IMT School for Advanced Studies Lucca

Lucca, Italy

Mixed-Integer Quadratic Programming Algorithms for Embedded Control and Estimation

PhD Program in Control Systems

XXX Cycle

By

Vihangkumar Vinaykumar Naik

2018

The dissertation of Vihangkumar Vinaykumar Naik is approved.

Program Coordinator: Prof. Alberto Bemporad,
IMT School for Advanced Studies Lucca, Italy

Supervisor: Prof. Alberto Bemporad,
IMT School for Advanced Studies Lucca, Italy

The dissertation of Vihangkumar Vinaykumar Naik has been reviewed by:

Assoc. Prof. Michal Kvasnica,
Slovak University of Technology in Bratislava, Slovakia

Assoc. Prof. Daniel Axehill,
Linköping University, Sweden

IMT School for Advanced Studies Lucca

2018

To my family,
Madhuben, Dolatrai, Urvashi, Vinay, Kajal, Jay and Dhyey

Contents

List of Figures xi

List of Tables xiv

Acknowledgements xvi

Vita xviii

Publications xix

Abstract xxi

1 Introduction 1

1.1 Mixed-Integer Quadratic Programming 2

1.2 Mixed-Integer Programming for Control of Hybrid Dynamical Systems 4

1.2.1 Mixed Logical Dynamical Systems 5

1.2.2 Hybrid Model Predictive Control 5

1.3 Mixed-Integer Programming for Estimation 7

1.4 MIQP problem solving techniques 9

1.4.1 Optimal solution methods for MIQP 11

1.4.2 Suboptimal solution methods for MIQP 12

1.5 Branch and Bound Method 12

1.5.1 Branching 13

1.5.2 Tree exploration/Node selection 14

1.5.3 Pruning Rules 16

1.6 Notation . . . 18
1.7 Contributions and thesis outline . . . 18

2 Embedded Mixed-Integer Quadratic Optimization using Accelerated Dual Gradient Projection 23
2.1 Introduction . . . 23
2.1.1 Motivation . . . 24
2.1.2 Contributions . . . 24
2.1.3 Outline . . . 25
2.2 Accelerated dual gradient projection . . . 26
2.2.1 Stopping criteria . . . 28
2.2.2 Preconditioning . . . 28
2.2.3 Restart . . . 28
2.2.4 Infeasibility detection . . . 29
2.2.5 Early stopping criterion for the objective function . 30
2.3 Branch and Bound MIQP Algorithm . . . 31
2.3.1 Exploiting the fixed structure of dual QP relaxations 32
2.3.2 Warm-starting the QP subproblems . . . 33
2.4 Heuristic methods for suboptimal binary-feasible MIQP solutions . . . 34
2.4.1 Heuristic approach without using B&B . . . 35
2.4.2 Mid-way Heuristic approach . . . 36
2.5 ADMM based MIQP solver . . . 37
2.5.1 OSQP-based MIQP solver . . . 40
2.6 Numerical results . . . 42
2.6.1 Heuristic approach-Hybrid vehicle example . . . 42
2.6.2 Branch and Bound-random MIQPs . . . 44
2.6.3 Branch and Bound-PieceWise Affine (PWA) regression . . . 47
2.6.4 Mid-way heuristic approach-random MIQPs . . . 48
2.6.5 ARX model segmentation - Heuristic, Mid-way heuristic approach . . . 48
2.7 Conclusion . . . 56

3 A Numerically Robust Mixed-Integer Quadratic Programming Solver based on Nonnegative Least Squares **58**
3.1 Introduction . . . 58
3.1.1 Motivation . . . 59
3.1.2 Contributions . . . 60
3.1.3 Outline . . . 60
3.2 Problem formulation . . . 61
3.3 Solution of QP relaxations . . . 62
3.3.1 Outer proximal-point iterations . . . 63
3.3.2 Inner active-set solver . . . 63
3.3.3 Warm-starting . . . 66
3.3.4 Parameter selection and scaling . . . 67
3.3.5 Stopping criteria and optimality . . . 68
3.4 MIQP solver . . . 68
3.5 Warm-starting binary variables . . . 70
3.5.1 Using mid-way approach to warm-start binary variables . . . 75
3.6 Numerical results . . . 77
3.6.1 Hybrid MPC problem . . . 77
3.7 Conclusions . . . 77

4 Regularized moving-horizon PWA regression using mixed-integer quadratic programming **79**
4.1 Introduction . . . 79
4.1.1 Motivation . . . 80
4.1.2 Contribution . . . 81
4.1.3 Outline . . . 82
4.2 Problem Formulation . . . 82
4.3 PWA Regression Algorithm . . . 83
4.3.1 Recursive clustering and parameter estimation . . . 84
4.3.2 Construction of the state partition . . . 91
4.3.3 Regularized Moving-Horizon PWA Regression for LPV System Identification . . . 93
4.4 Simulation Examples . . . 94
4.4.1 Identification of SISO PWARX system . . . 94

4.4.2 Identification of MIMO PWARX system 96
4.4.3 Identification of SISO LPV system 97
4.5 Conclusion 100

5 Energy Disaggregation using Embedded Binary Quadratic Programming 101
5.1 Introduction 101
5.1.1 Motivation 102
5.1.2 Contribution 104
5.1.3 Outline 105
5.2 Problem setting 105
5.2.1 Single appliance modelling 105
5.2.2 Energy disaggregation problem 106
5.3 Disaggregation Algorithms 107
5.3.1 Approach **A1**: Binary Quadratic Programming . . . 107
5.3.2 Approach **A2**: Regularized Binary Quadratic Programming 108
5.3.3 Approach **A3**: Binary Quadratic Programming with S-AR models 109
5.4 Solving BQP problems using GPAD-based Branch and Bound 110
5.4.1 Solving relaxed QP problems 112
5.4.2 Branch and Bound for BQP problems 115
5.5 Experimental case study 118
5.5.1 Performance Evaluation measures 119
5.5.2 Numerical Results 120
5.6 Conclusion 124

6 Conclusions and outlook 129
6.1 Conclusions 129
6.2 Outlook of possible future directions 131

A Proof of Theorem 1 133

References 136

List of Figures

1 Typical flow of solving QP relaxation of MIQP problem at the root node. 14

2 Branching operation at the root node. 15

3 Upon solving the relaxed QP subproblem (other than root-node), checking for infeasibility and integer feasibility respectively. 17

4 Additional bounding criteria when (finite) upper bound on cost is available. 17

5 Conditions for the binary variables $\bar{A}_i z_R^*$ w.r.t $\epsilon_{\bar{\ell}}$ and $\epsilon_{\bar{u}}$ for $i = 1, \ldots, p$. Only $\epsilon_{\bar{\ell}} < \bar{A}_i z_R^* < \epsilon_{\bar{u}}$ are considered to be solved with B&B, while the others are set to equality as shown. 36

6 Battery energy, battery power, engine power and engine on/off signals versus time: GUROBI (solid blue line), miqpGPAD-H (dash-dotted red line), and miqpADMM-H (dashed black line). 44

7 Number of binary variables and error plots for MIQP problems with 60 vars, 60 ineq. constraints, 8 equality constraints, 50 binary constraints. 49

8 Number of binary variables and error plots for MIQP problems with 90 vars, 80 ineq. constraints, 8 equality constraints, 70 binary constraints. 50

9 Performance comparison for ARX model segmentation problem `iddemo6m.mat` : —●— true value, —— `segment`, - -△- - GUROBI. 54

10 Performance comparison for ARX model segmentation problem `iddemo6m.mat` with $\gamma = 0.7$: —●— true value, —— `segment`, - -⊖- - miqpGPAD-H. 55

11 Performance comparison for ARX model segmentation problem `iddemo6m.mat` with $\gamma = 0.7$, $\epsilon = 0.2$: —●— true value, —— `segment`, - -∗- - miqpGPAD-mH. 56

12 Illustration example with 3 binary variables and $(1, 0, \star)$ as a binary warm-start. The numbers indicate the order in which the QP relaxations are solved, dashed nodes correspond to QP subproblems that are ignored. 73

13 Illustration example with 3 binary variables and $(0, 0, \star)$ as a binary warm-start. The numbers denote the order in which the QP relaxations are solved, dashed nodes correspond to QP subproblems that are ignored. 74

14 Illustration example with 3 binary variables and $z_H^* = (0, 0, 1)$ as solution of heuristic approach, considered as a binary warm-start, the best know integer feasible solution so far V_0 is initialized as $V_0 \leftarrow V_H^*$, $\mathcal{K}_{\bar{\ell}} = \emptyset$, $\mathcal{K}_{\bar{u}} = \{3\}$, $\mathcal{H} = \{1, 2\}$. The numbers denote the order in which the QP relaxations are solved, dashed nodes correspond to QP subproblems that are ignored. 76

15 BFR vs N_p : —— $n_q = 1$, -·-·- $n_q = 2$, - - - $n_q = 3$. 96

16 Validation results for identification of MIMO-PWARX system, Output channel-1. 97

17 Validation results for identification of MIMO-PWARX system, Output channel-2. 98

18 Trends of disaggregated cloth dryer power with Approach A2 for 2-modes (top panel), 3-modes (middle panel), and A3 (bottom panel), where estimated values are calculated using bqpGPAD. 125
19 Trends of disaggregated fridge power with Approach A2 for 2-modes (top panel)), 3-modes (middle panel), and A3 (bottom panel), where estimated values are calculated using bqpGPAD. 126
20 Trends of disaggregated dish washer power with Approach A2 for 2-modes (top panel)), 3-modes (middle panel), and A3 (bottom panel), where estimated values are calculated using bqpGPAD. 127
21 Trends of disaggregated heat pump power with Approach A2 for 2-modes (top panel)), 3-modes (middle panel), and A3 (bottom panel), where estimated values are calculated using bqpGPAD. 128

List of Tables

1 Performance comparison with different values of ϵ_V, ϵ_G for miqpGPAD-H. 43
2 Average CPU time (ms) on random MIQP problems over 50 instances for each combination of n, m, p, q. 45
3 Average CPU time (ms) on random MIQP problems over 50 instances for each combination of n, m, p, q. 45
4 Average CPU time (ms) on random MIQP problems over 10 instances for each combination of n, m, p. 46
5 Average CPU time for training sample N=1000. 48
6 Performance comparison for ARX model segmentation problem using heuristic approach without using B&B. . . . 54
7 Performance comparison for ARX model segmentation problem using the "mid-way" heuristic approach, reported timings are for heuristic and B&B stages respectively. . 55
8 Hybrid MPC problem: CPU time (ms) per sampling step for different prediction horizons N. 78
9 Best Fit Rate on the validation dataset for SISO PWARX system. 95
10 Best Fit Rates on the validation dataset for MIMO PWARX system (4.21). 98
11 Average CPU time for training samples N=1000. 99
12 Power rating levels used in Approaches **A1** and **A2**. 119

13 Actual Energy Fraction Indices h_i and Estimated Energy Fraction Indices $\hat{h}_i$ obtained by Approach **A1** and **A2** using 2-mode models. 121
14 Actual Energy Fraction Indices h_i and Estimated Energy Fraction Indices $\hat{h}_i$ obtained by Approach **A1**, **A2**, and **A3** using 3-mode models, LASSO approach is from Piga et al. (2016). 121
15 RSE_i and R_i^2 coefficients obtained by Approach **A1** and **A2** using 2-mode models. 123
16 RSE_i and R_i^2 coefficients obtained by Approach **A1**, **A2** and **A3** using 3-mode models, LASSO approach is from Piga et al. (2016). 123
17 Average and maximum CPU time (in milliseconds) taken to solve the disaggregation problem for a given window size T. 123

Acknowledgements

I would first like to express my sincere gratitude to my advisor, Prof. Alberto Bemporad, for giving me this opportunity to be a part of his research group. This thesis would not have been possible without his guidance, motivation and encouragement. His passion for research, vast knowledge and research experience has been an inspiration to me and has helped me to grow personally and intellectually. I am indebted to him for his constant support, keen interest in my work and for providing learning opportunities throughout. It is an honour and a privilege to work closely with him. I am also extremely grateful to Dr. Dario Piga for his help and guidance. I cannot thank him enough for his support.

I am thankful to Prof. Panagiotis Patrinos for hosting me at KU Leuven. Working with him has been a great learning experience and a true pleasure for me. I thank Andreas for hosting me at his apartment in Leuven and for many brainstorming discussions. I also thank Puya, Giovanna, Domagoj and Pantelis for a wonderful time in Leuven. I would like to thank Prof. Miroslav Fikar and Assoc. Prof. Michal Kvasnica for hosting me and giving me an opportunity to collaborate with Deepak at the Slovak University of Technology in Bratislava, Slovakia. I am grateful to Shrutika for the warm hospitality in Bratislava. I want to thank Bartolomeo for our interesting collaboration and a wonderful company during his visit to Lucca. I would like to thank the review committee members for taking out their time to review my thesis and for their constructive feedback.

Many thanks to my friends who became family members Manas, Soumali, Armando, Elena, Valentina, Anita, Vitaly,

Lucia, Paolo, Valerio C, Valerio G, Vigneswaran, Xu, Alessandro, Mika, Vincenzo, Daria, Ajay, Ilkay, Davide, Yehia, Laurence, Emi, Marina, Arthur, Abhishek, Pakhee, Nilay, Niraj, Surya, Sampath, Laura, Rosaria, Mirko, Evgenia for such wonderful memories at IMT. I am grateful to Divyesh, Suchakra, Tanushri, Kalyani, Dr. Sonawane and Prof. Phadke for their encouragement and support.

Finally and most importantly, I would like to thank my grandparents Madhuben and Dolatrai for their blessings, my parents Urvashi and Vinay, my sister-in-law Kajal, my brother Jay and my nephew Dhyey for their constant love, care and encouragement, and for standing by me at all times.

Vita

Dec. 06, 1987	Born, Bilimora, India
2005 – 2009	B.E. in Instrumentation & Control Final mark: First Class with Distinction Sarvajanik College of Engineering & Technology Veer Narmad South Gujarat University Surat, Gujarat, India
2010 – 2012	M. Tech in Instrumentation & Control (Process Instru.) Final mark: CGPA 8.94/10 1^{st} Rank College of Engineering, Pune University of Pune, Maharashtra, India
2012 – 2013	Assistant Project Engineer Virtual Lab Project College of Engineering, Pune, Maharashtra, India
2013 – 2015	Engineer Embedded Systems Center of Excellence Eaton Technologies Private Limited Pune, Maharashtra, India
2015 – 2018	PhD in Control Systems IMT School for Advanced Studies Lucca, Italy
08/2017 – 10/2017	Visiting student ESAT – Department of Electrical Engineering KU Leuven, Belgium

Publications

1. V. V. Naik, M. Mejari, D. Piga, and A. Bemporad. Energy disaggregation using embedded binary quadratic programming. Submitted for publication, 2018.

2. V. V. Naik and A. Bemporad. Mixed-integer quadratic optimization based on accelerated dual gradient projection for embedded applications. Technical report, 2018.

3. M. Mejari, V. V. Naik, D. Piga, and A. Bemporad. Energy disaggregation using piecewise affine regression and binary quadratic programming. In *Proc. 57th IEEE Conference on Decision and Control*, Miami Beach, FL, USA, 2018a.

4. A. Bemporad and V. V. Naik. A numerically robust mixed-integer quadratic programming solver for embedded hybrid model predictive control. In *Proc. 6th IFAC Conference on Nonlinear Model Predictive Control*, pages 412–417, Madison, Wisconsin, USA, 2018. DOI: 10.1016/j.ifacol.2018.11.068

5. B. Stellato, V. V. Naik, A. Bemporad, P. Goulart, and S. Boyd. Embedded mixed-integer quadratic optimization using the OSQP solver. In *Proc. European Control Conference*, pages 1536–1541, Limassol, Cyprus, 2018. DOI: 10.23919/ECC.2018.8550136

6. M. Mejari, V. V. Naik, D. Piga, and A. Bemporad. Regularized moving-horizon PWA regression for LPV system identification. In *Proc. 18th IFAC Symposium on System Identification*, pages 1092–1097, Stockholm, Sweden, 2018b. DOI: 10.1016/j.ifacol.2018.09.048

7. V. V. Naik and A. Bemporad. Embedded mixed-integer quadratic optimization using accelerated dual gradient projection. In *Proc. 20th IFAC World Congress*, pages 10723–10728, Toulouse, France, 2017. DOI: 10.1016/j.ifacol.2017.08.2235

8. V. V. Naik, M. Mejari, D. Piga, and A. Bemporad. Regularized moving-horizon piecewise affine regression using mixed-integer quadratic programming. In *Proc. 25th Mediterranean Conference on Control and Automation*, pages 1349–1354, Valletta, Malta, 2017. DOI: 10.1109/MED.2017.7984306

9. V. V. Naik, D. N. Sonawane, D. D. Ingole, and D. Ginoya. Model predictive control of DC servomotor using active set method. In *Proc. IEEE International Conference on Control Applications*, pages 820–825, Aug 2013b. DOI: 10.1109/CCA.2013.6662851

10. V. V. Naik, D. Sonawane, D. D. Ingole, D. L. Ginoya, and V. V. Patki. Design and implementation of proportional integral observer based linear model predictive controller. *International Journal on Control System and Instrumentation*, 4(1):23–30, 2013a. https://tinyurl.com/ijcsi13-pio-mpc

11. D. D. Ingole, D. N. Sonawane, V. V. Naik, D. L. Ginoya, and V. V. Patki. Linear model predictive controller for closed-loop control of intravenous anesthesia with time delay. *International Journal on Control System and Instrumentation*, 4(1):8–15, 2013b. https://tinyurl.com/ijcsi13-bis-td

12. V. V. Naik, D. N. Sonawane, D. D. Ingole, D. L. Ginoya, and N. S. Girme. Design and implementation of interior-point method based linear model predictive controller. In *Mobile Communication and Power Engineering*, pages 255–261, Berlin, Heidelberg, 2013c. Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-35864-7_36 *(Best paper award)*

13. D. D. Ingole, D. N. Sonawane, V. V. Naik, D. L. Ginoya, and V. Patki. Implementation of model predictive control for closed loop control of anesthesia. In *Mobile Communication and Power Engineering*, pages 242–248, Berlin, Heidelberg, 2013a. Springer Berlin Heidelberg DOI: 10.1007/978-3-642-35864-7_34

Presentations

14. V. V. Naik and A. Bemporad. Embedded mixed-integer quadratic optimization using first-order methods. In *Proc. 4th European Conference on Computational Optimization*, Leuven, Belgium, 2016.

Abstract

The class of optimization problems involving both continuous and discrete variables is known as mixed-integer programming (MIP), which emerges in many fields of applications. Due to their inherent combinatorial nature, solving such a class of problems in real-time poses a major challenge, especially in embedded applications where computational and memory resources are limited. This thesis mainly focuses on novel solution methods tailored to small-scale Mixed-Integer Quadratic Programming (MIQP) problems, such as those that typically arise in embedded hybrid Model Predictive Control (MPC) and estimation problems. With an emphasis on algorithm simplicity, efficient solution techniques to solve MIQP problems are developed in the thesis based on first-order methods, specialized to find both exact and approximate solutions. In addition, a numerically robust algorithm is proposed in order to tackle MIQP problem with positive semidefinite Hessian matrices, often encountered in hybrid MPC formulations. The proposed techniques, being library-free and relatively simple to code, are specifically tailored to real-time embedded applications. Such techniques are also employed in a novel algorithm for the MIP-based PieceWise Affine (PWA) regression, as well as in new approaches for energy disaggregation using binary quadratic programming that are particularly suitable for smart energy meters.

Chapter 1

Introduction

Making the best/optimal choice from a set of possible decisions using a mathematical formulation is referred to as solving an *optimization problem*. Particularly, the optimal decision is made by taking into consideration the *constraints* which represent specified limits on the possible decisions that can be taken, deriving from the process or system that is optimized. In particular, for an efficient operation of any given system, the optimal decision making mechanism is at the core of it. An optimization problem containing integer decision variables is called *integer programming* problem. When some decision variables can take integer values and the others can take real values, the problem is referred to as *Mixed-Integer Programming* (MIP) problem.

A wide range of practical applications utilize optimization-based decision making mechanisms by employing iterative algorithms in order to compute the solution of the optimization problem in real-time. Solving optimization problems periodically at every time instance poses the challenge of requiring efficient solution methods as well as a powerful computing platform which can handle the computational burden imposed upon. Consequently, one often employs optimization algorithms on desktop computers running software packages, or uses optimization only in tasks which allow sufficiently long computation time. However, over the past few years, with considerable advancements in the field

of optimization as well as embedded computing systems, a significant paradigm of *embedded optimization* has attracted a lot of attention both from the academic community and industries. This progress widens the spectrum of the optimization methods beyond "desktop" computers, and makes it possible to apply optimization algorithms to applications such as automotive, aerospace, robotics, and power electronics. Optimization solvers used for embedded applications are generally imposed with restrictions on factors such as memory, computational resources and/or execution time. Due to such restrictions the solver is desired to be library free and simple to implement, and numerically robust for implementation using reduced precision arithmetic.

A large number of solution approaches for solving *Quadratic Programming* (QP) problems have been recently proposed for embedded applications. QP is a specific type of optimization problem with quadratic objective function and linear constraints having continuous decision variables. However, for problems arising in control and identification of hybrid dynamical systems simple QP solutions do not suffice, and MIP is required instead. The MIP problems belong to the class of NP-hard problems, and hence it poses even greater challenges for solving the problem in an embedded application.

In this thesis we deal with Mixed-Integer Quadratic Programming (MIQP) problems, which is a special case of mixed-integer nonlinear programming. Specifically, we mainly focus on novel solution methods tailored to small-scale MIQP problems, such as those that arise in embedded hybrid Model Predictive Control (MPC) and hybrid estimation problems.

1.1 Mixed-Integer Quadratic Programming

Mixed-Integer Quadratic Programming (MIQP) is a class of optimization problem involving both continuous and discrete decision variables, having a quadratic objective function subject to linear constraints (Balas, 1969; Lazimy, 1985; Fletcher and Leyffer, 1998; Axehill and Hansson,

2006). We consider the MIQP problem of the general form

$$\min_{z} \quad V(z) \triangleq \frac{1}{2} z'Qz + c'z \tag{1.1a}$$
$$\text{s.t.} \quad \ell \leq Az \leq u \tag{1.1b}$$
$$Gz = g \tag{1.1c}$$
$$\bar{A}_i z \in \{\bar{\ell}_i, \bar{u}_i\},\ i = 1, \ldots, p \tag{1.1d}$$

where $z \in \mathbb{R}^n$ is the vector of decision variables, $Q \in \mathbb{R}^{n \times n}$ is the symmetric positive definite Hessian matrix, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $\ell, u \in \mathbb{R}^m$, $\ell \leq u$, represents linear inequality constraints, $G \in \mathbb{R}^{q \times n}$, $g \in \mathbb{R}^q$ describes linear equality constraints and the integrality constraints are described by $\bar{A} \in \mathbb{R}^{p \times n}$, $\bar{\ell}, \bar{u} \in \mathbb{R}^p$, $\bar{\ell} \leq \bar{u}$, and $p \leq m$. The usual binary constraints $z_i \in \{0, 1\}$ are a special case of (1.1d) having $\bar{\ell}_i = 0, \bar{u}_i = 1$, and $\bar{A}_i$ the i-th row of the identity matrix (such problem is also referred to as a mixed-binary quadratic programming (MBQP) problem).

