Experimental Design and Updating of Finite Element Models. Thesis No. 611, 1997.

U. Forssell: Properties and Usage of Closed-Loop Identification Methods. Thesis No. 641, 1997.

M. Larsson: On Modeling and Diagnosis of Discrete Event Dynamic systems. Thesis No. 648, 1997.

N. Bergman: Bayesian Inference in Terrain Navigation. Thesis No. 649, 1997.

V. Einarsson: On Verification of Switched Systems Using Abstractions. Thesis No. 705, 1998.

J. Blom, F. Gunnarsson: Power Control in Cellular Radio Systems. Thesis No. 706, 1998.

P. Spångéus: Hybrid Control using LP and LMI methods – Some Applications. Thesis No. 724, 1998.

M. Norrlöf: On Analysis and Implementation of Iterative Learning Control. Thesis No. 727, 1998.

A. Hagenblad: Aspects of the Identification of Wiener Models. Thesis No. 793, 1999.

F. Tjärnström: Quality Estimation of Approximate Models. Thesis No. 810, 2000.

C. Carlsson: Vehicle Size and Orientation Estimation Using Geometric Fitting. Thesis No. 840, 2000.

J. Löfberg: Linear Model Predictive Control: Stability and Robustness. Thesis No. 866, 2001.

O. Härkegård: Flight Control Design Using Backstepping. Thesis No. 875, 2001.

J. Elbornsson: Equalization of Distortion in A/D Converters. Thesis No. 883, 2001.

J. Roll: Robust Verification and Identification of Piecewise Affine Systems. Thesis No. 899, 2001.

I. Lind: Regressor Selection in System Identification using ANOVA. Thesis No. 921, 2001.

R. Karlsson: Simulation Based Methods for Target Tracking. Thesis No. 930, 2002.

P.-J. Nordlund: Sequential Monte Carlo Filters and Integrated Navigation. Thesis No. 945, 2002.

M. Östring: Identification, Diagnosis, and Control of a Flexible Robot Arm. Thesis No. 948, 2002.

C. Olsson: Active Engine Vibration Isolation using Feedback Control. Thesis No. 968, 2002.

J. Jansson: Tracking and Decision Making for Automotive Collision Avoidance. Thesis No. 965, 2002.

N. Persson: Event Based Sampling with Application to Spectral Estimation. Thesis No. 981, 2002.

D. Lindgren: Subspace Selection Techniques for Classification Problems. Thesis No. 995, 2002.

E. Geijer Lundin: Uplink Load in CDMA Cellular Systems. Thesis No. 1045, 2003.

M. Enqvist: Some Results on Linear Models of Nonlinear Systems. Thesis No. 1046, 2003.

T. Schön: On Computational Methods for Nonlinear Estimation. Thesis No. 1047, 2003.

F. Gunnarsson: On Modeling and Control of Network Queue Dynamics. Thesis No. 1048, 2003.

S. Björklund: A Survey and Comparison of Time-Delay Estimation Methods in Linear Systems. Thesis No. 1061, 2003.

M. Gerdin: Parameter Estimation in Linear Descriptor Systems. Thesis No. 1085, 2004.

A. Eidehall: An Automotive Lane Guidance System. Thesis No. 1122, 2004.

E. Wernholt: On Multivariable and Nonlinear Identification of Industrial Robots. Thesis No. 1131, 2004.

J. Gillberg: Methods for Frequency Domain Estimation of Continuous-Time Models. Thesis No. 1133, 2004.

G. Hendeby: Fundamental Estimation and Detection Limits in Linear Non-Gaussian Systems. Thesis No. 1199, 2005.

D. Axehill: Applications of Integer Quadratic Programming in Control and Communication. Thesis No. 1218, 2005.

J. Sjöberg: Some Results On Optimal Control for Nonlinear Descriptor Systems. Thesis No. 1227, 2006.

D. Törnqvist: Statistical Fault Detection with Applications to IMU Disturbances. Thesis No. 1258, 2006.

H. Tidefelt: Structural algorithms and perturbations in differential-algebraic equations. Thesis No. 1318, 2007.

S. Moberg: On Modeling and Control of Flexible Manipulators. Thesis No. 1336, 2007.

J. Wallén: On Kinematic Modelling and Iterative Learning Control of Industrial Robots. Thesis No. 1343, 2008.

J. Harju Johansson: A Structure Utilizing Inexact Primal-Dual Interior-Point Method for Analysis of Linear Differential Inclusions. Thesis No. 1367, 2008.

J. D. Hol: Pose Estimation and Calibration Algorithms for Vision and Inertial Sensors. Thesis No. 1370, 2008.

H. Ohlsson: Regression on Manifolds with Implications for System Identification. Thesis No. 1382, 2008.

D. Ankelhed: On low order controller synthesis using rational constraints. Thesis No. 1398, 2009.

P. Skoglar: Planning Methods for Aerial Exploration and Ground Target Tracking. Thesis No. 1420, 2009.

C. Lundquist: Automotive Sensor Fusion for Situation Awareness. Thesis No. 1422, 2009.

C. Lyzell: Initialization Methods for System Identification. Thesis No. 1426, 2009.

R. Falkeborn: Structure exploitation in semidefinite programming for control. Thesis No. 1430, 2010.

D. Petersson: Nonlinear Optimization Approaches to $\mathcal{H}_2$-Norm Based LPV Modelling and Control. Thesis No. 1453, 2010.