When the integrality constraint (1.1d) is relaxed as

$$\bar{\ell}_i \leq \bar{A}_i z \leq \bar{u}_i,\ i = 1, \ldots, p \tag{1.2}$$

the following QP problem

$$\min_{z} \quad V(z) \triangleq \frac{1}{2} z'Qz + c'z \tag{1.3a}$$
$$\text{s.t.} \quad \ell \leq Az \leq u \tag{1.3b}$$
$$Gz = g \tag{1.3c}$$
$$\bar{\ell} \leq \bar{A}z \leq \bar{u} \tag{1.3d}$$

is termed as "QP relaxation" of the MIQP problem (1.1). This general form of MIQP problem (1.1) with strictly convex QP relaxations is built upon extensively throughout this thesis.

MIQP problems arise in diverse application areas including hybrid model predictive control (Bemporad and Morari, 1999a), moving-horizon estimation (Bemporad et al., 1999a; Ferrari-Trecate et al., 2002), piecewise affine regression (Bemporad et al., 2001; Roll et al., 2004), trajectory generation (Mellinger et al., 2012), economic dispatch (Papageorgiou and Fraga, 2007), planning and design (Propato and Uber, 2004;

Yang et al., 2007), scheduling (Catalão et al., 2010), network topology identification (Tian et al., 2016). In MIQP problems the worst-case computational complexity grows exponentially with the number of binary (or integer) decision variables. This inherent characteristic restricts solving large-scale problems using "desktop" computers, for which efficient commercial software packages (IBM, Inc., 2014; Gurobi Optimization, Inc., 2014; Fair Isaac Corporation, 2015; MOSEK ApS, 2015) are available.

However, the possibility of solving *small-scale* MIQP problems on an embedded platform has attracted considerable attention from the scientific community in recent years, as motivated in the next section.

1.2 Mixed-Integer Programming for Control of Hybrid Dynamical Systems

A wide class of practical systems has interactions between continuous dynamics and discrete components, which are known as *hybrid systems*. One of the initial papers on hybrid systems dates back to the late 1960s by Witsenhausen (1966), thereafter several modelling frameworks have been proposed (Antsaklis, 2000; Cassandras et al., 2001; Goebel et al., 2009). As compared to the continuous-time hybrid systems (Lincoln and Rantzer, 2001; Xu and Antsaklis, 2003; Borrelli et al., 2005), the optimal control of discrete-time hybrid systems are easy to formulate and to solve numerically. We focus on discrete-time hybrid systems hereafter.

Since its introduction almost two decades ago by Bemporad and Morari (1999a), hybrid Model Predictive Control (MPC) has attracted a lot of attention both from academia and industry. The reason for such popularity is mainly due to the fact that hybrid systems can model a very large spectrum of real-world systems where physical processes with switching dynamics, discrete actuators, logic rules, and constraints on system variables coexist, and that MPC provides an optimal way of controlling them.

1.2.1 Mixed Logical Dynamical Systems

Hybrid systems can efficiently be modeled using the following Mixed Logical Dynamical (MLD) framework (Bemporad and Morari, 1999a),

$$x_{k+1} = \mathcal{A}x_k + \mathcal{B}_1 v_k + \mathcal{B}_2 \delta_k + \mathcal{B}_3 \zeta_k + \mathcal{B}_5 \tag{1.4a}$$

$$y_k = \mathcal{C}x_k + \mathcal{D}_1 v_k + \mathcal{D}_2 \delta_k + \mathcal{D}_3 \zeta_k + \mathcal{D}_5 \tag{1.4b}$$

$$\mathcal{E}_2 \delta_k + \mathcal{E}_3 \zeta_k \leq \mathcal{E}_1 v_k + \mathcal{E}_4 x_k + \mathcal{E}_5, \tag{1.4c}$$

where $k \in \mathbb{N}$ is the time index, x_k is the state vector, y_k is the output vector, v_k is the input vector, ζ_k and δ_k are the vectors of auxiliary variables which are real-valued and binary respectively. The vectors x_k, v_k, y_k can have both real and binary components, $\mathcal{A}$, $\mathcal{B}_i$, $\mathcal{C}$, $\mathcal{D}_i$ and $\mathcal{E}_i$ are the constant matrices describing the behavior of the hybrid dynamical system. The tool HYSDEL (hybrid systems description language) (Torrisi and Bemporad, 2004) allows one to describe a hybrid dynamical model and get the equivalent MLD transformation (1.4).

Under certain assumptions, MLD systems are equivalent to other classes of hybrid systems, such as *Piecewise Affine* (PWA) and linear complementarity systems (Heemels et al., 2001; Bemporad et al., 2000; Herceg et al., 2013), and the MLD representation allows to systematically formulate control/estimation objective into an optimization problem.

1.2.2 Hybrid Model Predictive Control

Model Predictive Control (MPC) is an advanced control technique which is very popular in the process industry. MPC is a feedback, optimal control strategy based on numerical optimization. At each sampling instance, MPC solves an optimization problem online, and computes the sequence of optimal current and future control inputs by minimizing the difference between set-points and future outputs predicted by a given plant model over a finite time horizon in forward time. Then, only the current optimal input is applied to the plant. The updated plant information is used to formulate and solve a new optimal control problem at the next sampling instance. This procedure is repeated at every sampling

instance, hence called as *receding horizon control* (RHC). Model predictive control has received attention largely due to its ability to handle hard constraints (Wright, 1997; Rao et al., 1998; Bemporad and Morari, 1999b; Mayne et al., 2000). However, the need to solve an optimization problem at every sampling instance, typically via iterative optimization algorithms, imposes a large computational load.

Receding horizon control of mixed logical dynamical (MLD) systems, also referred to as hybrid model predictive control (hybrid MPC) (Bemporad and Morari, 1999a) has enticed notable attention from researchers in various fields. Based on the MLD model (1.4), a possible hybrid MPC problem formulation is the following

$$\min_{\{v_k,\delta_k,\zeta_k\}_{k=0}^{N-1}} \sum_{k=0}^{N-1} \quad \|L_x(x_k - r_k^x)\|_2^2 + \|L_v(v_k - r_k^v)\|_2^2 + \tag{1.5a}$$

$$\|L_\delta(\delta_k - r_k^\delta)\|_2^2 + \|L_\zeta(\zeta_k - r_k^\zeta)\|_2^2$$

$$\text{s.t.} \quad \text{MLD model (1.4)} \tag{1.5b}$$

$$x_0 = x(t).$$

where N is the length of horizon, and objective (1.5a) minimizes the (weighted squared) norm of the difference between set-points and decision variables. Problem (1.5) can be recast as an MIQP problem of the form (1.1) by defining the n-dimensional optimization vector

$$z = \left[v_0' \cdots v_{N-1}' \; \delta_0' \cdots \delta_{N-1}' \; \zeta_0' \cdots \zeta_{N-1}' \right]'.$$

On-line implementation of hybrid model predictive control (MPC) requires the solution of a Mixed-Integer Quadratic Programming (MIQP) problem at every sampling instance (Bemporad and Morari, 1999a). The only exceptional cases are hybrid MPC problems for small systems having few binary variables that can be solved off line using multi-parametric programming (Bemporad et al., 2002; Borrelli et al., 2005; Alessio and Bemporad, 2009; Bemporad, 2015a). Mature software tools for multi-parametric programming are available (Bemporad, 2003; Kvasnica et al., 2004; Löfberg, 2004; Herceg et al., 2013) to facilitate a development environment for modeling, analysis, and control.

Implementation of hybrid MPC on personal computers are in practice thanks to the mature software tools such as HYSDEL (Torrisi and Bemporad, 2004) for designing MLD models; Hybrid Toolbox (Bemporad, 2003) for modelling, analysis and controller design for hybrid systems; MPT toolbox (Kvasnica et al., 2004; Herceg et al., 2013) for modeling, control, analysis, and deployment of constrained optimal controllers; YALMIP (Löfberg, 2004) for prototyping and solver interface; mixed-integer optimization commercial solvers CPLEX (IBM, Inc., 2014), GUROBI (Gurobi Optimization, Inc., 2014), FICO Xpress (Fair Isaac Corporation, 2015) and, MOSEK (MOSEK ApS, 2015). However, such existing MIQP solvers are not tailored to be implemented on resource constrained embedded platforms. This limits the applicability of hybrid MPC and possesses major challenge in the area of real-time hybrid MPC implementation for fast dynamic systems.

Motivated by this fact, in this thesis solution approaches for solving MIQP problems in an embedded control setting have been investigated. While "desktop" applications utilize a numerical package to solve (sometimes large-scale) MIQPs with large memory and computing resources available and no stringent limits on execution time, in embedded applications, severe restrictions on CPU/memory/time resources are imposed, the number of variables (especially binary variables) should be small, the code must be simple and library-free, the algorithm must be numerically robust, and shall render a solution within specified tolerance limits also when executed in reduced precision arithmetic. The MIQP solution methods described in this thesis are driven by such requirements.

1.3 Mixed-Integer Programming for Estimation

The availability of a system model is essential for control and analysis of hybrid dynamical systems. In some cases, deriving a mathematical model of the system from first-principle laws can be a costly and time consuming task. In these cases, model parameters need to be estimated using experimental data. Such a data-driven modelling approach is generally referred to as *system identification*.

Among various approaches available in the literature for the identification of hybrid and switching systems (see the survey papers (Garulli et al., 2012; Paoletti et al., 2007)), we focus particularly on *PieceWise Affine* (PWA) regression algorithms using mixed-integer programming. PWA models are simple and flexible model structures having universal approximation properties such that any nonlinear function can be modelled with arbitrary accuracy by a PWA map (Breiman, 1993). Due to the equivalence between PWA models and several classes of hybrid models (Heemels et al., 2001; Bemporad, 2004), available tools for control and analysis of hybrid systems are also applicable for PWA systems (Bemporad and Morari, 1999a; Bemporad et al., 2000).

Mixed-integer programming was proposed to solve PWA regression problems in the identification of hybrid systems in (Roll et al., 2004; Bemporad et al., 2001). In these approaches, the estimation of hinging-hyperplane ARX models and piecewise affine Wiener models is formulated as a mixed-integer linear or quadratic programming problem, and then solved through a branch and bound algorithm. The computation complexity of such problems grows proportional with the number of linear sub-models (or "modes") defining the PWA models and with the size of the training dataset, as the number of integer variables directly depends on both these parameters. Hence, these approaches are limited to problems with a moderate number of integer variables, for example cases in which comparatively fewer number of data points are available (e.g., when it is very costly to obtain data). Although computationally expensive, these approaches give a global optimal solution. In this thesis, we propose novel MIQP based algorithm for the PWA regression problems.

Furthermore, in this thesis we employ MIQP and hybrid system identification for an estimation problem from the energy sector. Specifically, we consider an estimation problem of non-intrusive load monitoring (NILM) also known as Energy Disaggregation or non-intrusive appliance load monitoring (NIALM). The NILM approach calculates the estimates of energy consumption of the individual appliance based only on the aggregate/total energy consumption measured at a single point in a whole building/premise (Hart, 1992).

We focus particularly on the NILM techniques based on integer programming. An integer programming based approach is presented in (Suzuki et al., 2008) for small-scale problems, and for the cases where some appliance has multiple modes. One of the challenges for such approaches is to distinguish between appliances with similar or overlapping load signature. The aided linear integer programming (ALIP) based approach presented in (Bhotto et al., 2017) is with correction based on a state diagram, median filtering, and linear-programming-based refinement. The mixed-integer linear programming (MILP) based NILM approach presented in (Wittmann et al., 2018) introduces a new set of integer linear constraints.

As discussed in Section 1.2.2, similar to control approaches, the estimation methods based on integer and mixed-integer programming available in literature rely on the desktop computer packages, and the embedded implementation of such approaches is still a very recent research area.

1.4 MIQP problem solving techniques

A large variety of approaches have been proposed over the last few decades for solving mixed-integer programming problems including Generalized Benders decomposition method (Geoffrion, 1972; Lazimy, 1985), cutting plane methods (Gomory, 1958, 1963), branch and bound (Land and Doig, 1960; Gupta and Ravindran, 1985), branch and cut (Bienstock, 1996; Stubbs and Mehrotra, 1999) and outer approximation algorithms (Duran and Grossmann, 1986; Fletcher and Leyffer, 1994). Review of methods for solving mixed-integer programming problems can be found in (Schrijver, 1986; Sherali and Driscoll, 2000; Grossmann, 2002). Among these methods, the branch and bound (B&B) algorithm is the most widely adapted method (Fletcher and Leyffer, 1998; Linderoth and Savelsbergh, 1999; Bemporad and Morari, 1999a; Axehill and Hansson, 2006; Morrison et al., 2016) for finding the global solution.

In 1960, Land and Doig (1960) pioneered the branch and bound algorithm for solving mixed-integer linear programs, which has been

widely adapted for general integer programming problems. Dakin (1965) proposed a branching dichotomy and the elaborations to B&B were proposed by Beale and Small (1965); Little et al. (1963). A thorough treatment of B&B technique can be found in survey papers (Lawler and Wood, 1966; Linderoth and Savelsbergh, 1999; Morrison et al., 2016). Over almost last five decades with significant improvements in computing platforms, mathematical algorithms and efficient software implementation, led to the evolution of integer programming which has been summarized in (Bixby, 2010; Jünger et al., 2010).

A combinatorial problem can be in principle solved by enumerating all possible combinations of the integer variables, and finding the combination which makes the objective function the smallest. This procedure is known as "exhaustive enumeration" or "exhaustive search". However, this technique becomes impractical as the number of integer variables grow, due to the exponential growth of the potential problems to be solved. The branch and bound algorithm provides a structured methodology to possibly avoid such an exhaustive search. This algorithm in a basic form relies on sequentially partitioning the integer feasible solution space into small subsets or subregions, which is proceeded by constructing a tree structure to search the space of all feasible solutions. An optimization problem is required to be solved corresponding to each subregion. By using the bounds on the optimal value available so far, the algorithm avoids exhaustive search by possibly eliminating certain sections of the tree. Overall, the relative efficiency of the B&B algorithm primarily relies on possibly eliminating large subsets of the solution space, and on an efficient solution of the corresponding optimization problems. Employing a branch and bound algorithm for mixed-integer quadratic programming (MIQP) problems relies on efficient solution of the quadratic programming (QP) problems obtained from relaxation of integer constraints.

A number of embedded optimization solution techniques for solving QP problems have been presented including the active-set method qpOASES (Ferreau et al., 2014), nonnegative least squares (NNLS) (Bemporad, 2016, 2018); the interior-point method CVXGEN (Mattingley

and Boyd, 2012), FORCES (Domahidi et al., 2012); the first order method FiOrdOs (Ullmann, 2011), accelerated dual gradient projection (GPAD) (Patrinos and Bemporad, 2014), operator splitting quadratic program (OSQP) (Stellato et al., 2017); commercial solvers FORCES PRO (Domahidi and Jerez, 2014), ODYS QP (Cimini et al., 2017). For a detailed summary see the survey paper on embedded optimization methods (Ferreau et al., 2017). However, the challenge still remains open for MIQP solvers tailored to embedded applications, which are described in the following section.

1.4.1 Optimal solution methods for MIQP

For finding the optimal solution of MIQP problems, various approaches have been presented using the well-known branch and bound (B&B) algorithm (Floudas, 1995), which rely on the solution of QP relaxations of the form (1.3). A B&B based approach to solve MIQPs tailored for MPC is proposed by Axehill and Hansson (2006). The method uses a dual QP formulation and employs warm-starting. However, this implementation does not exploit dual lower bounds on the optimal cost, that is very useful to terminate the relaxed QP solver prematurely (Fletcher and Leyffer, 1998; Axehill and Hansson, 2008).

Recently Bemporad (2015b) and Frick et al. (2015) have developed new algorithms for solving MIQPs which are tailored for embedded systems. In (Frick et al., 2015) an embedded interior-point based convex programming solver is extended and is combined with a B&B setting. The authors investigated relaxations of MPC problems with long horizons, and provided the trade-off computation time against closed-loop performance. They solved hybrid MPC problems with a significant number of binary variables in acceptable time frames for implementation on embedded systems. Bemporad (2015b) proposed an approach combining B&B with an active-set method based on nonnegative least squares (NNLS) to solve QP relaxations (Bemporad, 2016). This approach was particularly tailored to solving small-scale MIQPs such as those typically arising in embedded hybrid MPC applications.

1.4.2 Suboptimal solution methods for MIQP

Due to its combinatorial nature, solving an MIQP to optimality may be impractical in fast applications. For this reason, efforts have been made recently for solving MIQPs *approximately* in a very quick manner. Heuristic methods were proposed which lead to a suboptimal solution of the MIQP while taking considerably less computation time than finding the optimal solution. Fischetti et al. (2005) proposed a heuristic method for finding a feasible solution of a generic MIP, known as the feasibility pump (FP). Bertacco et al. (2007) extended this approach for binary and general-integer variables and the subsequent extension is shown in (Achterberg and Berthold, 2007). Recently, Takapoui et al. (2016) presented a heuristic with alternating direction method of multipliers (ADMM) to find an approximate solution of MIQP problem, within very short time. An approach based on operator splitting method for finding suboptimal solution, with guaranteed local convergence results under certain assumptions was presented in (Frick et al., 2016).

1.5 Branch and Bound Method

The B&B algorithm proceeds as a tree search, where each node of the tree represents a unique relaxed QP subproblem. The tree is initialized by relaxing all the integrality constraints (1.1d) into the form (1.2), by allowing the binary expressions $\bar{A}_i z$ to take any continuous value in the interval $\left[\bar{\ell}_i, \bar{u}_i\right]$. Therefore, the relaxed QP problem (1.3) is represented by the *root* node of the tree, which is a unique origin node. The tree evolves by the *branching* operation where two *children* candidate nodes are created from each *parent* or *ancestor* node. Specifically, branching picks one relaxed integer variable (of form $\bar{\ell}_i \leq \bar{A}_i z \leq \bar{u}_i$) from the parent problem and creates two children problems by fixing the selected integer variable to either bounds, $\bar{A}_i z = \bar{\ell}_i$ and $\bar{A}_i z = \bar{u}_i$. Nodes with all the integers fixed either to $\bar{\ell}_i$ or $\bar{u}_i$, or in other words nodes with no children, are called *leaf* nodes or *leaves* of the tree.

A typical flow of solving QP relaxations is shown in Figure 1 (Bem-

porad, 2015b). If at the root node, the QP relaxation (1.3) is found infeasible, then the given MIQP problem (1.1) is declared infeasible. If the QP relaxation is feasible, this is followed by checking for integer feasibility. Specifically, this checks if the integrality constraint (1.1d) is satisfied, i.e., if all $i = \{1, \ldots, p\}$ integer variables $\bar{A}_i z$ are either equal to $\bar{\ell}_i$ or $\bar{u}_i$ then the corresponding solution is considered to be an *integer feasible solution*. If the QP relaxation at the root node is integer feasible then the MIQP problem (1.1) is solved. If any of the integer variables $\bar{A}_i z$ has *fractional value*, then the tree further evolves by the branching operation.

1.5.1 Branching

The branching operation picks a variable $\bar{A}_i z$ (with an index i) having fractional value between $\bar{\ell}_i$ and $\bar{u}_i$, and transforms the corresponding relaxed inequality $\bar{\ell}_i \leq \bar{A}_i z \leq \bar{u}_i$ into two equalities $\bar{A}_i z = \bar{\ell}_i$ and $\bar{A}_i z = \bar{u}_i$, thus creating two children nodes (i.e., two new subproblems). This operation is graphically represented in Figure 2. This is continued until all the integer variables $\bar{A}_i z$, with $i = \{1, \ldots, p\}$, are branched upon in same fashion.

Thus during the branching process, an integer variable needs to be selected for branching. This is done by the branching rule which selects the next variable to be branched upon. Some of the commonly used branching rules are,

- **Maximum fractional part:** picks up index i such that $\bar{A}_i z$ is closest to $\frac{\bar{\ell}_i + \bar{u}_i}{2}$
- **First free variable:** selects the variable with smallest index i from the set of relaxed integer variables

This operation results in two new children QP subproblems QP_1 and QP_2 from the root node QP_0, as shown in Figure 2. The indices corresponding to these two children subproblems are stored on a *list* or a *stack*, which holds the indices of the remaining/pending nodes to be explored.

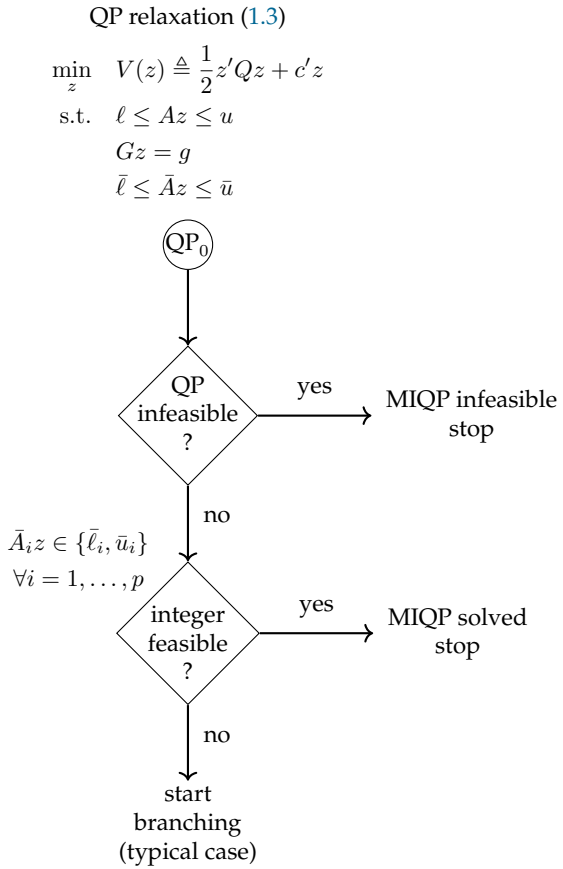


Figure 1: Typical flow of solving QP relaxation of MIQP problem at the root node.

1.5.2 Tree exploration/Node selection

Once the branching variable is selected and two subproblems are stored on the stack, the order of solving the relaxed QP subproblems is determined using the tree exploration rule or node selection rule. The tree exploration strategies used in practice are,

- **Breadth first search:** this is a first-in, first-out (FIFO) strategy, ex-

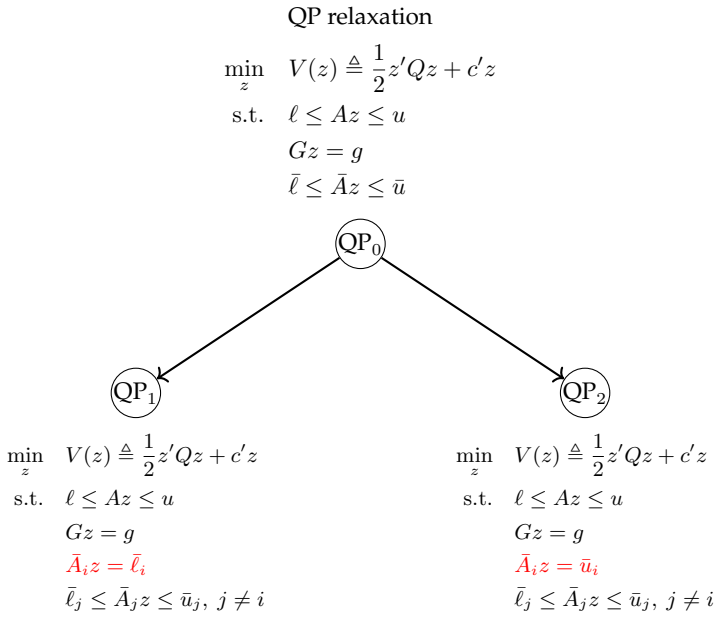


Figure 2: Branching operation at the root node.

pands a binary tree horizontally by prioritizing the nodes at the same level (starting from the root, the row of nodes till leaves is called a levels and they are characterized by the cardinality of the set of the integers).

- **Depth first search:** this is a last-in, first-out (LIFO) strategy, expands a binary tree vertically by selecting one of the children nodes.

- **Best first search/ best bound search:** this selects the node with lowest bound.

Thus, one of two nodes are selected using one of these strategies and the corresponding relaxed subproblem is solved. Each of these rule results in a different enumeration of the binary tree, and consequently af-

fects the number of nodes explored, computation time of the overall B&B algorithm. The overall idea of the B&B scheme is to continue the tree search until each leaf is explored.

1.5.3 Pruning Rules

A set of checks are used to possibly avoid the complete enumeration, which are the key element of efficient B&B implementation. Bounding rules are employed which enables to *fathom* or *prune* a node i.e., a section of a tree that can be eliminated from the exploration process. Pruning rules are employed such as pruning by

- **infeasibility:** if a subproblem is infeasible;
- **integer feasibility:** if a subproblem returns an integer feasible solution;
- **optimality/bound:** if the optimal cost of a subproblem is worse than the best known integer feasible solution found so far.

Figure 3 describes the flow of these checks for nodes other than the root node, where V_0 denotes the best known integer feasible solution found so far (also known as an *incumbent*).

The best known integer feasible solution V_0 is initialized with infinity and updated as soon the solution of a subproblem is integer feasible, as shown in Figure 3.

Note that the latter pruning rule avoids further branching, as children subproblems would provide higher costs only, see Figure 4. Hence, a good initial integer feasible solution helps to possibly prune subsections of a tree, which is essential for an efficient execution of the B&B algorithm.

The tree search/exploration process continues until no pending nodes (problems) remain to be explored.

We build upon this general idea of B&B in Chapter 2 and Chapter 3.

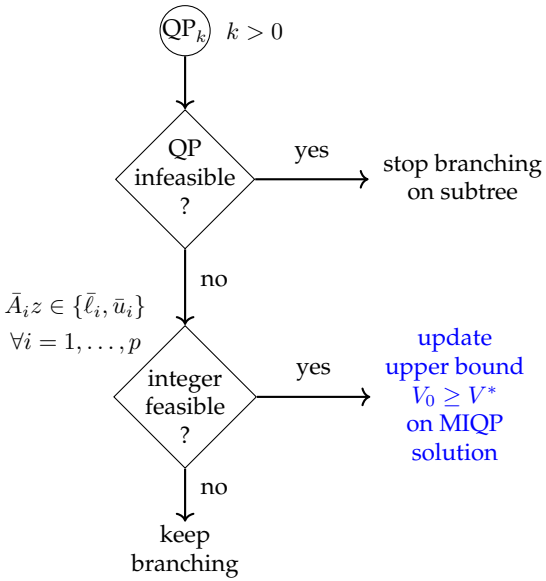


Figure 3: Upon solving the relaxed QP subproblem (other than root-node), checking for infeasibility and integer feasibility respectively.

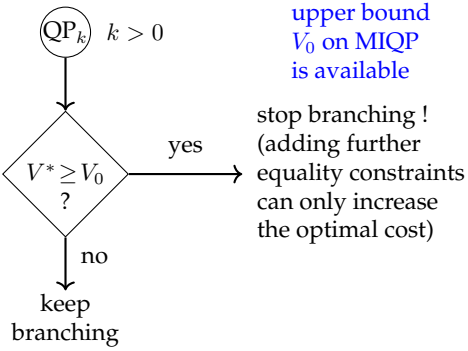


Figure 4: Additional bounding criteria when (finite) upper bound on cost is available.

1.6 Notation

The following notation will be used throughout the thesis. Let $\mathbb{R}^n$ denote the set of real vector of dimension n; $\mathbb{R}^{m\times n}$ is the set of real matrices with m rows and n columns, $\mathbb{N}$ is the set of natural integers, and $\{0,1\}^p$ is p-tuples of binary variables. For a vector $a \in \mathbb{R}^n$, a_i denotes the i-th component of a, the condition $a > 0$ is equivalent to $a_i > 0, \forall i = 1,\dots,n$ (and similarly for $\geq$, $\leq$, $<$). A square diagonal matrix in $\mathbb{R}^{n\times n}$ formed with the components of a vector $a \in \mathbb{R}^n$ on the main diagonal is denoted by $\text{diag}(a)$. For a matrix $A \in \mathbb{R}^{m\times n}$, A_i denotes its i-th row, A' its transpose, $\lambda_{max}(A)$ its maximum eigenvalue, and its Frobenius norm by $\|A\|_F$. For a square matrix $A \in \mathbb{R}^{n\times n}$, its inverse is denoted by A^{-1} (if it exists). Matrix $A^{\#} \in \mathbb{R}^{m\times n}$ denotes the pseudoinverse matrix of A, $A^{\#} \triangleq (A'A)^{-1}A'$ if A is full column rank. $A \succ 0$ denotes that A is positive definite, and similarly $\succeq$, $\prec$, and $\preceq$ denote positive semidefiniteness, negative definiteness, and negative semidefiniteness, respectively. Let $\mathcal{I} \subset \mathbb{N}$ be a finite subset, we denote by $a_{\mathcal{I}}$ the subvector obtained by collecting all the components a_i for all $i \in \mathcal{I}$, and similarly the submatrix $A_{\mathcal{I}}$ is obtained by collecting all the rows A_i. The number of elements (cardinality) of a discrete set $\mathcal{I}$ is denoted by $card(\mathcal{I})$. The Euclidean norm of a is denoted by $\|a\|_2$, the 1-norm of a by $\|a\|_1 = \sum_{i=1}^{n} |a_i|$, and its infinite-norm by $\|a\|_\infty$. Matrix I_n denotes the $n \times n$ identity matrix, $\mathbf{1}_n$ and $\mathbf{0}_n$ represents vector of ones and zeros, respectively, with n elements. Dimension subscript n is dropped whenever it is clear from the context. The symbol $\odot$ denotes the element-wise multiplication between two matrices, and $\otimes$ the Kronecker product.

1.7 Contributions and thesis outline

Chapter 2 : Embedded Mixed-Integer Quadratic Optimization using Accelerated Dual Gradient Projection

The Mixed-Integer Quadratic Programing (MIQP) problem is a combinatorial problem. It poses a major challenge to solve such a class of problems on resource and memory constrained computing environments. To

address this issue, with an emphasis on algorithmic simplicity, in this chapter we propose Accelerated Dual Gradient Projection (GPAD) based approaches tailored to solve small-scale MIQP problems such as those that typically arise in embedded applications.

In particular, an existing GPAD algorithm is devised with additional specializations for the efficient solution of Quadratic Programming (QP) relaxations that arise during Branch and Bound (B&B) while solving the MIQP to optimality. Moreover, in order to find an integer feasible combination of the binary variables, two heuristic approaches are presented: (i) without using B&B, and (ii) using B&B with significantly reduced number of QP relaxations referred to as "mid-way" heuristic approach. In addition, the proposed idea of keeping a fixed QP matrix structure throughout the execution of B&B is extended to MIQP solving approaches using the Alternating Direction Method of Multipliers (ADMM) algorithm and the Operator Splitting Quadratic Program (OSQP) solver. The presented algorithms are very simple to code and require only basic arithmetic operations to be performed online, which makes them well suited for an embedded implementation.

The work presented in this chapter led to the following contributions:

- V. V. Naik and A. Bemporad. Embedded mixed-integer quadratic optimization using accelerated dual gradient projection. In *Proc. 20th IFAC World Congress*, pages 10723–10728, Toulouse, France, 2017.

- V. V. Naik and A. Bemporad. Mixed-integer quadratic optimization based on accelerated dual gradient projection for embedded applications. Technical report, 2018.

- B. Stellato, V. V. Naik, A. Bemporad, P. Goulart, and S. Boyd. Embedded mixed-integer quadratic optimization using the OSQP solver. In *Proc. European Control Conference*, pages 1536–1541, Limassol, Cyprus, 2018.

Presentation

- V. V. Naik and A. Bemporad. Embedded mixed-integer quadratic optimization using first-order methods. In *Proc. 4th European Conference on Computational Optimization*, Leuven, Belgium, 2016.

Chapter 3 : A Numerically Robust Mixed-Integer Quadratic Programming Solver based on Nonnegative Least Squares

The deployment of hybrid Model Predictive Control (MPC) in practical applications requires primarily an efficient and robust on-line Mixed-Integer Quadratic Programming (MIQP) solver that runs in real-time. However, hybrid MPC formulations often result in positive semidefinite Hessian matrices, due to some variables having zero weight in the MPC cost function. The MIQP solution approach using Accelerated Dual Gradient Projection (GPAD) and Alternating Direction Method of Multipliers (ADMM) presented in Chapter 2 requires a strictly convex objective function, and numerical performance of the such solvers deteriorates with the condition number of the Hessian matrices. While regularizing the cost function would improve numerical robustness, it would bias the solution from optimality.

A new approach is proposed in this chapter to tackle such problems which uses a numerically robust QP solver based on an active-set method for solving nonnegative least squares (NNLS) and proximal-point iterations combined with branch and bound (B&B), which handles positive semidefinite Hessian matrices. The B&B algorithm is further equipped with a generic framework to warm-start the binary variables, which provides a way to exploit the knowledge of the system/process/problem under consideration. It is especially useful in the case of hybrid MPC and moving-horizon estimation where a good initial guess for the binary variables is available by shifting the optimal solution computed at the previous sample step. This idea is further utilized for the PieceWise Affine (PWA) regression algorithm proposed in the following chapter. The "mid-way" heuristic approach presented in Chapter 2 is also demon-

strated as a special case of the proposed binary warm-starting framework.

The work presented in this chapter resulted in the following publication:

- A. Bemporad and V. V. Naik. A numerically robust mixed-integer quadratic programming solver for embedded hybrid model predictive control. In *Proc. 6th IFAC Conference on Nonlinear Model Predictive Control*, pages 412–417, Madison, Wisconsin, USA, 2018.

Chapter 4 : Regularized moving-horizon PWA regression using mixed-integer quadratic programming

In this chapter, we present a novel regularized moving-horizon algorithm for *PieceWise Affine* (PWA) regression. At each iteration, an MIQP problem is formulated (and solved utilizing the GPAD-based MIQP solver presented in Chapter 2) to find the model parameters and active linear sub-model, which best match the training data within a relatively short time window in the past. The training samples are processed iteratively. The presented framework is properly adapted for the identification of Linear Parameter-Varying (LPV) systems. In addition, we exploit the binary warm-starting framework presented in Chapter 3 for efficient solution of the formulated MIQP problems.

The work presented in this chapter appears in the following publications:

- V. V. Naik, M. Mejari, D. Piga, and A. Bemporad. Regularized moving-horizon piecewise affine regression using mixed-integer quadratic programming. In *Proc. 25th Mediterranean Conference on Control and Automation*, pages 1349–1354, Valletta, Malta, 2017.

- M. Mejari, V. V. Naik, D. Piga, and A. Bemporad. Regularized moving-horizon PWA regression for LPV system identification. In *Proc. 18th IFAC Symposium on System Identification*, pages 1092–1097, Stockholm, Sweden, 2018b.

Chapter 5 : Energy Disaggregation using Embedded Binary Quadratic Programming

In this chapter, we consider an estimation problem commonly referred in the literature as *energy disaggregation problem* or non-intrusive load monitoring (NILM), which amounts to estimating the power consumption profiles of individual household appliances using only aggregated power measurements, essential to devise energy saving strategies.

Specifically, this chapter presents novel NILM methods using Binary Quadratic Programming (BQP) tailored for embedded implementation inside commercial smart energy meters, for a typical household where the number of appliances are moderate. The BQP solver used in this work is a special case of the Accelerated Dual Gradient Projection (GPAD) based branch and bound (B&B) approach presented in Chapter 2. Furthermore, we employ the regularized moving-horizon PieceWise Affine (PWA) regression algorithm proposed in Chapter 4 in order to identify the dynamic model for each appliance. The performance of the proposed approaches has been demonstrated using a household energy consumption benchmark dataset available in the literature.

The work described in this chapter led to the following contributions:

- V. V. Naik, M. Mejari, D. Piga, and A. Bemporad. Energy disaggregation using embedded binary quadratic programming. Submitted for publication, 2018.
- M. Mejari, V. V. Naik, D. Piga, and A. Bemporad. Energy disaggregation using piecewise affine regression and binary quadratic programming. In *Proc. 57th IEEE Conference on Decision and Control*, Miami Beach, FL, USA, 2018a.

The concluding remarks and an outline of possible future directions are contained in Chapter 6.

Chapter 2

Embedded Mixed-Integer Quadratic Optimization using Accelerated Dual Gradient Projection

2.1 Introduction

Solving an NP-hard problem like Mixed-Integer Quadratic Programming (MIQP) problem on an embedded platform poses a major challenge due to limited number of memory and computation resources. To address this issue, we propose to use branch and bound (B&B) algorithm combined with first-order methods to solve the relaxed Quadratic Programming (QP) subproblems. In this chapter we also introduce the approaches to find a suboptimal integer feasible solution of the MIQP problem: (i) without using B&B, and (ii) using B&B with significantly reduced number of QP relaxations. The suggested approaches need only basic arithmetic operations which makes them particularly well suited for embedded implementation.

2.1.1 Motivation

Nesterov's fast gradient method (Nesterov, 1983, 2004) has recently received great attention among the QP solving methods utilized for embedded applications by the embedded control community (Richter et al., 2011; Kögel and Findeisen, 2011; Patrinos and Bemporad, 2014; Giselsson, 2014). The primary reasons for this are its simplicity, relatively good performance, and good bounds on the worst-case number of iterations. These characteristics make the method favorable for real-time embedded applications. Jerez et al. (2013) demonstrated Field-Programmable Gate Array (FPGA) implementation of fast gradient method for embedded Model Predictive Control (MPC). FPGA implementation of Nesterov's fast gradient method and of the Alternating Direction Method of Multipliers (ADMM) for embedded MPC, and an analysis for reduced precision fixed-point arithmetic was provided in (Jerez et al., 2014). Patrinos and Bemporad (2014) presented an accelerated gradient projection method applied to solve the dual QP problem (Accelerated Dual Gradient Projection, GPAD). An analysis of dual gradient projection algorithm for fixed-point implementation (Patrinos et al., 2013) and FPGA implementation was presented in (Rubagotti et al., 2016). Thus, these methods have been demonstrated to be suitable for embedded implementation.

2.1.2 Contributions

This chapter introduces three new approaches for solving embedded MIQPs using the *Accelerated Dual Gradient Projection* (GPAD) method: (i) using *branch and bound* (B&B) to find the optimal solution, (ii) without B&B to find a suboptimal integer feasible solution, (iii) combining heuristic and B&B to find a suboptimal integer feasible solution. The first approach exploits the fact that the same matrix structure is used in all QP relaxations, so that preconditioning and factorization is only required at the root node; subsequent QP relaxations require simply the removal of sign restrictions on some of the dual variables. In addition, the basic GPAD algorithm of Patrinos and Bemporad (2014) is extended to include restart, infeasibility detection, and early termination based on dual cost

to enhance the overall performance of the proposed scheme.

Furthermore, two heuristic approaches are presented with an objective of initializing the B&B algorithm with an upper bound on the optimal cost. The first heuristic is a simple modification in the GPAD algorithm which avoids using the branch and bound algorithm. The second method exploits the possibility of heuristic approach to pick the binary variables to be solved using branch and bound by fixing the remaining binaries to either of the bounds. Though there is no guarantee of convergence for the heuristic methods, in practice when it converges turns out to be quite effective in solving MIQPs approximately, in most cases very close to the optimal solution. In addition, the ideas presented for combining GPAD with B&B algorithm are extended for MIQP solving approaches using *Alternating Direction Method of Multipliers* (ADMM) algorithm (Boyd et al., 2011) and ADMM based on the *Operator Splitting Quadratic Program* (OSQP) solver (Stellato et al., 2017). The advantage of OSQP based MIQP solver is, it does not require strict convexity of the objective function. The effectiveness of the proposed approaches have been demonstrated using the numerical examples.

2.1.3 Outline

The organization of this chapter is as follows. In Section 2.2, the basic GPAD algorithm of Patrinos and Bemporad (2014) is extended to include some specific features to complement the requirements of QP subproblems arising within the B&B framework, which is presented in the subsequent Section 2.3. The two heuristic approaches to find suboptimal integer feasible solution are introduced in Section 2.4. Specifically, Section 2.4.1 describes the heuristic approach without B&B, and the "midway" approach combining the heuristic approach with B&B is described in Section 2.4.2. MIQP solving approaches based on ADMM are presented in Section 2.5. The numerical results of the proposed approaches are reported in Section 2.6. Finally, the concluding remarks are given in Section 2.7.

2.2 Accelerated dual gradient projection

We recall the Mixed-Integer Quadratic Programming (MIQP) problem introduced in Chapter 1, which is

$$\begin{aligned} \min_{z} \quad & V(z) \triangleq \frac{1}{2} z'Qz + c'z && \text{(2.1a)} \\ \text{s.t.} \quad & \ell \leq Az \leq u && \text{(2.1b)} \\ & Gz = g && \text{(2.1c)} \\ & \bar{A}_i z \in \{\bar{\ell}_i, \bar{u}_i\},\ i = 1, \ldots, p, && \text{(2.1d)} \end{aligned}$$

where $Q \in \mathbb{R}^{n \times n}$ is the Hessian matrix, $Q = Q' \succ 0$, $z \in \mathbb{R}^n$ is the optimization vector, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $\bar{A} \in \mathbb{R}^{p \times n}$ and $G \in \mathbb{R}^{q \times n}$, $\ell, u \in \mathbb{R}^m$, $\bar{\ell}, \bar{u} \in \mathbb{R}^p$ and $g \in \mathbb{R}^q$, $\ell \leq u$, $\bar{\ell} \leq \bar{u}$, and $p \leq m$. Binary constraints $z_i \in \{0, 1\}$ are a special case of (2.1d) in which $\bar{A}_i$ is the i-th row of the identity matrix and the corresponding ℓ, u values are $\bar{\ell}_i = 0$ and $\bar{u}_i = 1$. The QP relaxation of problem (2.1) is obtained by replacing the integrality constraint (2.1d) with

$$\bar{\ell}_i \leq \bar{A}_i z \leq \bar{u}_i,\ i = 1, \ldots, p. \tag{2.2}$$

which leads to the following relaxed QP problem

$$\begin{aligned} \min_{z} \quad & V(z) \triangleq \frac{1}{2} z'Qz + c'z && \text{(2.3a)} \\ \text{s.t.} \quad & \ell \leq Az \leq u && \text{(2.3b)} \\ & Gz = g && \text{(2.3c)} \\ & \bar{\ell} \leq \bar{A}z \leq \bar{u} && \text{(2.3d)} \end{aligned}$$

which is termed as QP relaxation of the MIQP problem (2.1). The dual QP of problem (2.3) is also convex and has the form

$$\begin{aligned} \max_{\lambda, \nu} \ & \Psi(\lambda, \nu) \triangleq -\frac{1}{2} \left[\begin{smallmatrix} \lambda \\ \nu \end{smallmatrix}\right]' \mathcal{A} Q^{-1} \mathcal{A}' \left[\begin{smallmatrix} \lambda \\ \nu \end{smallmatrix}\right] - d' \left[\begin{smallmatrix} \lambda \\ \nu \end{smallmatrix}\right] - \frac{1}{2} c' Q^{-1} c \\ \text{s.t.} \quad & \lambda \geq 0,\ \nu \text{ free} && \text{(2.4a)} \end{aligned}$$

where

$$\mathcal{A} = \begin{bmatrix} A \\ -A \\ \bar{A} \\ -\bar{A} \\ G \end{bmatrix},\ \mathcal{B} = \begin{bmatrix} u \\ -\ell \\ \bar{u} \\ -\bar{\ell} \\ g \end{bmatrix},\ d = \begin{bmatrix} d_u \\ d_\ell \\ d_{\bar{u}} \\ d_{\bar{\ell}} \\ f \end{bmatrix} = \mathcal{B} + \mathcal{A} Q^{-1} c \tag{2.4b}$$

and $\lambda = \begin{bmatrix} \lambda_u \\ \lambda_\ell \\ \lambda_{\bar{u}} \\ \lambda_{\bar{\ell}} \end{bmatrix}$, $\lambda_u, \lambda_\ell, d_u, d_\ell \in \mathbb{R}^m$, $\lambda_{\bar{u}}, \lambda_{\bar{\ell}}, d_{\bar{u}}, d_{\bar{\ell}} \in \mathbb{R}^p$, $\nu, f \in \mathbb{R}^q$.