Z. Sjanic: Navigation and SAR Auto-focusing in a Sensor Fusion Framework. Thesis No. 1464, 2011.

K. Granström: Loop detection and extended target tracking using laser data. Thesis No. 1465, 2011.

J. Callmer: Topics in Localization and Mapping. Thesis No. 1489, 2011.

F. Lindsten: Rao-Blackwellised particle methods for inference and identification. Thesis No. 1480, 2011.

M. Skoglund: Visual Inertial Navigation and Calibration. Thesis No. 1500, 2011.

S. Khoshfetrat Pakazad: Topics in Robustness Analysis. Thesis No. 1512, 2011.

P. Axelsson: On Sensor Fusion Applied to Industrial Manipulators. Thesis No. 1511, 2011.

A. Carvalho Bittencourt: On Modeling and Diagnosis of Friction and Wear in Industrial Robots. Thesis No. 1516, 2012.

P. Rosander: Averaging level control in the presence of frequent inlet flow upsets. Thesis No. 1527, 2012.

N. Wahlström: Localization using Magnetometers and Light Sensors. Thesis No. 1581, 2013.

R. Larsson: System Identification of Flight Mechanical Characteristics. Thesis No. 1599, 2013.

Y. Jung: Estimation of Inverse Models Applied to Power Amplifier Predistortion. Thesis No. 1605, 2013.

M. Syldatk: On Calibration of Ground Sensor Networks. Thesis No. 1611, 2013.

M. Roth: Kalman Filters for Nonlinear Systems and Heavy-Tailed Noise. Thesis No. 1613, 2013.

D. Simon: Model Predictive Control in Flight Control Design — Stability and Reference Tracking. Thesis No. 1642, 2014.

J. Dahlin: Sequential Monte Carlo for inference in nonlinear state space models. Thesis No. 1652, 2014.

M. Kok: Probabilistic modeling for positioning applications using inertial sensors. Thesis No. 1656, 2014.

J. Linder: Graybox Modelling of Ships Using Indirect Input Measurements. Thesis No. 1681, 2014.

G. Mathai: Direction of Arrival Estimation of Wideband Acoustic Wavefields in a Passive Sensing Environment. Thesis No. 1721, 2015.

I. Nielsen: On Structure Exploiting Numerical Algorithms for Model Predictive Control. Thesis No. 1727, 2015.

C. Veibäck: Tracking of Animals Using Airborne Cameras. Thesis No. 1761, 2016.

N. Evestedt: Sampling Based Motion Planning for Heavy Duty Autonomous Vehicles. Thesis No. 1762, 2016.

H. Nyqvist: On Pose Estimation in Room-Scaled Environments. Thesis No. 1765, 2016.

Y. Zhao: Position Estimation in Uncertain Radio Environments and Trajectory Learning. Thesis No. 1772, 2017.

P. Kasebzadeh: Parameter Estimation for Mobile Positioning Applications. Thesis No. 1786, 2017.

K. Radnosrati: On Timing-Based Localization in Cellular Radio Networks. Thesis No. 1808, 2018.

G. Lindmark: Methods and Algorithms for Control Input Placement in Complex Networks. Thesis No. 1814, 2018.

M. Lindfors: Frequency Tracking for Speed Estimation. Thesis No. 1815, 2018.

D. Ho: Some results on closed-loop identification of quadcopters. Thesis No. 1826, 2018.

O. Ljungqvist: On motion planning and control for truck and trailer systems. Thesis No. 1832, 2019.

P. Boström-Rost: On Informative Path Planning for Tracking and Surveillance. Thesis No. 1838, 2019.

K. Bergman: On Motion Planning Using Numerical Optimal Control. Thesis No. 1843, 2019.

M. Klingspor: Low-rank optimization in system identification. Thesis No. 1855, 2019.

A. Bergström: Timing-Based Localization using Multipath Information. Thesis No. 1867, 2019.

F. Ljungberg: Estimation of Nonlinear Greybox Models for Marine Applications. Thesis No. 1880, 2020.

E. Hedberg: Control, Models and Industrial Manipulators. Thesis No. 1894, 2020.

R. Forsling: Decentralized Estimation Using Conservative Information Extraction. Thesis No. 1897, 2020.

D. Arnström: On Complexity Certification of Active-Set QP Methods with Applications to Linear MPC. Thesis No. 1901, 2021.

M. Malmström: Uncertainties in Neural Networks: A System Identification Approach. Thesis No. 1902, 2021.

K. Nielsen: Robust LIDAR-Based Localization in Underground Mines. Thesis No. 1906, 2021.

H. Haghshenas: Time-Optimal Cooperative Path Tracking for Multi-Robot Systems. Thesis No. 1915, 2021.

A. Kullberg: On Joint State Estimation and Model Learning using Gaussian Process Approximations. Thesis No. 1917, 2021.

J. Nordlöf: On Landmark Densities in Minimum-Uncertainty Motion Planning. Thesis No. 1927, 2022.

S. A. Zimmermann: Data-driven Modeling of Robotic Manipulators—Efficiency Aspects. Thesis No. 1963, 2023.

FACULTY OF SCIENCE AND ENGINEERING

Linköping studies in science and technology. Licentiate Thesis No. 1967

Department of Electrical Engineering

Linköping University
SE-581 83 Linköping, Sweden

www.liu.se

LINKÖPING
UNIVERSITY