Algorithm 1 extends the formulation of Nesterov's fast gradient method proposed by Patrinos and Bemporad (2014) on the dual QP problem of the form (2.4) in order to solve QP problem (2.3).

Algorithm 1 Accelerated dual gradient projection method to solve QP problem of the form (2.3)

Input: matrices Q, A, G, $\bar{A}$, and vectors c, ℓ, u, g, $\bar{\ell}$, $\bar{u}$;

1: $H \leftarrow \mathcal{A}Q^{-1}\mathcal{A}'$, $L \leftarrow \|H\|_F$ or $L \leftarrow \lambda_{max}(H)$,
$\mathcal{A}_L \leftarrow \frac{1}{L}\mathcal{A}$, $\mathcal{B}_L \leftarrow \frac{1}{L}\mathcal{B}$;
2: $\lambda_0, \lambda_{-1} \leftarrow 0$, $\nu \leftarrow 0$;
3: $k \leftarrow 0$;
4: **repeat**
5: $\quad \beta_k \leftarrow \max\left\{\frac{k-1}{k+2}, 0\right\}$;
6: $\quad \begin{bmatrix} w_k \\ w_{eq,k} \end{bmatrix} \leftarrow \begin{bmatrix} \lambda_k \\ \nu_k \end{bmatrix} + \beta_k \left(\begin{bmatrix} \lambda_k \\ \nu_k \end{bmatrix} - \begin{bmatrix} \lambda_{k-1} \\ \nu_{k-1} \end{bmatrix} \right)$;
7: $\quad z_k \leftarrow -Q^{-1}\mathcal{A}' \begin{bmatrix} w_k \\ w_{eq,k} \end{bmatrix} - Q^{-1}c$;
8: $\quad \begin{bmatrix} s_k \\ s_{eq,k} \end{bmatrix} \leftarrow (\mathcal{A}_L z_k - \mathcal{B}_L)$;
9: $\quad \lambda_{k+1} \leftarrow \max\{w_k + s_k, 0\}$;
10: $\quad \nu_{k+1} \leftarrow w_{eq,k} + s_{eq,k}$;
11: **until** convergence;
12: $z^* \leftarrow z_k$, $\lambda^* \leftarrow w_k$, $\nu^* \leftarrow w_{eq,k}$;
13: $a^* \leftarrow \mathcal{A}' \begin{bmatrix} \lambda^* \\ \nu^* \end{bmatrix}$, $V^* \leftarrow -\frac{1}{2}(a^*)'Q^{-1}a^* - \mathcal{B}' \begin{bmatrix} \lambda^* \\ \nu^* \end{bmatrix} - (Q^{-1}c)'(a^* + \frac{1}{2}c) = \Psi^*$.

Output Primal solution z^*, optimal cost V^*, dual solution (λ^*, ν^*).

Note that the only difference between inequality constraints (Step 9) and equality constraints (Step 10) in Algorithm 1 is simply the sign restriction of the corresponding dual variables. This simple observation will be exploited in the B&B approach described in Section 2.3 when fixing binary constraints during branching.

2.2.1 Stopping criteria

The iterations of Algorithm 1 are stopped when the primal feasibility criterion

$$\begin{aligned} s_k^j &\leq \frac{1}{L}\epsilon_G,\ \forall j = 1,\dots,2(m+p) \\ |s_{eq,k}^j| &\leq \frac{1}{L}\epsilon_G,\ \forall j = 1,\dots,q \end{aligned} \tag{2.5}$$

and the optimality criterion

$$-\left[\begin{smallmatrix} w_k \\ w_{eq,k} \end{smallmatrix}\right]' \left[\begin{smallmatrix} s_k \\ s_{eq,k} \end{smallmatrix}\right] \leq \frac{1}{L}\epsilon_V \tag{2.6}$$

are satisfied, where $\epsilon_G > 0$ is the feasibility tolerance and $\epsilon_V \geq 0$ the optimality tolerance (Patrinos and Bemporad, 2014). Condition (2.6) derives from the duality gap calculation $V(z_k) - V^* \leq V(z_k) - \Psi([\begin{smallmatrix} w_k \\ w_{eq,k} \end{smallmatrix}]) = -[\begin{smallmatrix} w_k \\ w_{eq,k} \end{smallmatrix}]' [\begin{smallmatrix} s_k \\ s_{eq,k} \end{smallmatrix}] \leq \frac{1}{L}\epsilon_V$.

2.2.2 Preconditioning

It is well known that in first-order optimization methods the number of iterations strongly depends on how the problem matrices are scaled. Preconditioning the problem by appropriate scaling is usually adopted for (often largely) improving performance. In this work we adopt the Jacobi diagonal scaling (Bertsekas, 2009) of the dual Hessian matrix $H = \mathcal{A}Q^{-1}\mathcal{A}'$:

$$\theta_j \triangleq \frac{1}{\sqrt{\mathcal{A}_j Q^{-1} \mathcal{A}_j'}} \tag{2.7a}$$

$$\begin{aligned} \mathcal{A}_j \leftarrow \theta_j \mathcal{A}_j,\ \mathcal{B}_j \leftarrow \theta_j \mathcal{B}_j \\ j = 1,\dots,2(m+p)+q. \end{aligned} \tag{2.7b}$$

2.2.3 Restart

Accelerated gradient methods do not guarantee that the objective function decreases monotonically during iterations, and indeed ripples in the

sequence of function values are often observed. Restarting the sequence of scalars β_k in Step 5 of Algorithm 1 can largely improve the convergence property of the method. We use the gradient-based adaptive restart idea of (O' Donoghue and Candès, 2015; Giselsson and Boyd, 2014) for the dual problem (2.4) by checking the following condition

$$-\nabla\Psi\left(\left[\begin{smallmatrix} w_k \\ w_{eq,k} \end{smallmatrix}\right]\right)' \left(\left[\begin{smallmatrix} \lambda_{k+1} \\ \nu_{k+1} \end{smallmatrix}\right] - \left[\begin{smallmatrix} \lambda_k \\ \nu_k \end{smallmatrix}\right]\right) > 0. \tag{2.8}$$

Thus, whenever condition (2.8) is satisfied, the value of scalar k becomes $k \leftarrow 0$, this in turn resets the momentum term β_k to zero. The advantage of the gradient-based restart condition (2.8) is that it can be immediately computed by available quantities.

2.2.4 Infeasibility detection

Infeasibility detection of first-order methods was investigated in (Raghunathan and Di Cairano, 2014; O' Donoghue et al., 2016). The QP problem (2.3) may be infeasible, a case that frequently occurs during a B&B procedure. In this case the dual cost $\Psi(\lambda_k, \nu_k)$ tends to $+\infty$. The following Lemma 1 characterizes the asymptotic behavior of Algorithm 1 in case of QP infeasibility.

Lemma 1 *Let the QP problem (2.3) be infeasible. Then* $\lim_{k\to\infty} \mathcal{A}' \left[\begin{smallmatrix} w_k \\ w_{eq,k} \end{smallmatrix}\right] / \| \left[\begin{smallmatrix} w_k \\ w_{eq,k} \end{smallmatrix}\right] \|_\infty = 0.$

Proof. Let $\eta_k = \left[\begin{smallmatrix} w_k \\ w_{eq,k} \end{smallmatrix}\right]$. Since $H = \mathcal{A}Q^{-1}\mathcal{A}' \succeq 0$, for $\Psi(w_k, w_{eq,k}) \to +\infty$ it must occur that $d'\eta_k \to -\infty$, and therefore some components $\eta_{k,i}$ must tend to $+\infty$ for corresponding negative entries in vector d. Assume now by contradiction that the quantity $\mathcal{A}' \frac{\eta_k}{\|\eta_k\|_\infty}$ does not go to zero for $k \to \infty$. In this case there would exist a subsequence η_r such that $\|\mathcal{A}' \frac{\eta_r}{\|\eta_r\|_\infty}\|_2 \geq \epsilon$ for some $\epsilon > 0$. In this case, $\Psi(\lambda_r, \nu_r) = -\frac{1}{2}\eta_r'\mathcal{A}Q^{-1}\mathcal{A}'\eta_r - d'\eta_r \leq -\frac{1}{2\lambda_{\max}(Q)}\|\mathcal{A}'\eta_r\|_2^2 + \|d\|_2\|\eta_r\|_2 \leq -\frac{\epsilon^2}{2\lambda_{\max}(Q)}\|\eta_r\|_\infty^2 + \|d\|_2\sqrt{2(m+p)+q}\|\eta_r\|_\infty$, where $\lambda_{\max}(Q)$ is the largest eigenvalue of Q. Since some components of η_r diverge, $\|\eta_r\|_\infty$ diverges as well, and therefore $\Psi(w_r, w_{eq,r}) \leq 0$ for r sufficiently large. This contradicts the fact that $\Psi(w_k, w_{eq,k}) \to +\infty$, and therefore $\mathcal{A}' \frac{\eta_k}{\|\eta_k\|_\infty}$ must go to zero asymptotically if the QP is infeasible. ■

Motivated by Lemma 1, we propose the infeasibility detection criterion summarized in Algorithm 2, where $\epsilon_I > 0$ is a given infeasibility detection tolerance. By letting $\mu_k = \frac{w_k}{\|\eta_k\|_\infty}$, $\pi_k = \frac{w_{eq,k}}{\|\eta_k\|_\infty}$, $\mu_k \in \mathbb{R}^{2(m+p)}$, $\pi_k \in \mathbb{R}^q$ the criterion in Step 2 of Algorithm 2 amounts to verify the following condition

$$\begin{cases} \mathcal{A}' \left[\begin{smallmatrix} \mu_k \\ \pi_k \end{smallmatrix}\right] \approx 0 \\ \mathcal{B}' \left[\begin{smallmatrix} \mu_k \\ \pi_k \end{smallmatrix}\right] < 0 \\ \mu_k \geq 0. \end{cases} \tag{2.9}$$

According to Farkas Lemma (Rockafellar, 1970, p. 201), condition (2.9) is equivalent to an indication of the infeasibility of the QP (2.3). Moreover when $\mathcal{A}' \left[\begin{smallmatrix} \mu_k \\ \pi_k \end{smallmatrix}\right] = 0$, the condition $\mathcal{B}' \left[\begin{smallmatrix} \mu_k \\ \pi_k \end{smallmatrix}\right] < 0$ can be equivalently replaced by $d' \left[\begin{smallmatrix} \mu_k \\ \pi_k \end{smallmatrix}\right] < 0$.

Algorithm 2 Infeasibility detection

1: $\alpha_k \leftarrow \| \left[\begin{smallmatrix} w_k \\ w_{eq,k} \end{smallmatrix}\right] \|_\infty$;
2: **if** $\|\mathcal{A}' \left[\begin{smallmatrix} w_k \\ w_{eq,k} \end{smallmatrix}\right] \|_\infty \leq \epsilon_I \alpha_k$ **and** $d' \left[\begin{smallmatrix} w_k \\ w_{eq,k} \end{smallmatrix}\right] < -\epsilon_I \alpha_k$ **then**
3: **stop** (problem is infeasible)
4: **end if**

2.2.5 Early stopping criterion for the objective function

Assume that we want to stop the QP algorithm if we are not interested in getting a solution whose objective is greater than a given value V_0. In this case, since $V(z_k) \geq \Psi(\lambda_k, \nu_k) \geq V_0$, Algorithm 1 can be stopped if

$$\Psi(\lambda_k, \nu_k) \geq V_0. \tag{2.10}$$

This condition is particularly advantageous in a B&B setting to prematurely terminate the execution of the QP algorithm solving the relaxation, where V_0 is the best known cost associated with an integer-feasible solution.

Algorithm 3 MIQP solver based on GPAD

Input: MIQP problem matrices $Q = Q' \succ 0$, A, $\bar{A}$, G and vectors ℓ, u, g, $\bar{\ell}$, $\bar{u}$; feasibility tolerance $\epsilon \geq 0$.

1. **set** $V_0 \leftarrow +\infty$; $\zeta^* \leftarrow \emptyset$; $\mathcal{I}_{\bar{\ell}} \leftarrow \emptyset$;$\mathcal{I}_{\bar{u}} \leftarrow \emptyset$; $\mathcal{J} \leftarrow \{1, \ldots, p\}$; $\mathcal{T} \leftarrow \{\mathcal{I}_{\bar{\ell}}, \mathcal{I}_{\bar{u}}\}$; $\mathcal{S} \leftarrow \{\mathcal{T}\}$;
2. **while** $\mathcal{S} \neq \emptyset$ **do**:
   - 2.1. $\{\mathcal{I}_{\bar{\ell}}, \mathcal{I}_{\bar{u}}\} \leftarrow$ last element $\mathcal{T}$ of $\mathcal{S}$; $\mathcal{S} \leftarrow \mathcal{S} \setminus \{\mathcal{T}\}$;
   - 2.2. **execute** Algorithm 1 to solve (2.1a)–(2.1c), (2.11) under condition (2.10);
   - 2.3. **if** the solution z^*, V^* is returned **then**
     - 2.3.1. **if** $\mathcal{I}_{\bar{\ell}} \cup \mathcal{I}_{\bar{u}} = \{1, \ldots, p\}$ **or** $t_i \triangleq \bar{A}_i z^* \in \{\bar{\ell}_i, \bar{u}_i\}, \forall i \in \mathcal{J}$ **then** $V_0 \leftarrow V^*$, $\zeta^* \leftarrow z^*$; **otherwise**
       - 2.2.3.1. $j \leftarrow \arg\min_{i \in \mathcal{J}} \left| t_i - \frac{\bar{\ell}_i + \bar{u}_i}{2} \right|$;
       - 2.2.3.2. $\mathcal{T}_0 \leftarrow \{\mathcal{I}_{\bar{\ell}} \cup \{j\}, \mathcal{I}_{\bar{u}}\}$; $\mathcal{T}_1 \leftarrow \{\mathcal{I}_{\bar{\ell}}, \mathcal{I}_{\bar{u}} \cup \{j\}\}$;
       - 2.2.3.3. **if** $t_j \leq \frac{\bar{\ell}_i + \bar{u}_i}{2}$ **then** append $\{\mathcal{T}_1, \mathcal{T}_0\}$ to $\mathcal{S}$ **otherwise** append $\{\mathcal{T}_0, \mathcal{T}_1\}$ to $\mathcal{S}$;
3. **if** $V_0 = +\infty$ **then** (2.1) infeasible **otherwise** $\mathcal{V}^* \leftarrow V_0$;
4. **end**.

Output: Solution ζ^* of the MIQP problem (2.1), optimal cost $\mathcal{V}^*$, or infeasibility status.

2.3 Branch and Bound MIQP Algorithm

Algorithm 3 describes a B&B solver for the MIQP problem (2.1) that is based on the QP solver described by Algorithm 1 and its extensions presented in Section 2.2.

The sets $\mathcal{I}_{\bar{\ell}}$, $\mathcal{I}_{\bar{u}}$, $\mathcal{J}$ denote a partition of the set of indices $i \in \{1, \ldots, p\}$ ($\mathcal{I}_{\bar{u}} \cap \mathcal{I}_{\bar{\ell}} = \emptyset$, $\mathcal{J} = \{1, \ldots, p\} \setminus (\mathcal{I}_{\bar{u}} \cup \mathcal{I}_{\bar{\ell}})$) such that the integrality con-

straints (2.1d) are changed to

$$\bar{A}_{\mathcal{I}_{\bar{u}}} z = \bar{u}_{\mathcal{I}_{\bar{u}}} \tag{2.11a}$$

$$\bar{A}_{\mathcal{I}_{\bar{\ell}}} z = \bar{\ell}_{\mathcal{I}_{\bar{\ell}}} \tag{2.11b}$$

$$\bar{\ell}_{\mathcal{J}} \leq \bar{A}_{\mathcal{J}} z \leq \bar{u}_{\mathcal{J}} \tag{2.11c}$$

in a given QP relaxation.

At Step 1, all integrality constraints (2.1d) are relaxed to (2.2), that is $\mathcal{I}_{\bar{\ell}} = \mathcal{I}_{\bar{u}} = \emptyset$. The tuple $\mathcal{T} = \{\mathcal{I}_{\bar{\ell}}, \mathcal{I}_{\bar{u}}\}$ uniquely identifies a QP relaxation. The set $\mathcal{S}$ of tuples denotes the set of pending relaxations to be solved by Algorithm 1.

At Step 2.1, the element $\mathcal{T} = \{\mathcal{I}_{\bar{\ell}}, \mathcal{I}_{\bar{u}}\}$ is popped from the stack $\mathcal{S}$ and the corresponding QP relaxation (2.1a)–(2.1c), (2.11) is solved at Step 2.2, under the additional stopping criterion (2.10), where V_0 cost of the best integer-feasible solution found so far.

Step 2.3.1 is only executed if the QP relaxation was feasible and did not halt because the condition $V^* > V_0$ was verified by (2.10). In this case, Step 2.3.1 checks whether all integrality constraints are satisfied and eventually updates the best known integer-feasible solution ζ^* and its corresponding cost V_0 to the value V^*. Otherwise, branching is executed at Steps 2.2.3.1–2.2.3.3 by picking up the index j corresponding to the quantity $t_i = \bar{A}_i z$ that is most distant from $\bar{\ell}_i, \bar{u}_i$ (Step 2.2.3.1). Such a constraint is moved from the set of inequality constraints to the set of equality constraints and two new QP relaxations $\mathcal{T}_0$, $\mathcal{T}_1$ are formed at Step 2.2.3.2. Step 2.2.3.3 push the new problems $\mathcal{T}_0$, $\mathcal{T}_1$ to the stack $\mathcal{S}$ so that the problem with the least fractional part will be solved first.

Once no more QP relaxations are left, Step 3 checks if the value of V_0 is still $+\infty$, and in this case the MIQP problem (2.1) is reported infeasible.

2.3.1 Exploiting the fixed structure of dual QP relaxations

During the execution of the B&B algorithm, from one QP relaxation to another only the relaxed constraints (2.11) change. These simply map to the modified constraints

$$\lambda_{\bar{u},\mathcal{I}_{\bar{u}}} = w_{\bar{u},\mathcal{I}_{\bar{u}}} + s_{\bar{u},\mathcal{I}_{\bar{u}}}, \;\; \lambda_{\bar{\ell},\mathcal{I}_{\bar{u}}} = 0 \tag{2.12a}$$

$$\lambda_{\bar{\ell},\mathcal{I}_{\bar{\ell}}} = w_{\bar{\ell},\mathcal{I}_{\bar{\ell}}} + s_{\bar{\ell},\mathcal{I}_{\bar{\ell}}}, \quad \lambda_{\bar{u},\mathcal{I}_{\bar{\ell}}} = 0 \tag{2.12b}$$

$$\lambda_{\bar{u},\mathcal{J}} \geq 0, \qquad \lambda_{\bar{\ell},\mathcal{J}} \geq 0 \tag{2.12c}$$

in the dual QP problem, which determines a minor change in Step 9 of Algorithm 1: the max with 0 is not taken for the components of λ_{k+1} corresponding to $\lambda_{\bar{\ell},\mathcal{I}_{\bar{\ell}}}$, $\lambda_{\bar{u},\mathcal{I}_{\bar{u}}}$, and the components corresponding to $\lambda_{\bar{\ell},\mathcal{I}_{\bar{u}}}$, $\lambda_{\bar{u},\mathcal{I}_{\bar{\ell}}}$ are zeroed, therefore avoiding updating the corresponding quantities in w_k, s_k during the execution of Algorithm 1.

This is a very attractive feature due to the use of a dual QP solver, as the same QP structure (matrices and preconditioning) can be computed just once for initial relaxation at the root node and maintained unaltered during further branching.

2.3.2 Warm-starting the QP subproblems

When the binary tree is explored, at each node only one binary variable is fixed to either $\bar{\ell}_i$ or $\bar{u}_i$ with respect to the corresponding parent problem. The warm-start can be useful to recycle information from the solution of the parent problem. One approach is to use the solution of the parent problem to warm-start the child problems.

For the second approach, let the solution of the parent problem be z^*. We denote the set indices of the active constraints at the solution by $\mathcal{I}_\text{a}$ then

$$\mathcal{A}_{\mathcal{I}_\text{a}} z^* = \mathcal{B}_{\mathcal{I}_\text{a}}$$

and the remaining inactive constraints are satisfied with strict inequality.

The branching results in two children problems, specifically two problems with j-th variable fixed to $\bar{\ell}_j$ and $\bar{u}_j$, which is done by adding constraints $\bar{A}_j z = \bar{u}_j$ and $\bar{A}_j z = \bar{\ell}_j$ at Step 2.2.3.2 of Algorithm 3. With a slight abuse of the notation, the general form $\mathcal{A}_i z = \mathcal{B}_i$ is used to represent the index of either $\bar{A}_j z = \bar{u}_j$ or $\bar{A}_j z = \bar{\ell}_j$ constraint.

Using this notation, the j-th constraint $\mathcal{A}_i z^* < \mathcal{B}_i$ in the parent problem is

$$-\mathcal{A}_i Q^{-1} \mathcal{A}' \left[\begin{smallmatrix} \lambda^* \\ \nu^* \end{smallmatrix}\right] < \mathcal{B}_i + \mathcal{A} Q^{-1} c$$

Now the child problem enforces the equality constraint

$$r_i = \mathcal{B}_i + \mathcal{A}Q^{-1}c + \mathcal{A}_i Q^{-1}\mathcal{A}' \left[\begin{smallmatrix} \lambda^* \\ \nu^* \end{smallmatrix}\right] = 0$$

This can be done by

$$\lambda_0 = \lambda^* \text{ with } \lambda_{0i} = \lambda_i^* - \alpha r_i$$

where λ^* denotes the optimal dual vector of the parent problem, λ_0 denotes dual guess for the child problem, α is defined as

$$\alpha = \max_{k:r_k>0} \left(\frac{\lambda_k^*}{r_k} \right)$$

with $r = \mathcal{B} - \mathcal{A}z^*$.

2.4 Heuristic methods for suboptimal binary-feasible MIQP solutions

In this section we propose two heuristic approaches which explore the possibility to obtain suboptimal integer feasible solution of the MIQP problem (2.1), first without using B&B, and second by selecting a few variables to be solved using B&B. These approaches aid the B&B algorithm by providing an upper bound on the cost of the MIQP problem upfront. Therefore these can be considered as presolvers.

Thus, the suboptimal integer feasible solution of the heuristic approach which is denoted by V_H^* can play an important role. First, it can be used as upper bound on the optimal cost V_0 in B&B, so the QP relaxations in which the cost goes above V_H^* can be terminated prematurely by checking the condition (2.10), and the corresponding node being fathomed. Second, if the B&B fails to find better integer feasible solution within the specified time limit, then V_H^* can be used as a local solution of the problem.

The proposed heuristic methods are based on the adopted dual gradient projection framework and is similar to the one described in (Takapoui et al., 2016) for the alternating direction method of multipliers (ADMM).

2.4.1 Heuristic approach without using B&B

Let z_H^* denote the solution of presented heuristic approach, then since either $\bar{A}_i z_H^* = \bar{\ell}_i$ or $\bar{A}_i z_H^* = \bar{u}_i$ must hold to satisfy the integrality constraint (2.1d), the corresponding dual variables are such that either $\lambda^*_{\bar{u}_i H} = 0$ and $\lambda^*_{\bar{\ell}_i H}$ unconstrained/unrestricted in sign, or vice versa for all $i = 1, \ldots, p$. In other words, the vector $\begin{bmatrix} \lambda^*_{\bar{u}H} \\ \lambda^*_{\bar{\ell}H} \end{bmatrix}$ must belong to the nonconvex set given by the union of the orthogonal real axes. We propose the heuristic described in Algorithm 4 to defining the values of $\lambda_{\bar{\ell}}$, $\lambda_{\bar{u}}$ during the execution of Algorithm 1 to impose such a nonconvex constraint.

Algorithm 4 Heuristic approach for suboptimal binary feasible MIQP solution

$\begin{bmatrix} \lambda_{u_{k+1}} \\ \lambda_{\ell_{k+1}} \end{bmatrix} \leftarrow \max\left\{ \begin{bmatrix} w_{u_k} + s_{u_k} \\ w_{\ell_k} + s_{\ell_k} \end{bmatrix}, 0 \right\}$;
for all $i = 1, \ldots, p$ **do**
  if $\bar{A}_i z_k \geq \frac{\bar{l}_i + \bar{u}_i}{2}$ **then**
    $\lambda_{\bar{u}_{i_{k+1}}} \leftarrow w_{\bar{u}_{i_k}} + s_{\bar{u}_{i_k}}$;
    $\lambda_{\bar{\ell}_{i_{k+1}}} \leftarrow 0$;
  else
    $\lambda_{\bar{u}_{i_{k+1}}} \leftarrow 0$;
    $\lambda_{\bar{\ell}_{i_{k+1}}} \leftarrow w_{\bar{\ell}_{i_k}} + s_{\bar{\ell}_{i_k}}$;
  end if
end for

Assuming that the Algorithm 4 converges, this forces the quantity $\bar{A}_i z_H^*$ to satisfy the integrality constraints (2.1d). We propose to apply the quantization described in Algorithm 4 only after the QP relaxation (2.3) is solved by Algorithm 1 to optimality and the problem is found feasible but not integer feasible. In this case, Algorithm 1 is executed again from Step 4 where now Step 9 is replaced by Algorithm 4, until stopping criteria (2.5) and (2.6) are satisfied. We replace $-\begin{bmatrix} w_k \\ w_{eq,k} \end{bmatrix}' \begin{bmatrix} s_k \\ s_{eq,k} \end{bmatrix}$ with $\left| -\begin{bmatrix} w_k \\ w_{eq,k} \end{bmatrix}' \begin{bmatrix} s_k \\ s_{eq,k} \end{bmatrix} \right|$ in condition (2.6) when executing Algorithm 1.

2.4.2 Mid-way Heuristic approach

As the name suggests, the "mid-way" heuristic approach is a combination between solving MIQPs for exact solutions using B&B and finding approximate solutions using heuristic without B&B. The idea here is to explore the possibility to fix a number of binary variables $\bar{A}_i z$ to either $\bar{\ell}_i$ or $\bar{u}_i$ using a heuristic and, pick the rest of them to be solved using B&B by exploiting the information on offer by the solution of QP relaxation.

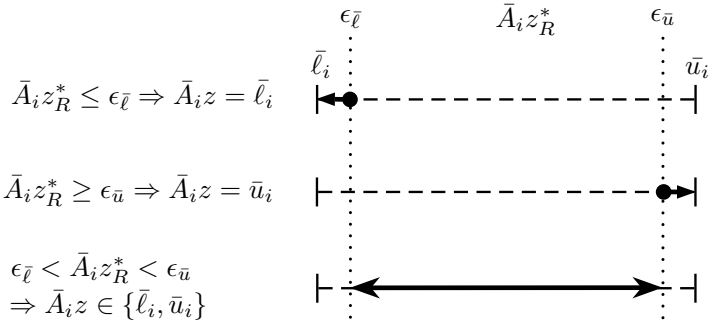


Figure 5: Conditions for the binary variables $\bar{A}_i z_R^*$ w.r.t $\epsilon_{\bar{\ell}}$ and $\epsilon_{\bar{u}}$ for $i = 1, \ldots, p$. Only $\epsilon_{\bar{\ell}} < \bar{A}_i z_R^* < \epsilon_{\bar{u}}$ are considered to be solved with B&B, while the others are set to equality as shown.

The overall idea is sketched in Figure 5. We denote the solution of QP relaxation (2.3) as z_R^*. If z_R^* is found to be feasible but not integer feasible, then the value of relaxed binary variables $\bar{A}_i z_R^*$ for each $i = 1, \ldots, p$ and $[\bar{\ell}_i, \bar{u}_i]$ is observed with respect to the tolerances $\epsilon_{\bar{\ell}}$, $\epsilon_{\bar{u}}$ as reference points, where $\bar{\ell}_i \leq \epsilon_{\bar{\ell}} < \frac{\bar{\ell}_i + \bar{u}_i}{2}$ and $\frac{\bar{\ell}_i + \bar{u}_i}{2} < \epsilon_{\bar{u}} \leq \bar{u}_i$. Then sets $\mathcal{K}_{\bar{\ell}}, \mathcal{K}_{\bar{u}}$ and $\mathcal{H}$ are formed which corresponds to the binary variables satisfying $\bar{A}_i z_R^* \leq \epsilon_{\bar{\ell}}$, $\bar{A}_i z_R^* \geq \epsilon_{\bar{u}}$ and $\epsilon_{\bar{\ell}} < \bar{A}_i z_R^* < \epsilon_{\bar{u}}$ conditions respectively, where $\mathcal{K}_{\bar{u}} \cap \mathcal{K}_{\bar{\ell}} \cap \mathcal{H} = \emptyset$, $\mathcal{K}_{\bar{u}} \cup \mathcal{K}_{\bar{\ell}} \cup \mathcal{H} = \{1, \ldots, p\}$. Then, the binary variables in the set $\mathcal{H}$ are considered to be solved using branch and bound while considering the binary variables in sets $\mathcal{K}_{\bar{\ell}}, \mathcal{K}_{\bar{u}}$ as fixed equalities.

The order of execution of this approach is as follows:

- the QP relaxation (2.3) is solved by Algorithm 1 to optimality and the problem is found feasible but not integer feasible.

Algorithm 5 mid-way heuristic

$\mathcal{K}_{\bar{\ell}}, \mathcal{K}_{\bar{u}}, \mathcal{H} \leftarrow \emptyset$
$\mathcal{K}_{\bar{\ell}} \leftarrow \{i_\ell \in \{1, \ldots, p\} | \bar{A}_{i_\ell} z_R^* \leq \epsilon_{\bar{\ell}}\}$
$\mathcal{K}_{\bar{u}} \leftarrow \{i_u \in \{1, \ldots, p\} | \bar{A}_{i_u} z_R^* \geq \epsilon_{\bar{u}}\}$
$\mathcal{H} \leftarrow \{1, \ldots, p\} \setminus \{\mathcal{K}_{\bar{\ell}} \cup \mathcal{K}_{\bar{u}}\}$
$z_R^* \leftarrow z_k, \lambda_R^* \leftarrow \lambda_k, \nu_R^* \leftarrow \nu_k$ (solution of relaxed QP)

- In this case, Algorithm 5 is executed for the extraction of the indices, and optimal vectors representing solution of the relaxed problem.

- This is followed by execution of the heuristic approach as described in section 2.4.1.

If the heuristic approach gives a suboptimal integer feasible solution denoted by V_H^* then the upper bound on the cost as in Step 1 of B&B Algorithm 3 can be initialized with $V_0 \leftarrow V_H^*$. The binary variables belonging to the sets $\mathcal{K}_{\bar{\ell}}, \mathcal{K}_{\bar{u}}$ are fixed to $\bar{A}_i z = \bar{\ell}_i$, $\bar{A}_i z = \bar{u}_i$ respectively. Only the variables $\epsilon_{\bar{\ell}} < \bar{A}_i z_R^* < \epsilon_{\bar{u}} \in \mathcal{H}$ are considered to be solved using B&B Algorithm 3. We remark that the values $z_R^*, \lambda_R^*, \nu_R^*$ are same as solution of the root node of the branch and bound. Hence, we start the branch and bound Algorithm 3 directly from step 2.2.3.1.

In some time and/or resource constrained applications, the B&B algorithm can be leveraged but only for limiting number of binary variables. In such cases, if the problem to be solved consists of large number of binary variables, then the proposed approach can provide a heuristic tool to select possibly a few to be solved using B&B. The effectiveness of this approach is demonstrated in the numerical results section.

2.5 ADMM based MIQP solver

In this section we employ *Alternating Direction Method of Multipliers* (ADMM) in order to solve the relaxed QP subproblem within the B&B framework, exploiting the ideas presented in Section 2.3.

First, we adapt the ADMM algorithm from Boyd et al. (2011) for solving QP problems with a quadratic objective (2.3a), subject to constraints (2.11), is described in Algorithm 6 where $\rho > 0$ is the step-size, see (Boyd et al., 2011) for further details.

Algorithm 6 ADMM algorithm to solve QP problem quadratic objective (2.3a), subject to constraints (2.11)

Input: initial values z_0, x_0, w_0; Q, c, A, ℓ, u, $\bar{A}$, $\bar{\ell}$, $\bar{u}$, ρ;

1: **repeat**

2: $\quad z_{k+1} \leftarrow -(Q + \rho \begin{bmatrix} A \\ \bar{A} \end{bmatrix}' \begin{bmatrix} A \\ \bar{A} \end{bmatrix})^{-1}(\rho \begin{bmatrix} A \\ \bar{A} \end{bmatrix}' \begin{bmatrix} w_k - x_k \\ \bar{w}_k - \bar{x}_k \end{bmatrix} + c)$

3: $\quad \begin{bmatrix} x_{k+1} \\ \bar{x}_{k+1,\mathcal{J}} \end{bmatrix} \leftarrow \min \left\{ \max \left\{ \begin{bmatrix} A \\ \bar{A}_{\mathcal{J}} \end{bmatrix} z_{k+1} + \begin{bmatrix} w_k \\ \bar{w}_{k,\mathcal{J}} \end{bmatrix}, \begin{bmatrix} \ell \\ \bar{\ell}_{\mathcal{J}} \end{bmatrix} \right\}, \begin{bmatrix} u \\ \bar{u}_{\mathcal{J}} \end{bmatrix} \right\}$

4: $\quad (\bar{x}_{k+1})_i \leftarrow \begin{cases} \bar{\ell}_i & \text{if } i \in \mathcal{I}_{\bar{\ell}} \\ \bar{u}_i & \text{if } i \in \mathcal{I}_{\bar{u}} \end{cases}$

5: $\quad \begin{bmatrix} w_{k+1} \\ \bar{w}_{k+1} \end{bmatrix} \leftarrow \begin{bmatrix} w_k \\ \bar{w}_k \end{bmatrix} + \begin{bmatrix} A \\ \bar{A} \end{bmatrix} z_{k+1} - \begin{bmatrix} x_{k+1} \\ \bar{x}_{k+1} \end{bmatrix}$

6: **until** convergence;

Output Primal solution z^*, optimal cost V^*.

Stopping criteria

We employ the stopping criteria of (Boyd et al., 2011, Sec. 3.3), which is based on values of primal and dual residuals computed at every iteration given by

$$r_k^{\text{pri}} = Az_k - x_k$$
$$r_k^{\text{dual}} = \rho A'(x_k - x_{k-1}).$$

Algorithm 6 is stopped when the residual values are small, quantified by the conditions

$$\|r_k^{\text{pri}}\|_2 \leq \epsilon^{\text{pri}}$$
$$\|r_k^{\text{dual}}\|_2 \leq \epsilon^{\text{dual}}$$

are satisfied, where the tolerances $\epsilon^{\text{pri}} > 0$ and $\epsilon^{\text{dual}} > 0$ are for the primal and dual feasibility respectively.

Infeasibility detection

While combining ADMM algorithm to solve the QP subproblems within the B&B framework, infeasibility detection is essential as some of the integer combinations explored during B&B tree search might be infeasible. We use the infeasibility criteria proposed in (Raghunathan and Di Cairano, 2014), by checking the following set of conditions

$$\max(\rho \left\| x_k - x_{k-1} \right\|, \left\| w_k - w_{k-1} \right\|) > \epsilon_o \tag{2.13a}$$

$$\frac{\max(\rho \left\| z_k - z_{k-1} \right\|, \left\| x_k - x_{k-1} \right\|)}{\max(\rho \left\| x_k - x_{k-1} \right\|, \left\| w_k - w_{k-1} \right\|)} \leq \epsilon_r \tag{2.13b}$$

$$\frac{w_k'(x_k - z_k)}{\left\| w_k \right\| \left\| x_k - z_k \right\|} \geq 1 - \epsilon_a \tag{2.13c}$$

$$w_k \circ (x_k - z_k) \geq 0 \tag{2.13d}$$

where the tolerance values $0 \leq \epsilon_o,\ \epsilon_r,\ \epsilon_a \ll 1$, the operator $\circ$ denotes component-wise multiplication. The condition (2.13d) checks that each component of w_k and $(x_k - z_k)$ have the same sign, see (Raghunathan and Di Cairano, 2014) for the further details.

We utilize the B&B setup described in Algorithm 3, where at Step 2.2 we employ ADMM Algorithm 6, while ignoring the condition (2.10) which is not applicable in this case. We utilize preconditioning and warm-starting as described in the previous sections.

Exploiting the fixed structure of QP relaxations

An idea similar to the fixed structure for GPAD algorithm presented in Section 2.3.1 is adapted here for ADMM algorithm. Specifically, during the execution of B&B, the change in constraints (2.11) is handled by changing the corresponding updates of $\bar{x}$ as described in Step 3 and Step 4 of Algorithm 6. This is a very attractive feature that allows to use the same QP structure (matrices, preconditioning and factorization), and can be computed just once for initial relaxation at the root node and maintained unaltered during further branching. Thus, online solution of MIQP problem requires only basic arithmetic operations. Note that the approaches

proposed with GPAD, heuristic approach in Section 2.4.1 (a slightly different approach than Takapoui et al. (2016)) and a mid-way heuristic approach presented in Section 2.4.2 are readily adaptable for ADMM algorithm presented in this section.

2.5.1 OSQP-based MIQP solver

Next, we consider a new efficient solver based on the ADMM (Boyd et al., 2011), *Operator Splitting Quadratic Program* (OSQP) solver, which is recently developed by Stellato et al. (2017). The OSQP solver is coupled with the B&B algorithm for the solution to relaxed QP subproblems. For brevity, Algorithm 7 describes the OSQP solver to solve the relaxed QP subproblem of form (2.1a)–(2.1b) i.e., only with inequality constraints, and can handle equality constraints similar to described in Algorithm 6.

Algorithm 7 OSQP solver to solve QP problem of form (2.3a)–(2.3b)

Input: initial values z_0, x_0, w_0; Q, c, A, ℓ, u, ρ, σ, α;

1: **repeat**

2: solve $\begin{bmatrix} Q+\sigma I & A' \\ A & -\frac{1}{\rho}I \end{bmatrix} \begin{bmatrix} \hat{z}_{k+1} \\ \nu_{k+1} \end{bmatrix} = \begin{bmatrix} \sigma z_k - c \\ x_k - \frac{1}{\rho} w_k \end{bmatrix}$

3: $\hat{x}_{k+1} \leftarrow x_k + \frac{1}{\rho}(\nu_{k+1} - w_k)$

4: $z_{k+1} \leftarrow \alpha \hat{z}_{k+1} + (1-\alpha) z_k$

5: $x_{k+1} \leftarrow \Pi\left(\alpha \hat{x}_{k+1} + (1-\alpha) x_k + \frac{1}{\rho} w_k\right)$

6: $w_{k+1} \leftarrow w_k + \rho\left(\alpha \hat{x}_{k+1} + (1-\alpha) x_k - x_{k+1}\right)$

7: **until** convergence;

Output Primal solution z^*, optimal cost V^*.

Scalars $\rho, \sigma > 0$ are the step-size parameters and $\alpha \in (0, 2)$ is the relaxation-parameter. The operator Π denotes projection onto the set $[\ell, u]$ with the closed-form solution $\Pi(x) = \max\left(\min\left(x, u\right), \ell\right)$.

Stopping criteria

At each iteration, the algorithm produces iterates (z_k, x_k, w_k) whose primal and dual residuals are defined as

$$r_k^{\text{prim}} = Az_k - x_k$$
$$r_k^{\text{dual}} = Qz_k + c + A'w_k.$$

If problem (2.3a)–(2.3b) is solvable, the residuals converge to zero as $k \to \infty$ (Boyd et al., 2011). The algorithm stops when the Euclidean norm of the residuals is below predefined tolerances $\epsilon_{\text{prim}} > 0$ and $\epsilon_{\text{dual}} > 0$. Note that $\epsilon_{\text{prim}}, \epsilon_{\text{dual}}$ are often chosen relative to the scaling of problem iterates, see (Boyd et al., 2011, Sec. 3.3).

Infeasibility detection

If the problem is primal infeasible, the algorithm produces a vector $v \in \mathbb{R}^m$ serving as a certificate of infeasibility,

$$A'v = 0, \quad u'v_+ + \ell'v_- < 0,$$

where $v_+ = \max(v, 0) > 0$ and $v_- = \min(v, 0) < 0$. On the other hand, if the problem is dual infeasible, the algorithm generates a vector $s \in \mathbb{R}^n$ certifying dual infeasibility,

$$Qs = 0, \quad c's < 0, \quad (As)_i = \begin{cases} 0 & \ell_i, u_i \in \mathbb{R} \\ \geq 0 & u_i = +\infty, \ell_i \in \mathbb{R} \\ \leq 0 & \ell_i = -\infty, u_i \in \mathbb{R} \end{cases}.$$

For more details we refer the reader to (Banjac et al., 2017).

Solving linear system at Step 2 of Algorithm 7 imparts a major computations load, which is handled using sparse LDL' factorization (Davis, 2006) followed by forward and backward substitution. Since the coefficient matrix in this linear system is quasi-definite, it always has a well-defined LDL' factorization, with L being a lower triangular matrix with unit diagonal elements, and D a diagonal matrix with nonzero diagonal elements (Vanderbei, 1995).

Exploiting the fixed structure of QP relaxations

An idea similar to the fixed structure presented in Section 2.3.1 and Section 2.5, is utilized here which keeps the matrix constant throughout the execution of B&B algorithm, as it does not change with the changes in c, ℓ, u. Hence, the factorization is performed once at the root node and utilized throughout the execution of B&B. The other algorithm steps are computationally much cheaper and involve only scalar-vector multiplications, vector additions and element-wise projections (clipping).

2.6 Numerical results

Numerical experiments were carried out on a desktop computer with Intel Core i7-4700MQ CPU with 2.40 GHz and 8 GB of RAM, running MATLAB R2015a.

2.6.1 Heuristic approach-Hybrid vehicle example

First we consider the hybrid vehicle example described in (Takapoui et al., 2016), that consists of the combination of a battery, an electric motor/generator, and an engine. For a given power demand P_t^{des} at time $t = 0, \ldots, T-1$, the objective is to plan the battery power P_t^{batt} and engine power P_t^{eng} for the given time interval, such that $P_t^{batt} + P_t^{eng} \geq P_t^{des}$. Let E_t be the energy of the battery at time t, $E_{t+1} = E_t - \tau P_t^{batt}$, where τ is the sample time and $0 \leq E_t \leq E^{max}$. The fuel cost is given by $f(P_t^{eng}, z_t)$ where $f(P, z) = \alpha P^2 + \beta P + \gamma z$ and the constraint on power is $0 \leq P_t^{eng} \leq P^{max} z_t$. The hybrid vehicle control problem is

$$
\begin{aligned}
\min \quad & \eta(E_T - E^{max})^2 + \sum_{t=0}^{T-1} f(P_t^{eng}, z_t) + \delta(z_t - z_{t-1})_+ \\
\text{s.t.} \quad & E_{t+1} = E_t - \tau P_t^{batt} \\
& P_t^{batt} + P_t^{eng} \geq P_t^{des} \\
& z_t \in \{0, 1\}, \ t = 0, \ldots, T-1
\end{aligned}
$$

where P_t^{batt}, P_t^{eng}, z_t (engine on/off) and E_t are the optimization variables. The term $\delta(z_t - z_{t-1})_+$ penalizes the engine going from the off to the on state, where $\delta \geq 0$ for all t. By choosing $T = 72$ steps, the resulting MIQP problem has $n = 862$ optimization variables, $p = 72$ binary variables, $m = 503$ inequality constraints, and $q = 575$ equality constraints.

The solver GUROBI (Gurobi Optimization, Inc., 2014) computes the optimal cost $V^* = 135.9$ in 21.05 s with default options. The cost calculated by the ADMM-based heuristic approach of Takapoui et al. (2016), denoted as miqpADMM-H, is 138.1[1] and is obtained in 0.40 s for preconditioning + 3.55 s for solving the problem.

The performance of Algorithm 1+4+2 implemented in interpreted MATLAB code, denoted as miqpGPAD-H, is reported in Table 1 for different values of the feasibility tolerance ϵ_G and optimality tolerance ϵ_V. Whenever condition (2.8) is satisfied at a given iteration k, rather than restarting the values of β_k we just assign $w_k \leftarrow \lambda_k$, $w_{eq,k} \leftarrow \nu_k$ in Step 6 of Algorithm 1 for that iteration. We also introduce the regularization term $10^{-3}I$ in the primal Hessian matrix Q to make it positive definite. The tolerance value used in Algorithm 2 is $\epsilon_I = 10^{-2}$.

Table 1: Performance comparison with different values of ϵ_V, ϵ_G for miqpGPAD-H.

ϵ_V, ϵ_G	Cost	Precond., Solving	Constr. violation
10^{-2}, 10^{-2}	131.5	1.19 s, 3.81 s	$3.62 \cdot 10^{-1}$
10^{-2}, 10^{-3}	135.9	1.12 s, 8.62 s	$8.09 \cdot 10^{-3}$
10^{-3}, 10^{-3}	135.9	1.07 s, 8.67 s	$7.98 \cdot 10^{-3}$
10^{-3}, 10^{-4}	136.0	1.14 s, 13.37 s	$2.59 \cdot 10^{-3}$
10^{-4}, 10^{-3}	136.1	1.10 s, 15.87 s	$1.08 \cdot 10^{-3}$

Figure 6 shows the trajectories obtained with miqpGPAD-H for $\epsilon_V = 10^{-2}$, $\epsilon_G = 10^{-3}$ and compare them with the ones obtained by GUROBI and miqpADMM-H. It is apparent that the proposed heuristic approach

[1] https://github.com/cvxgrp/miqp_admm/tree/master/matlab/vehicle.m

is computationally faster and very simple to implement in an embedded control platform, and keeps the quality of the solution over a sufficient level for the practical application at hand.

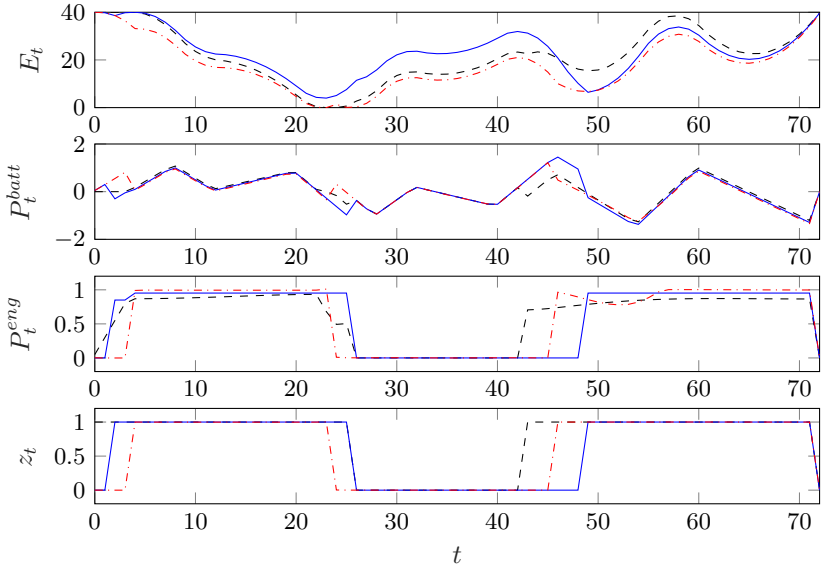


Figure 6: Battery energy, battery power, engine power and engine on/off signals versus time: GUROBI (solid blue line), miqpGPAD-H (dash-dotted red line), and miqpADMM-H (dashed black line).

2.6.2 Branch and Bound-random MIQPs

Next, we test the B&B method, denoted as miqpGPAD for Algorithm 3, and as miqpADMM for Algorithm 6, on randomly generated MIQP problems with n variables, m inequality constraints, p binary constraints, q equality constraints, and condition number $\kappa = 10$ of the primal Hessian Q[2]. Algorithm 3 is implemented in interpreted MATLAB code and

[2]The entries of matrix A are generated from the normal distribution $\mathcal{N}(0, 0.0025)$, ℓ, u from the uniform distribution $\mathcal{U}(0, 100)$, c from $\mathcal{N}(0, 1)$; matrix $Q = U\Sigma V'$, where U, V are orthogonal matrices generated by QR decomposition of random $n \times n$ matrices, and Σ is diagonal with nonzero entries having logarithms equally spaced between $\pm \log(\kappa)/4$ Bier-

Algorithms 1, 2, 4, 6 are implemented in Embedded MATLAB and complied. The tolerance values used in Algorithm 1 are $\epsilon_V, \epsilon_G = 10^{-5}$, in

Table 2: Average CPU time (ms) on random MIQP problems over 50 instances for each combination of n, m, p, q.

n	m	p	q	miqpGPAD	GUROBI
10	100	2	2	15.6	6.56
50	25	5	3	3.44	8.74
50	150	10	5	63.22	46.25
100	50	2	5	6.22	26.24
100	200	15	5	164.06	188.42
150	100	5	5	31.26	88.13
150	200	20	5	258.80	274.06
200	50	15	6	35.08	144.38

Algorithm 2 $\epsilon_I = 10^{-2}$. The CPU time reported for solving feasible MIQP problems, averaged over 50 executions with n variables, m inequality constraints, p binary constraints, q equality constraints are listed in Table 2.

Table 3: Average CPU time (ms) on random MIQP problems over 50 instances for each combination of n, m, p, q.

n	m	p	q	miqpGPAD	miqpADMM	GUROBI
10	100	2	2	6.98	4.10	3.05
50	25	5	3	5.24	5.00	9.37
50	200	10	5	46.92	30.20	37.81
100	50	2	0	3.42	3.12	14.51
100	200	15	5	63.46	61.84	121.33
150	100	5	5	11.26	11.84	64.00
150	300	20	0	250.42	271.46	298.27
200	50	15	6	18.24	21.74	108.36

In order to compare the performance of miqpGPAD against miqpADMM, the experiment is carried out with tolerance values for miqpGPAD $\epsilon_V, \epsilon_A = 10^{-3}$, $\epsilon_I = 10^{-2}$, and for miqpADMM are $\rho = 0.9$, $\epsilon^{\text{pri}} =$

laire et al. (1991).

$\epsilon^{\text{dual}} = 10^{-4}$. The observed numerical results are noted in Table 3, are the average computation time over 50 different problem instances.

Table 4: Average CPU time (ms) on random MIQP problems over 10 instances for each combination of n, m, p.

n	m	p	miOSQP	GUROBI
10	5	2	1.40	1.80
10	100	2	2.20	9.51
50	25	5	4.10	15.62
50	200	10	54.24	81.35
100	50	2	5.61	40.32
100	200	15	105.88	211.22
150	100	5	37.42	198.34
150	300	20	343.05	682.78

Table 4 describes the results of OSQP based MIQP solver named as miOSQP, utilized for solving randomly generated MIQP problems with varying dimensions n, m and number of integer variables p[3]. The solver miOSQP has been implemented in Python and interfaced to the OSQP compiled binaries from (Stellato and Banjac, 2017). Timing benchmarks are compared to GUROBI with the default options running Python 3.5. In addition, both algorithms are executed single-threaded for fairness[4]. Depending on the problem size, the Python overhead can be significant. For example, when $(n, m, p) = (10, 5, 2)$ OSQP takes only $4.6\,\%$ of the total computation time and GUROBI turns out to be faster. This suggests that further speedups could be obtained by using a low-level branch and bound implementation.

The results show that the proposed scheme performs well as compared to the commercial GUROBI solver, but has the advantage of a very simple coding, therefore making it very suitable for embedded control

[3] for randomly generated problems, the entries of Q are computed as $Q = MM'$ where $M \in \mathbb{R}^{n \times n}$ is generated from the uniform distribution $\mathcal{U}(0, 1)$ with 70% nonzero elements and the linear part of the cost q with the normal distribution $\mathcal{N}(0, 1)$. The constraints are generated as $A \sim \mathcal{U}(0, 1)$, $l \sim \mathcal{U}(0, 1) - 2$ and $l \sim \mathcal{U}(0, 1) + 2$.

[4] The code and all benchmark examples are available at https://github.com/oxfordcontrol/miosqp

applications.

2.6.3 Branch and Bound-PieceWise Affine (PWA) regression

Next, we consider the identification of Linear Parameter Varying (LPV) models as a PieceWise Affine (PWA) regression problem. The approach in (Mejari et al., 2018b) requires to solve an MIQP problem at every sampling instance in receding horizon fashion. We consider single-input and single-output (SISO)-LPV ARX data generating system:

$$y(k) = a_1^{\circ}(p(k))y(k-1) + a_2^{\circ}(p(k))y(k-2) + b_1^{\circ}(p(k))u(k-1) + e(k).$$

The p-dependent coefficients $a_1^{\circ}(p(k))$, $a_2^{\circ}(p(k)$ and $b_1^{\circ}(p(k))$ are described by the nonlinear functions:

$$a_1^{\circ}(p(k)) = \begin{cases} -0.5, & \text{if } p(k) > 0.5 \\ -p(k), & \text{if } -0.5 \leq p(k) \leq 0.5 \\ 0.5, & \text{if } p(k) < -0.5 \end{cases}$$
$$a_2^{\circ}(p(k)) = p^3(k), \;\; b_1^{\circ}(p(k)) = \sin(\pi p(k)).$$

A training data and a validation dataset of length $N = 1000$ and $N_{\text{val}} = 2000$, respectively, are generated. The training input u and the scheduling signal p are considered as independent white-noise processes with uniform distribution $\mathcal{U}(-1, 1)$. The standard deviation of the noise e is 0.05, which corresponds to SNR of 20 dB. A PWA model with $s = 6$ modes is considered with prediction horizon $T = 6$. Each MIQP subproblem, solved with GPAD-B&B, contains 36 binary and 72 continuous variables, 144 inequality and 6 equality constraints. In the second stage, off-line multicategory discrimination algorithm is executed for partitioning the scheduling space, and the estimated model parameters are refined based on the computed partition, using simple least-squares for each sub-model.

The Best fit rate calculated is 95% and the average time to solve the MIQP problem are shown in Table 5.

Table 5: Average CPU time for training sample N=1000.

Solver	Time (ms)
GUROBI	259.38
miqpGPAD	125.06

2.6.4 Mid-way heuristic approach-random MIQPs

We test the mid-way heuristic approach using randomly generated MIQP problems. Two separate experiments for MIQP problem are carried out, first with $n = 60$ variables, $m = 60$ inequality constraints, $q = 8$ equality constraints and $p = 50$ binary variables; and second with $n = 90$ variables, $m = 80$ inequality, $q = 8$ equality constraints and $p = 70$ binary variables. The tolerance values used in Algorithm 4 are $\epsilon_{\bar{\ell}} = 0.01$ and $\epsilon_{\bar{u}} = 0.99$, and the results are shown in Figure 7 and 8. In both the cases it is observed that the rounded mean value over 50 runs of the actual number of binary variables solved using B&B are 19 and 25 which is approximately 37% of the original number of binary variables 50 and 70 respectively for the given set of problems. The results shows this approach renders solution very close to the global solution.

2.6.5 ARX model segmentation - Heuristic, Mid-way heuristic approach

In this Section, we present the application of the proposed heuristic approaches to the problems arising in the field of system identification. More specifically, we consider the *Auto-Regressive eXogenous (ARX)* model segmentation problem which can be solved using sparse optimization with ℓ_0 norm.

Mixed-Integer Quadratic Programming approach for exact solution of the ℓ_0 norm

The problem of finding the sparsest solution satisfying a system of linear equations $A\theta = b$ with more unknowns than equalities can be formulated

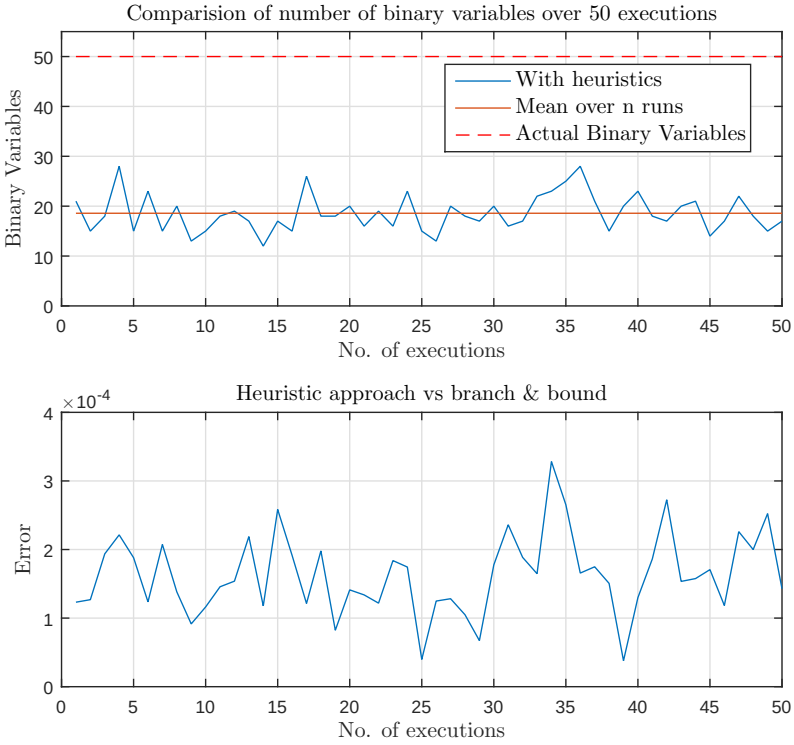


Figure 7: Number of binary variables and error plots for MIQP problems with 60 vars, 60 ineq. constraints, 8 equality constraints, 50 binary constraints.

as

$$\min_{\theta} \|\theta\|_0 \text{ s.t. } A\theta = b, \tag{2.14}$$

where $A \in \mathbb{R}^{m\times n}$ with $m < n$, $b \in \mathbb{R}^m$, $\theta \in \mathbb{R}^n$ and $\|\cdot\|_0$ denotes the ℓ_0-seminorm, which gives the number of nonzero components of its argument. In case the entries of the matrix A or the components of the vector b are affected by noise, the equality constraint $A\theta = b$ can be relaxed and an error tolerance ε can be allowed in the equation $A\theta = b$.

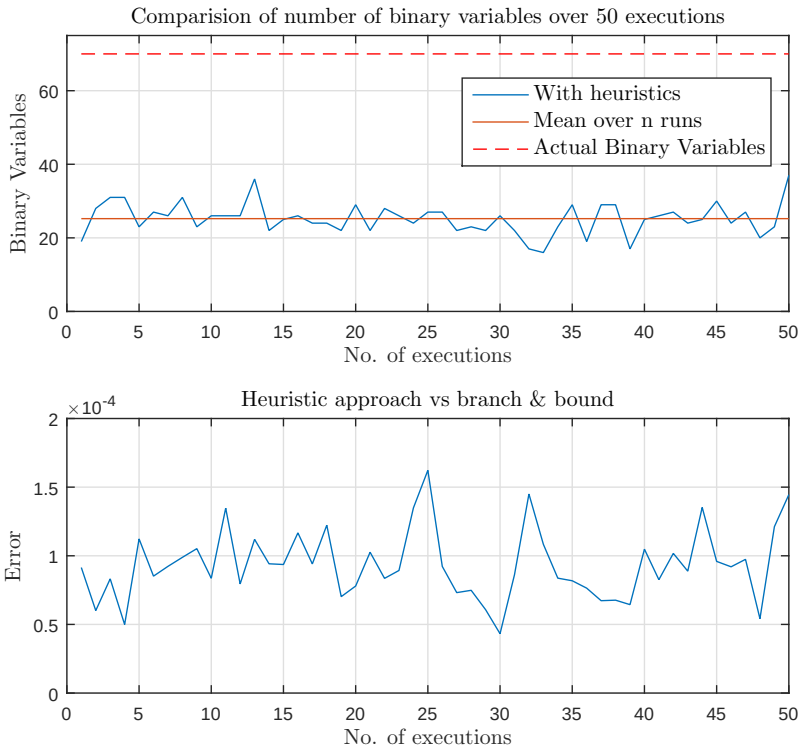


Figure 8: Number of binary variables and error plots for MIQP problems with 90 vars, 80 ineq. constraints, 8 equality constraints, 70 binary constraints.

This leads to the following variation of Problem (2.14):

$$\min_{\theta} \|\theta\|_0 \;\text{ s.t. }\; \|A\theta - b\|_2^2 \leq \varepsilon, \tag{2.15}$$

Problem (2.15) is often considered in system identification (Piga and Tóth, 2013).

Problem (2.15) can be also written in the *Lagrangian form*:

$$\min_{\theta} \quad \|A\theta - b\|_2^2 + \gamma\|\theta\|_0 \tag{2.16}$$

with $\gamma > 0$ is a tunning parameter. This means for a given value of ε, there exists a value of γ such that the minima of (2.15) and (2.16) are equal.

Unfortunately, Problems (2.14), (2.15) and (2.16) are NP-hard as well as non-convex and they are difficult to solve in practice for large values of n. One common approach is to compute the approximate solution by replacing the ℓ_0 norm with ℓ_1 norm, which leads to solving a convex optimization problem. However, exact solution may not be achieved with such relaxations.

To overcome this drawback and to obtain an exact solution of the non-convex Problem (2.16), we formulate it as an MIQP problem as follows:

$$\min_{\theta,\omega} \quad \|A\theta - b\|_2^2 + \gamma \textstyle\sum_{i=1}^{n} \omega_i \tag{2.17a}$$

$$\text{s.t.} \quad m\omega_i \leq \theta_i \leq M\omega_i \;\; i = 1, \ldots, n. \tag{2.17b}$$

where, we have introduced binary variables $\omega_i \in \{0, 1\}$ for $i = 1, \ldots, n$, M and m are user-specified tuning parameters which are used to define upper and lower bounds on the components θ_i. It can be easily shown that the Problem 2.16 and Problem 2.17 are equivalent[5].

We consider the optimization problem (2.14) from (Piga and Tóth, 2013) with :

$$A = \begin{bmatrix} 2 & -1 & 21 & 3 \\ 10/3 & 3.33 & 44.17 & 2.5 \\ -4/3 & 4.67 & -26.67 & -4 \end{bmatrix}; \;\; b = \begin{bmatrix} 6 \\ 10 \\ -4 \end{bmatrix}$$

The global minimizer θ^* of the original nonconvex problem (2.14) is equal to $\begin{bmatrix} 3 & 0 & 0 & 0 \end{bmatrix}'$, whereas with ℓ_1-norm $\begin{bmatrix} 0 & \frac{22}{99} & \frac{22}{99} & -\frac{22}{99} \end{bmatrix}'$ (Piga and Tóth, 2013). The heuristic approach miqpGPAD-H with $\epsilon_V, \epsilon_G = 10^{-5}$, $\epsilon_I = 10^{-5}$, regularization term $10^{-3}I$ to make the Hessian positive definite, for solving (2.17) with $\gamma = 0.2$, $M = 4$ gives, $\begin{bmatrix} 3 & 0 & 0 & 0 \end{bmatrix}'$.

[5]In the MIQP formulation, $\|\theta\|_0$ is replaced by a sum of *binary* variables $\sum_{i=1}^{n} \omega_i$, and the optimum $\{\theta^*, \omega^*\}$ is equal to that of Problem 2.16.

ARX Model Segmentation

The idea of using MIQP for sparse optimization using ℓ_0 norm can be applied to the ARX model segmentation problem.

The ARX models

$$\begin{aligned} &y(t) + a_1 y(t-1) + \ldots + a_{n_a} y(t-n_a) = \\ &b_1 u(t-n_k-1) + \ldots + b_{n_b} u(t-n_k-n_b) + e(t), \end{aligned} \tag{2.18}$$

can be represented in a linear regression form

$$y(t) = \varphi'(t)\theta + e(t) \tag{2.19}$$

with,

$\varphi(t) = [-y(t-1) \ \cdots \ -y(t-n_a) \ u(t-n_k-1) \ \cdots \ u(t-n_k-n_b)]$ is a regressor vector and $\theta = [a_1 \ \cdots \ a_{n_a} \ b_1 \ \cdots \ b_{n_b}]' \in \mathbb{R}^n$ is the vector of unknown parameters. A time-varying system (model) can be equivalently represented as

$$y(t) = \varphi'(t)\theta(t)$$

In the ARX model segmentation problem, a time-varying system parameters $\theta(t)$ are to be estimated from a given noise-corrupted $N-$ length dataset $\{y(t), u(t)\}_{t=1}^{N}$. When the system parameter are piecewise constants for a given time interval i.e.,

$$\theta(t) = \theta_k, \quad t_k < t \leq t_{k+1}$$

it is known as *model or signal segmentation*.

We consider the example from `iddemo6m.mat` in the System Identification Toolbox (Ljung, 1988; Ohlsson and Ljung, 2013), which consists of the system

$$y(t) + 0.9y(t-1) = u(t-n_k) + e(t) \tag{2.20}$$

The values input u are ± 1 Pseudo-Random Binary Sequence, $e(t)$ has variance 0.1. The value of transport delay n_k changes from 2 to 1 at $t = 20$. This demo uses `segment` method (Ljung, 1988) to estimate model parameters.

The following ARX model

$$\begin{aligned} y(t) &= -ay(t-1) + b_1 u(t-1) + b_2 u(t-2) \\ &= \varphi(t)'\theta(t) \end{aligned}$$

is used to estimate $\theta(t) = [a\ b_1\ b_2]'$, which equivalently represents the system (2.20).

The estimation problem is formulated as the following sparse-optimization problem with ℓ_0 norm.

$$\min_{\theta(t)} \quad \sum_{t=1}^{N} (y(t) - \varphi(t)'\theta(t))^2 + \gamma \sum_{t=2}^{N} \|\theta(t) - \theta(t-1)\|_0 \tag{2.21}$$

Here, we have added the regularization term $\|\theta(t)-\theta(t-1)\|_0$ to penalize model parameter change over time. The hyper-parameter γ can be tuned to achieve a trade-off between model fit and time variation of the model parameters. The problem (2.21) can be formulated as an equivalent MIQP problem (2.17).

The formulated MIQP problem consists of $n = 240$ variables out of which $p = 120$ are the binary variables, $q = 240$ inequality constraints, and value $M = -m = 1$ is considered. This problem is solved using the heuristic miqpGPAD-H, "mid-way" heuristic miqpGPAD-mH approaches. GUROBI with presolver enabled, thread count fixed to 1 (for fair comparison) is used to compare performance of the presented approaches. For both approaches, a regularization term $10^{-2}I$ is added to the Hessian to make it positive definite.

The results using miqpGPAD-H are summarized in Table 6, and the tolerance values used are $\epsilon_V = 10^{-5}, \epsilon_G = 10^{-1},\ \epsilon_I = 10^{-5}$.

For miqpGPAD-mH, experiments are carried out with varying values of $\epsilon_{\bar{\ell}} = 1 - \epsilon_{\bar{u}} = \epsilon$. The results are shown in Table 7, where this approach gives only p binary variables to be solved using B&B from the total of 120 for the same problem. The tolerance for the B&B algorithm miqpGPAD are $\epsilon_V, \epsilon_G = 10^{-5},\ \epsilon_I = 10^{-2}$.

The estimated parameter values $a,\ b_1$ and b_2 are compared with the true values and `segment` are shown in the Figure 9, 10 and 11 for GUROBI, miqpGPAD-H and miqpGPAD-mH respectively.

Table 6: Performance comparison for ARX model segmentation problem using heuristic approach without using B&B.

Solver	γ	Cost	Time (s)
GUROBI	0.5	46.17	1.78
miqpGPAD-H		47.15	0.41
GUROBI	0.7	46.97	0.39
miqpGPAD-H		47.95	0.78

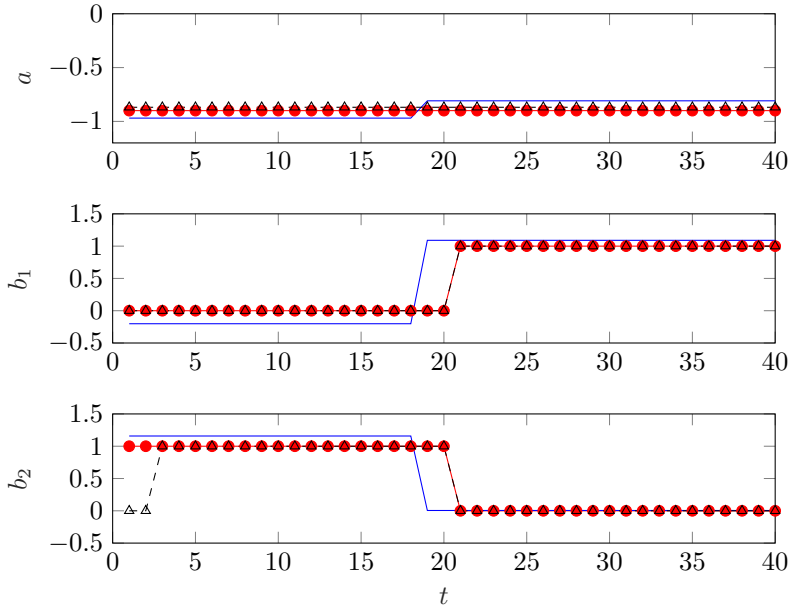


Figure 9: Performance comparison for ARX model segmentation problem `iddemo6m.mat` : true value, `segment`, - GUROBI.

These results demonstrate the effectiveness of proposed approaches. The heuristic approach gives a suboptimal integer feasible solution which is close to the optimal solution, within comparable computation time as compared to state-of-the-art commercial solver. Further, these results are shown to be improved by using the "mid-way" heuristic approach, which chooses significantly reduced number of binary variables

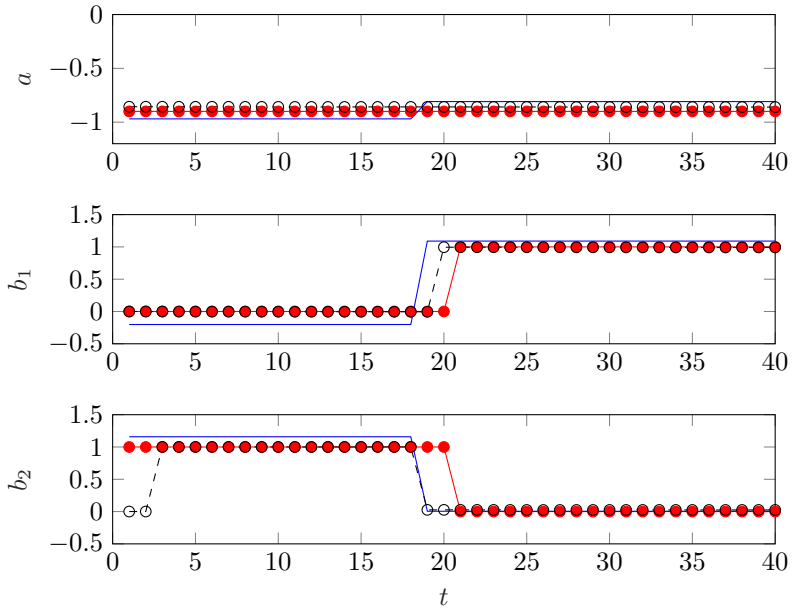


Figure 10: Performance comparison for ARX model segmentation problem `iddemo6m.mat` with $\gamma = 0.7$: —●—true value, ——`segment`, - ⊖ - miqpGPAD-H.

Table 7: Performance comparison for ARX model segmentation problem using the "mid-way" heuristic approach, reported timings are for heuristic and B&B stages respectively.

Solver	γ	ϵ	p	Cost	Time (s)
GUROBI	0.5	-	120	46.17	1.83
miqpGPAD-mH		0.3	5	46.39	0.41, 0.89
		0.2	8	46.17	0.41, 1.33
		0.1	12	46.17	0.41, 2.11
GUROBI	0.7	-	120	46.97	0.41
miqpGPAD-mH		0.3	5	47.19	0.78, 0.94
		0.2	7	46.97	0.78, 1.17
		0.1	10	46.97	0.78, 1.97

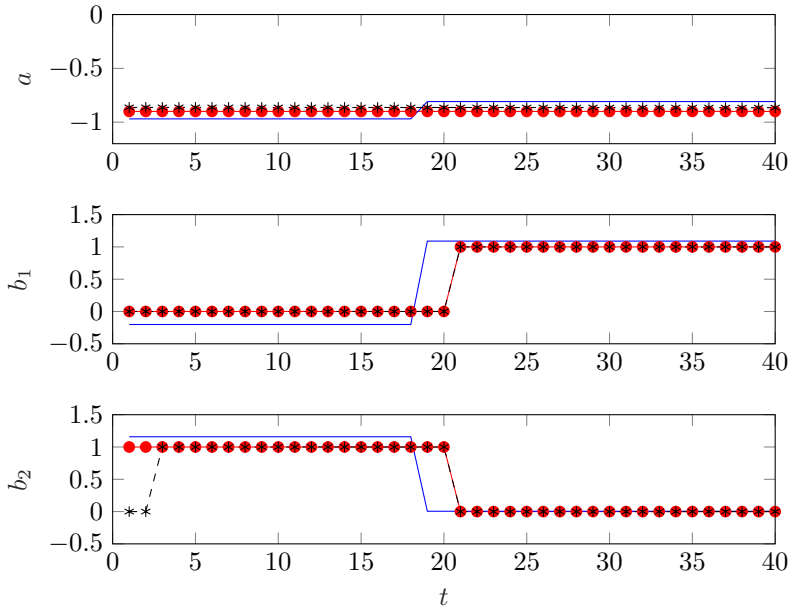


Figure 11: Performance comparison for ARX model segmentation problem `iddemo6m.mat` with $\gamma = 0.7$, $\epsilon = 0.2$: —●— true value, —— `segment`, - ∗ - miqpGPAD-mH.

to be solved using B&B, approximately 7% of total number in this case, and yet render a solution which is close to the optimal solution within the specified tolerance values.

2.7 Conclusion

In this chapter we have presented an exact and two heuristic approaches to solve MIQPs based on Accelerated Dual Gradient Projection (GPAD) method applied on the dual QP relaxations. Moreover, B&B based MIQP solving approaches with ADMM and OSQP have been presented, where OSQP based solver is robust and does not require the Hessian to be positive definite. Whereas, the GPAD based approach has an advantage of employing dual cost based pruning for premature termination of

QP solver. In spite of their simplicity of code, the proposed approaches turn out to be quite effective. In particular, the heuristic method can often provide solutions close to optimality with much reduced coding and computation efforts, which can further be used to provide upper bound on the cost. The performance of the proposed approaches is comparable with the state-of-the-art MIQP solvers for small-scale problems, such as those that arise in embedded applications.

Chapter 3

A Numerically Robust Mixed-Integer Quadratic Programming Solver based on Nonnegative Least Squares

3.1 Introduction

The algorithms for solving Mixed-Integer Quadratic Programming (MIQP) problems presented in Chapter 2 are very simple to code, and are demonstrated to be quite effective for small-scale problems. However, the Accelerated Dual Gradient Projection (GPAD) and Alternating Direction Method of Multipliers (ADMM) based MIQP solvers require a strictly convex objective function, with their numerical performance deteriorating with the condition number of the Hessian matrix. While regularizing the cost function would improve numerical robustness, it would bias the solution away from optimality.

In this chapter we propose a new algorithm for solving MIQP problems which is particularly tailored to solve small-scale MIQPs, such as

those that arise in embedded hybrid MPC applications. The algorithm couples a branch and bound (B&B) scheme with a recently proposed numerically robust Quadratic Programming (QP) solver based on an active-set method for solving *nonnegative least squares* (NNLS) problems combined with proximal-point iterations. The resulting MIQP solver supports positive semidefinite Hessian matrices, often appearing in hybrid MPC formulations, and warm-starts with respect to both binary and real variables.

3.1.1 Motivation

Since hybrid Model Predictive Control (MPC) was introduced almost two decades ago (Bemporad and Morari, 1999a), it has attracted a lot of attention in both academia and industry. The widespread popularity of hybrid systems is mainly due to their ability of modeling a very large spectrum of practical systems where physical processes coexist with switching dynamics, discrete actuators, logic rules, and constraints on system variables. MPC based on hybrid models provides an optimized way of controlling such systems. However, the real-time implementation of hybrid MPC on embedded platforms is still a challenge, as it requires solving an MIQP problem at every sample time in a numerically efficient and robust manner.

For this reason, several solution approaches to solve MIQP problems in an embedded control setting have been investigated, which have been reviewed in Section 1.4.1 and 1.4.2. In "desktop" applications the numerical package is used to solve (sometimes large-scale) MIQPs with large memory and computing resources available and no stringent limits on execution time. Conversely, in embedded applications, severe restrictions on CPU/memory/time resources pose considerable differences: the number of variables (especially binary variables) should be small, the code must be simple and library-free, the algorithm must be numerically robust also when executed in single precision arithmetic.

3.1.2 Contributions

In this chapter, we propose an MIQP algorithm based on B&B and the numerically robust solver for positive semidefinite QPs recently introduced in Bemporad (2018). The QP solver is based on an active-set method for solving nonnegative least-squares (NNLS) problems, but differently from (Bemporad, 2016) it uses proximal-point iterations, that greatly improve robustness without worsening execution time. Compared to the MIQP approach in (Bemporad, 2015b), the numerical robustness of the resulting MIQP solver is largely improved and can handle positive semidefinite Hessian matrices. Moreover, it handles equality constraints, supports warm-starting of QP relaxations from parent nodes in the search tree, and more general bilateral inequality constraints.

In addition, we include a warm-start strategy for binary variables within the B&B setup, which allows prioritizing the exploration of a specific section of the binary tree. This is especially useful in hybrid MPC (Bemporad and Morari, 1999a) and moving-horizon estimation (Bemporad et al., 1999a; Ferrari-Trecate et al., 2002) where a good initial guess for the binary variables is available by shifting the optimal solution computed at the previous sample step. Moreover, the heuristic approach proposed in Section 2.4.2 of Chapter 2 is also combined with the warm-start framework for binary variable as a special case.

We will show in numerical experiments that the resulting approaches gives quite comparable results against well-known commercial solvers when tested on small-scale MIQPs, such as those arising in embedded hybrid MPC applications.

3.1.3 Outline

This chapter is organized as follows. The problem formulation is presented in Section 3.2. A robust QP solver based on NNLS and proximal-point iterations described in Section 3.3. The branch and bound (B&B) algorithm is presented in Section 3.4. In addition, a generic framework to warm-start binary variables is proposed in Section 3.5, which is employed in combination with the B&B presented in the previous section.

Section 3.6 reports the numerical results of the proposed approach applied to a hybrid model predictive control (MPC) problem. The chapter is concluded with final remarks in Section 3.7.

3.2 Problem formulation

We recall the Mixed-Integer Quadratic Programming (MIQP) problem in the general form, which is

$$\min_{z} \quad V(z) \triangleq \frac{1}{2} z'Qz + c'z \tag{3.1a}$$
$$\text{s.t.} \quad \ell \leq Az \leq u \tag{3.1b}$$
$$Gz = g \tag{3.1c}$$
$$\bar{A}_i z \in \{\bar{\ell}_i, \bar{u}_i\},\ i = 1, \ldots, p \tag{3.1d}$$

where $z \in \mathbb{R}^n$ is the vector of optimization variables, $Q \in \mathbb{R}^{n \times n}$ is the positive semidefinite Hessian matrix, $Q \succeq 0$, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $\ell, u \in \mathbb{R}^m$ describe the linear inequality constraints, $\ell \leq u$, and $G \in \mathbb{R}^{q \times n}$, $g \in \mathbb{R}^q$ the linear equality constraints. The QP relaxation of problem (3.1) is obtained by replacing the integrality constraint (3.1d) with the following linear inequality constraint of the form (3.1b) as

$$\bar{\ell}_i \leq \bar{A}_i z \leq \bar{u}_i,\ i = 1, \ldots, p. \tag{3.2}$$

MIQP problems as in (3.1) arise in hybrid MPC. In the remainder of this section, we briefly recall the modelling and control of hybrid dynamical systems from Section 1.2.

Hybrid systems can efficiently be modeled using the following Mixed Logical Dynamical (MLD) framework (Bemporad and Morari, 1999a)

$$x_{k+1} = \mathcal{A}x_k + \mathcal{B}_1 v_k + \mathcal{B}_2 \delta_k + \mathcal{B}_3 \zeta_k + \mathcal{B}_5 \tag{3.3a}$$
$$y_k = \mathcal{C}x_k + \mathcal{D}_1 v_k + \mathcal{D}_2 \delta_k + \mathcal{D}_3 \zeta_k + \mathcal{D}_5 \tag{3.3b}$$
$$\mathcal{E}_2 \delta_k + \mathcal{E}_3 \zeta_k \leq \mathcal{E}_1 v_k + \mathcal{E}_4 x_k + \mathcal{E}_5, \tag{3.3c}$$

where $k \in \mathbb{N}$ is the time index, x_k is the state vector, y_k is the output vector, v_k is the input vector, ζ_k and δ_k are the vectors of auxiliary variables

which are real-valued and binary respectively. The vectors x_k, v_k, y_k can have both real and binary components, $\mathcal{A}$, $\mathcal{B}_i$, $\mathcal{C}$, $\mathcal{D}_i$ and $\mathcal{E}_i$ are the constant matrices describing the behavior of hybrid dynamical systems. The tool HYSDEL (Torrisi and Bemporad, 2004) allows one to describe a hybrid dynamical model and get the equivalent MLD transformation (3.3).

Based on the MLD model (3.3), a possible hybrid MPC problem formulation is the following

$$\min_{\{v_k,\delta_k,\zeta_k\}_{k=0}^{N-1}} \sum_{k=0}^{N-1} \quad \|L_x(x_k - r_k^x)\|_2^2 + \|L_v(v_k - r_k^v)\|_2^2 + \quad (3.4a)$$

$$\|L_\delta(\delta_k - r_k^\delta)\|_2^2 + \|L_\zeta(\zeta_k - r_k^\zeta)\|_2^2$$

$$\text{s.t.} \quad \text{MLD model (3.3)} \quad (3.4b)$$

$$x_0 = x(t).$$

Problem (3.4) can be recast as an MIQP problem of the form (3.1) by defining the n-dimensional optimization vector

$$z = \left[v_0' \ \cdots \ v_{N-1}' \ \delta_0' \ \cdots \ \delta_{N-1}' \ \zeta_0' \ \cdots \ \zeta_{N-1}' \right]'.$$

Very often the MIQP problem (3.1) derived from the hybrid MPC formulation (3.4) has a positive semidefinite Hessian matrix Q, depending on the weight matrices L_x, L_v, L_δ, L_ζ. The MIQP solution approach described in this chapter does not require regularizing Q to make the problem strictly convex.

3.3 Solution of QP relaxations

Solving problem (3.1a)–(3.1d) using B&B requires solving QP relaxations, in which constraints (3.1d) are either relaxed to inequality constraints $\bar{\ell}_i \le \bar{A}_i z \le \bar{u}_i$ or to equality constraints $A_i z = \bar{\ell}_i$ or $A_i z = \bar{u}_i$.

To solve QP relaxations of the form (3.1a)–(3.1c), we slightly extend the solver in (Bemporad, 2018) to handle bilateral constraints of the form (3.1b), as summarized in Algorithm 8. It is based on the idea of solving proximal-point iterations, where each iteration consists of solving a regularized QP. The latter that is recast as a least distance problem

(LDP) and solved using a nonnegative least squares (NNLS) algorithm. We describe the steps of Algorithm 8 in the following sections.

3.3.1 Outer proximal-point iterations

Let $Q = Q' \succeq 0$ in (3.1a)–(3.1c). For any $\epsilon > 0$, the sequence $\{z_k\}$ of solutions to the following strictly convex quadratic programs

$$\begin{array}{rl} z_{k+1} = \arg\min_z & \frac{1}{2} z'Qz + c'z + \frac{\epsilon}{2}\|z - z_k\|_2^2 \\ \text{s.t.} & \ell \leq Az \leq u \\ & Gz = g \end{array} \tag{3.5}$$

converges to a solution z^* of the QP relaxation (3.1a)–(3.1c) as k tends to infinity (Bemporad, 2018, Corollary 1).

3.3.2 Inner active-set solver

The dual of the QP problem (3.5) is the following convex QP

$$\min_{\lambda_\ell, \lambda_u, \mu} \quad \frac{1}{2} \begin{bmatrix} \lambda_\ell \\ \lambda_u \\ \mu \end{bmatrix}' \begin{bmatrix} -A \\ A \\ G \end{bmatrix} (Q + \epsilon I)^{-1} \begin{bmatrix} -A \\ A \\ G \end{bmatrix}' \begin{bmatrix} \lambda_\ell \\ \lambda_u \\ \mu \end{bmatrix} + \begin{bmatrix} d_\ell^k \\ d_u^k \\ f^k \end{bmatrix}' \begin{bmatrix} \lambda_\ell \\ \lambda_u \\ \mu \end{bmatrix} \tag{3.6a}$$

$$\text{s.t.} \quad \lambda_\ell, \lambda_u \geq 0, \ \mu \text{ free}, \tag{3.6b}$$

where $\lambda_\ell, \lambda_u \in \mathbb{R}^m$, $\mu \in \mathbb{R}^q$, and $d_\ell^k, d_u^k \in \mathbb{R}^m$, $f^k \in \mathbb{R}^q$ are defined as follows:

$$\begin{bmatrix} d_\ell^k \\ d_u^k \\ f^k \end{bmatrix} \triangleq \begin{bmatrix} -\ell \\ u \\ g \end{bmatrix} + \begin{bmatrix} -A \\ A \\ G \end{bmatrix} (Q + \epsilon I)^{-1}(c - \epsilon z_k). \tag{3.6c}$$

The following theorem shows how the QP problem (3.5) is equivalent to a least-squares problem in which some of the variables are constrained to be nonnegative, extending (Bemporad, 2016, Th. 1) to the case of equality constraints, bilateral inequalities as in (Bemporad, 2015b) and subsequently to scaling of d_ℓ^k, d_u^k, f^k for improved numerical robustness as in (Bemporad, 2018).

Theorem 1 *Consider the QP (3.5) and let $Q \succeq 0$, $\epsilon > 0$. Let $L'L$ be a Cholesky factorization of $Q + \epsilon I$ and define $M \triangleq AL^{-1}$, $N \triangleq GL^{-1}$. Let γ, β be any*

positive scalars. Consider the Partially Nonnegative Least Squares (PNNLS) problem

$$\min_{y_\ell, y_u, \nu} \frac{1}{2} \left\| \begin{bmatrix} -M' \\ \beta(d_\ell^k)' \end{bmatrix} y_\ell + \begin{bmatrix} M' \\ \beta(d_u^k)' \end{bmatrix} y_u + \begin{bmatrix} N' \\ \beta(f^k)' \end{bmatrix} \nu + \begin{bmatrix} 0 \\ \beta\gamma \end{bmatrix} \right\|_2^2 \tag{3.7a}$$

$$\text{s.t.} \quad y_\ell, y_u \geq 0, \ \nu \text{ free} \tag{3.7b}$$

with $y_\ell, y_u \in \mathbb{R}^m$, $\nu \in \mathbb{R}^q$, *and let*

$$\delta^* \triangleq \beta(\gamma + (d_\ell^k)' y_\ell^* + (d_u^k)' y_u^* + (f^k)' \nu^*) \tag{3.8a}$$

$$r^* \triangleq \begin{bmatrix} M'(y_\ell^* - y_u^*) - N'\nu^* \\ -\delta^* \end{bmatrix} \tag{3.8b}$$

where $r^* \in \mathbb{R}^{n+1}$ *is the residual obtained at the optimal solution* (y_ℓ^*, y_u^*, ν^*) *of* (3.7), *where* $y_\ell^*, y_u^* \in \mathbb{R}^m$, $\nu^* \in \mathbb{R}^q$. *The following statements hold:*

i) If $r^* = 0$ *then QP* (3.1a)–(3.1c) *is infeasible;*

ii) If $r^* \neq 0$ *then*

$$z^* \triangleq -(Q + \epsilon I)^{-1} \left(c - \epsilon z_k + \frac{A'(y_u^* - y_\ell^*) + G'\nu^*}{\beta\delta^*} \right) \tag{3.9}$$

and

$$\lambda_\ell^* \triangleq \frac{1}{\beta\delta^*} y_\ell^*, \ \lambda_u^* \triangleq \frac{1}{\beta\delta^*} y_u^*, \ \mu^* \triangleq \frac{1}{\beta\delta^*} \nu^*$$

solve QP (3.5) *and its dual* (3.6), *respectively.*

Proof. See Appendix A. ■

A robust QP solver based on NNLS and proximal-point iterations is described in Algorithm 8. Steps 1–4 performs initialization, followed by active-set iterations starting at Step 5. The active sets $\mathcal{P}_u, \mathcal{P}_\ell$ change for every iteration while solving problem (3.7) at a given proximal-point iteration k; this is done via rank-one updates of LDL′ factorization (Bemporad, 2018, Section III. C). Feasible values as in (3.10) are computed at Steps 7, 8. Steps 9, 10 check for infeasibility. The inner active-set iterations continue from Step 5 until w is feasible (within tolerance) at Step 11 or if all inequality constraints have entered the active set, which is followed by computation of solution z_k at Steps 14, 15. Step 16 continues executing proximal-point iterations until the criteria (3.12) is satisfied followed by computing the optimal solution at Step 17.

Algorithm 8 Robust QP solver based on NNLS and proximal-point iterations

Input: QP matrices Q, c, A, G, vectors u, ℓ, g, regularization term $\epsilon > 0$, feasibility tolerance $\sigma > 0$, stop tolerance $\eta > 0$, initial guess z_0.

1. Compute inverse Cholesky factor $\mathcal{L}^{-1}$, $\mathcal{L}'\mathcal{L} = (Q + \epsilon I)$;
2. $M \leftarrow A\mathcal{L}^{-1}$, $N \leftarrow G\mathcal{L}^{-1}$; $\mathcal{P}_\ell, \mathcal{P}_u \leftarrow \emptyset$;
3. Factorize NN' as $LDL' = NN'$;
4. $k, h \leftarrow 0$;
5. $v_k \leftarrow \mathcal{L}^{-T}(c - \epsilon z_k)$; $W_{\mathcal{P}} \leftarrow \begin{bmatrix} -M_{\mathcal{P}_\ell} \\ M_{\mathcal{P}_u} \\ N \end{bmatrix}$; $d_\ell^k \leftarrow -(\ell + Mv_k)$; $d_u^k \leftarrow u + Mv_k$; $\theta_{\mathcal{P}}^k \leftarrow \begin{bmatrix} d_{\ell\mathcal{P}_\ell}^k \\ d_{u\mathcal{P}_u}^k \\ g + Nv_k \end{bmatrix}$; $\gamma \leftarrow \|\theta_{\mathcal{P}}^k\|_1$; $\beta \leftarrow \frac{1}{\gamma}$;
6. $[L, D] \leftarrow$ *rankone*$(L, D, 1, \beta\theta_{\mathcal{P}}^k)$;
7. $(y_\ell, y_u, \nu, \mathcal{P}_\ell, \mathcal{P}_u, \gamma, \beta) \leftarrow$ *get_feasible*$(\mathcal{P}_\ell, \mathcal{P}_u)$; $h \leftarrow h + 1$;
8. $\delta \leftarrow \beta(\gamma + (\theta_{\mathcal{P}}^k)' \begin{bmatrix} y_{\ell\mathcal{P}_\ell} \\ y_{u\mathcal{P}_u} \\ \nu \end{bmatrix})$, $a \leftarrow W_{\mathcal{P}}' \begin{bmatrix} y_{\ell\mathcal{P}_\ell} \\ y_{u\mathcal{P}_u} \\ \nu \end{bmatrix}$, $\begin{bmatrix} w_\ell \\ w_u \end{bmatrix} \leftarrow Ma \begin{bmatrix} -I \\ I \end{bmatrix} + \beta\delta \begin{bmatrix} d_\ell^k \\ d_u^k \end{bmatrix}$;
9. $\rho \leftarrow a'a + \delta^2$;
10. **if** $\rho = 0$ **then** QP problem (3.1) is infeasible; **go to** Step 18;
11. **if** $\min\{w_{\ell_i}, w_{u_i}\} \geq -\beta\delta\sigma, \forall i \in \{1, \ldots, m\}$, **or** $\mathcal{P}_\ell \cup \mathcal{P}_u = \{1, \ldots, m\}$ **then go to** Step 14;
12. $i_\ell \leftarrow \arg\min_{i \in \{1,\ldots,m\} \setminus \mathcal{P}_\ell} w_{\ell i}$, $i_u \leftarrow \arg\min_{i \in \{1,\ldots,m\} \setminus \mathcal{P}_u} w_{ui}$, **if** $w_{\ell i_\ell} \leq w_{ui_u}$ **then** $\mathcal{P}_\ell \leftarrow \mathcal{P}_\ell \cup \{i_\ell\}$; $\gamma \leftarrow \gamma + |d_{\ell i_\ell}|$; **otherwise** $\mathcal{P}_u \leftarrow \mathcal{P}_u \cup \{i_u\}$; $\gamma \leftarrow \gamma + |d_{ui_u}|$; $\bar{\beta} = \beta$, $\beta = \frac{1}{\gamma}$;
13. update LDL′ factorization **go to** Step 7;
14. $k \leftarrow k + 1$;
15. $\begin{bmatrix} \lambda_{\ell k} \\ \lambda_{uk} \\ \mu_k \end{bmatrix} \leftarrow \frac{1}{\delta} \begin{bmatrix} y_\ell \\ y_u \\ \nu \end{bmatrix}$; $u_k \leftarrow \frac{1}{\delta}a$; $z_k \leftarrow \mathcal{L}^{-1}(u_k - v_k)$;
16. **if** $\|z_k - z_{k-1}\| > \eta$ **then** $[L, D] \leftarrow$ *rankone*$(L, D, -1, \beta d_{\mathcal{P}}^k)$; **go to** Step 5;
17. $z^* \leftarrow z_k$, $\lambda_\ell^* \leftarrow \lambda_{\ell_k}$, $\lambda_u^* \leftarrow \lambda_{u_k}$, $\mu^* \leftarrow \mu_k$, $\mathcal{P}_\ell^* \leftarrow \mathcal{P}_\ell$, $\mathcal{P}_u^* \leftarrow \mathcal{P}_u$;
18. **end**.

19. **procedure** $(y_\ell, y_u, \nu, \mathcal{P}_\ell, \mathcal{P}_u, \gamma, \beta) \leftarrow$ *get_feasible*$(\mathcal{P}_\ell, \mathcal{P}_u)$

19.1. solve the LS problem $\begin{bmatrix} s_{\ell\mathcal{P}_\ell} \\ s_{u\mathcal{P}_u} \\ s_\nu \end{bmatrix} \leftarrow \arg\min_z \left\| \begin{bmatrix} W'_{\mathcal{P}} \\ \beta(\theta^k_{\mathcal{P}})' \end{bmatrix} z + \begin{bmatrix} 0_n \\ \beta\gamma \end{bmatrix} \right\|_2^2$; $s_{\ell\{1,\ldots,m\}\setminus\mathcal{P}_\ell} \leftarrow 0, s_{u\{1,\ldots,m\}\setminus\mathcal{P}_u} \leftarrow 0$;

19.2. **if** $s_{\ell\mathcal{P}_\ell}, s_{u\mathcal{P}_u} \geq 0$ **then** $\begin{bmatrix} y_\ell \\ y_u \\ \nu \end{bmatrix} \leftarrow \begin{bmatrix} s_\ell \\ s_u \\ s_\nu \end{bmatrix}$ and **go to** Step 19.8;

19.3. $j_\ell \leftarrow \arg\min_{h\in\mathcal{P}_\ell:\, s_{\ell h}\leq 0} \left\{ \frac{y_{\ell h}}{y_{\ell h} - s_{\ell h}} \right\}$,
$j_u \leftarrow \arg\min_{h\in\mathcal{P}_u:\, s_{uh}\leq 0} \left\{ \frac{y_{uh}}{y_{uh} - s_{uh}} \right\}$;

19.4. $\begin{bmatrix} y_\ell \\ y_u \\ \nu \end{bmatrix} \leftarrow \begin{bmatrix} y_\ell \\ y_u \\ \nu \end{bmatrix} + \min\{j_\ell, j_u\} \cdot \left(\begin{bmatrix} s_\ell \\ s_u \\ s_\nu \end{bmatrix} - \begin{bmatrix} y_\ell \\ y_u \\ \nu \end{bmatrix} \right)$;

19.5. $\mathcal{F}_\ell \leftarrow \{h \in \mathcal{P}_\ell : y_{\ell h} = 0\}, \mathcal{P}_\ell \leftarrow \mathcal{P}_\ell \setminus \mathcal{F}_\ell$;
$\gamma \leftarrow \gamma - \|d_{\ell\mathcal{F}_\ell}\|_1$;
$\mathcal{F}_u \leftarrow \{h \in \mathcal{P}_u : y_{uh} = 0\}; \mathcal{P}_u \leftarrow \mathcal{P}_u \setminus \mathcal{F}_u$;
$\gamma \leftarrow \gamma - \|d_{u\mathcal{F}_u}\|_1; \bar{\beta} = \beta, \beta = \frac{1}{\gamma}$;

19.6. update LDL′ factorization

19.7. **go to** Step 19.1;

19.8. **end procedure**.

Output: Primal solution z^* solving (3.1a)–(3.1c), (3.2) dual solution $(\lambda^*_\ell, \lambda^*_u, \mu^*)$, optimal active sets $\mathcal{P}^*_\ell$, $\mathcal{P}^*_u$ or infeasibility status.

3.3.3 Warm-starting

At each proximal-point iteration, as well as when starting to solve a new QP relaxation after a solution is available from the parent node in the B&B search tree, the QP problem (3.5) can be warm-started from an initial guess z_k and the corresponding problem (3.7) from sets of active constraints $\mathcal{P}_u, \mathcal{P}_\ell \subseteq \{1, \ldots, m\}$, $\mathcal{P}_u \cap \mathcal{P}_\ell = \emptyset$, along with the initial guess $y_\ell \geq 0, y_u \geq 0, w_\ell, w_u \in \mathbb{R}^m$ and $\nu \in \mathbb{R}^q$ which satisfies the following conditions:

$$w_\ell = -M(M'(y_u - y_\ell) + N'\nu) + \beta\delta d^k_\ell \tag{3.10a}$$
$$w_u = M(M'(y_u - y_\ell) + N'\nu) + \beta\delta d^k_u \tag{3.10b}$$
$$y'_\ell w_\ell = y'_u w_u = 0 \tag{3.10c}$$

$$y_{\ell i} \geq 0,\ w_{\ell i} = 0,\ \forall i \in \mathcal{P}_\ell \tag{3.10d}$$
$$y_{ui} \geq 0,\ w_{ui} = 0,\ \forall i \in \mathcal{P}_u \tag{3.10e}$$
$$y_{\ell i} = 0,\ \forall i \in \{1, \dots, m\} \setminus \mathcal{P}_\ell \tag{3.10f}$$
$$y_{ui} = 0,\ \forall i \in \{1, \dots, m\} \setminus \mathcal{P}_u \tag{3.10g}$$
$$\nu = -\begin{bmatrix} N' \\ \beta(f^k)' \end{bmatrix}^{\#} \begin{bmatrix} M'(y_u - y_\ell) \\ \delta \end{bmatrix} \tag{3.10h}$$

where $\#$ denotes pseudoinverse, and

$$\delta = \beta(\gamma + (d_\ell^k)' y_\ell + (d_u^k)' y_u + (f^k)'\nu). \tag{3.10i}$$

The terms (3.10a)–(3.10g), (3.10i) are derived from the Karush-Kuhn-Tucker (KKT) conditions of problem (3.7). See the proof of Theorem 1 in Appendix A for a justification of conditions (3.10).

3.3.4 Parameter selection and scaling

The parameters $\beta, \gamma > 0$ are selected as in (Bemporad, 2018)

$$\gamma = 1 + \|f^k\|_1 + \|d^k_{\ell\mathcal{P}_\ell}\|_1 + \|d^k_{u\mathcal{P}_u}\|_1,\ \beta = \frac{1}{\gamma},$$

which are known to provide good numerical conditioning.

In order to ensure better numerical robustness of the overall MIQP solver, one may scale the constraints of type (3.1b), (3.1c) as

$$\begin{aligned} &\hat{M}_i \triangleq \frac{1}{\|M_i\|_2},\ \hat{N}_i \triangleq \frac{1}{\|N_i\|_2} \\ &\hat{M}_i \ell_i \leq \hat{M}_i A_i z \leq \hat{M}_i u_i \\ &\hat{N}_i G_i z = \hat{N}_i g_i. \end{aligned}$$

Unlike first-order methods which are well-known to be very sensitive to scaling, our numerical experiments suggest that this algorithm is less sensitive to the scaling.

3.3.5 Stopping criteria and optimality

At Step 16 the iterates of Algorithm 8 stop when

$$\|z_k - z_{k-1}\| \leq \eta.$$

This condition corresponds to satisfying the optimality condition

$$\|Qz_k + c + A'(\lambda_{u_k} - \lambda_{\ell_k}) + G'\mu_k\| \leq \epsilon\eta \tag{3.12}$$

of the QP problem of the form (3.1a)–(3.1c). In fact, from (3.9), we have

$$\begin{aligned}(Q + \epsilon I)z_k &= -c + \epsilon z_{k-1} - A'(\lambda_{u_k} - \lambda_{\ell_k}) - G'\mu_k \\ \epsilon(z_k - z_{k-1}) &= -Qz_k - c - A'(\lambda_{u_k} - \lambda_{\ell_k}) - G'\mu_k\end{aligned}$$

and hence $\|z_k - z_{k-1}\| \leq \eta$ implies (3.12).

3.4 MIQP solver

Algorithm 9 describes a standard B&B method to solve the MIQP problem (3.1), which is adapted from Chapter 2 and is essential for utilizing the warm-starting binary variables framework proposed in the subsequent section. It uses Algorithm 8 for solving the QP relaxations of type (3.1a)-(3.1c).

The B&B algorithm consists of three basic sets $\mathcal{J}, \mathcal{I}_{\bar{\ell}}, \mathcal{I}_{\bar{u}}$ that store the indices of the $i = 1, \ldots, p$ binary constraints throughout the tree exploration within B&B. The set $\mathcal{J}$ contains the indices of the relaxed binary constraints of the form (3.2), $\ell_i \leq \bar{A}_i z \leq u_i$. The sets $\mathcal{I}_{\bar{\ell}}$, $\mathcal{I}_{\bar{u}}$ correspond to the indices of the binary constraints fixed as either $\bar{A}_i z = \bar{\ell}_i$ or $\bar{A}_i z = \bar{u}_i$, respectively. The interdependency of these sets is given by $\mathcal{J} = \{1, \ldots, p\} \setminus (\mathcal{I}_{\bar{\ell}} \cup \mathcal{I}_{\bar{u}})$, $\mathcal{I}_{\bar{\ell}} \cap \mathcal{I}_{\bar{u}} = \emptyset$. The tuple $\mathcal{T}$ holds the sets $\mathcal{I}_{\bar{\ell}}$, $\mathcal{I}_{\bar{u}}$ for each relaxed QP subproblem. The stack $\mathcal{S}$ holds these tuples with the order of remaining QP subproblems to be solved. The best cost associated with an integer feasible solution is denoted by V_0, z^* denotes the optimal value of the MIQP problem. Vector z_k is the initial value of the optimization vector, which is initialized with an arbitrary initial

Algorithm 9 MIQP solver based on NNLS

Input: MIQP problem matrices $Q = Q' \succeq 0$, c, A, $\bar{A}$, G and vectors ℓ, u, g, $\bar{\ell}$, $\bar{u}$, initial value z_0; feasibility tolerance $\epsilon \geq 0$.

1. **set** $V_0 \leftarrow +\infty$; $z^* \leftarrow \emptyset$; $\mathcal{J} \leftarrow \{1, \ldots, p\}$; $\mathcal{I}_{\bar{\ell}} \leftarrow \emptyset$;$\mathcal{I}_{\bar{u}} \leftarrow \emptyset$; $z_k \leftarrow z_0$; $\mathcal{T} \leftarrow \{\mathcal{I}_{\bar{\ell}}, \mathcal{I}_{\bar{u}}, z_k\}$; $\mathcal{S} \leftarrow \{\mathcal{T}\}$;
2. **while** $\mathcal{S} \neq \emptyset$ **do**:
   - 2.1. $\{\mathcal{I}_{\bar{\ell}}, \mathcal{I}_{\bar{u}}, z_k\} \leftarrow$ last element $\mathcal{T}$ of $\mathcal{S}$; $\mathcal{S} \leftarrow \mathcal{S} \setminus \{\mathcal{T}\}$;
   - 2.2. **execute** Algorithm 8 to solve QP problem for z^*,V^*;
   - 2.3. **if** the QP problem is feasible within tolerance ϵ**and** $V^* \leq V_0$ **then**
     - 2.3.1. **if** $\bar{A}_i z^* \in \{\bar{\ell}_i, \bar{u}_i\}$, $\forall i \in \mathcal{J}$ **or** $\mathcal{I}_{\bar{\ell}} \cup \mathcal{I}_{\bar{u}} = \{1, \ldots, p\}$ **then** $V_0 \leftarrow V^*$, $\xi^* \leftarrow z^*$; **otherwise**
       - 2.2.3.1. $j \leftarrow \arg\min_{i \in \mathcal{J}} \left| \bar{A}_i z^* - \frac{\bar{\ell}_i + \bar{u}_i}{2} \right|$;
       - 2.2.3.2. $\mathcal{J} \leftarrow \mathcal{J} \setminus j$, $\mathcal{T}_0 \leftarrow \{\mathcal{I}_{\bar{\ell}} \cup \{j\}, \mathcal{I}_{\bar{u}}, z^*\}$; $\mathcal{T}_1 \leftarrow \{\mathcal{I}_{\bar{\ell}}, \mathcal{I}_{\bar{u}} \cup \{j\}, z^*\}$;
       - 2.2.3.3. **if** $\bar{A}_j z^* \leq \frac{\bar{\ell}_j + \bar{u}_j}{2}$ **then** append $\{\mathcal{T}_1, \mathcal{T}_0\}$ to $\mathcal{S}$ **otherwise** append $\{\mathcal{T}_0, \mathcal{T}_1\}$ to $\mathcal{S}$;
3. **if** $V_0 = +\infty$ **then** (3.1) infeasible **otherwise** $\mathcal{V}^* \leftarrow V_0$;
4. **end**.

Output: Solution ξ^* of the MIQP problem (3.1), optimal cost $\mathcal{V}^*$, or infeasibility status.

guess z_0. At Step 1, the set $\mathcal{J}$ is initialized with all p binary constraints $\{1, \ldots, p\}$, which is same as (3.2), where all constraints (3.1d) are relaxed. This denotes the root-node of the B&B tree.

At Step 2.1, the last tuple $\mathcal{T}$ is popped from the stack $\mathcal{S}$ which corresponds to the relaxed QP subproblem to be solved. Step 2.2 solves this QP relaxation using Algorithm 8.

Step 2.3 checks if the solution of the QP problem at the previous step

is feasible and the solution $V^* \leq V_0$. Step 2.3.1 checks if all the binary constraints are satisfied and updates the best known integer feasible solution V_0 with the optimal value V^*. The binary constraint to be branched upon j is picked from the set $\mathcal{J}$ having the largest fractional part at Step 2.2.3.1. At Step 2.2.3.1, the index j is removed from the set $\mathcal{J}$, and two tuples $\mathcal{T}_0$, $\mathcal{T}_1$ are created corresponding to the two subproblems with j-th binary constraint equal to $\bar{\ell}_j$ and $\bar{u}_j$ respectively. In case of binary constraints that derive from imposing binary variables $z_j \in \{0,1\}$, z_j^* is replaced by either 0 or 1 in Step 2.2.3.1 for warm-start. Step 2.2.3.3 decides the priority among the two subproblems $\mathcal{T}_0$, $\mathcal{T}_1$ based on the value of $\bar{A}_j z^*$ of the relaxed solution.

Once all QP problems are solved, the value of V_0 is checked at Step 3. If V_0 is still having $+\infty$ then (3.1) is infeasible, in the other case the optimal cost $\mathcal{V}^*$ and the optimal solution ξ^* is returned.

3.5 Warm-starting binary variables

For simplicity we focus on the case of constraints (3.1d) having the form $z_i \in \{0,1\}$ for some indices $i \in \mathcal{G}$, $\mathcal{G} \subseteq \{1,\dots,n\}$, with $card(\mathcal{G}) = p$. We want to introduce a strategy in the MIQP solver that exploits warm-starts on (all or some of) the binary variables z_i. The strategy can be immediately extended to more general constraints of the form (3.1d).

Let $\tilde{\mathcal{I}}_{\bar{\ell}}, \tilde{\mathcal{I}}_{\bar{u}} \subseteq \mathcal{G}$ be the sets containing the indices of warm-started binary variables set equal to 0 or 1 respectively, $\tilde{\mathcal{I}}_{\bar{\ell}} \cap \tilde{\mathcal{I}}_{\bar{u}} = \emptyset$, $\tilde{\mathcal{I}}_{\bar{\ell}} \cup \tilde{\mathcal{I}}_{\bar{u}} \neq \emptyset$. After solving the relaxed problem (3.1a)-(3.1c), (3.2) at the root node to compute the optimal solution z^* (steps up to Step 2.2 of Algorithm 9), and assuming that $z_i^* \notin \{0,1\}$ for some $i \in \mathcal{G}$, Steps 2.2.3.1-2.2.3.3 of Algorithm 9 are replaced by Algorithm 10, which is described below.

As we are just below the root node in the tree, at Step 1 of Algorithm 10 with $\mathcal{J} = \{1,\dots,p\}$, if the any of the warm-started binary variables have fractional value then Steps 1.1–1.3 are executed. In this case, the priority for branching is first given to these warm-started variables at Step 1.1. Specifically at Step 1.1, the variable with smallest index j which is either contained in $\tilde{\mathcal{I}}_{\bar{\ell}}$ or $\tilde{\mathcal{I}}_{\bar{u}}$ is selected for branching. The selected j is

removed from set $\mathcal{J}$, and two new subproblems are created at Step 1.2. These subproblems are pushed on $\mathcal{S}$ with the order defined in Step 1.3[1], which induces solving first the relaxations in which the binary values are assigned as specified by warm-start. At this step, $card(\mathcal{J}) = p - 1$, and hence Step 2 below will be executed.

At Step 2.1, all the warm-started values are removed from set $\mathcal{J}$. Step 2.2 picks up the binary variable j for branching having the largest fractional part (z_j will be a variable that is not warm-started). At Step 2.3, two new subproblems are created with the j-th variable set either to 0 or 1, along with the binary variables belonging to sets $\tilde{\mathcal{I}}_{\bar{\ell}}$ or $\tilde{\mathcal{I}}_{\bar{u}}$ fixed equal to 0 or 1, respectively, which now lead to a specific section of the tree. These two subproblems are pushed on the top of $\mathcal{S}$ as in Step 2.4. Note that the procedure described above from Step 2.1 to Step 2.4 if executed, then for only once, due to the condition in Step 2.

According to a last-in-first-out strategy, these two problems are solved before exploring nodes below the root node which were formerly pushed on $\mathcal{S}$. This step will force the tree exploration from these two new subproblems, and further branching will be done only if necessary, until leaf nodes are reached. Specifically, if further branching is needed, due to the fact that at Step 2.1, $\mathcal{J} \leftarrow \{1, \ldots, p\} \setminus (\tilde{\mathcal{I}}_{\bar{\ell}} \cup \tilde{\mathcal{I}}_{\bar{u}})$, this is done by following the Steps 1.4–1.6, until leaf nodes are reached as per warm-start provided. Note that the binary variables $z_j \in \{0, 1\}$, z_j^* is replaced by either 0 or 1 in Steps 1.2, 1.5 and 2.3 for warm-start. If the resulting leaves are feasible then the value of V_0 (the best known integer-feasible solution) is updated appropriately with V^*. In this case, finite value of V_0 upfront can reduce the number of QP subproblems solved in the course of finding the global solution of MIQP problem using the proposed B&B algorithm.

This step is followed by convention of popping one of the QP subproblems which was pushed on the stack after solving the relaxation at root node [1], and corresponding QP subproblem is solved. Here, the branching on the binary variables below the root node is carried out us-

[1] Having solved the relaxation at the root node, the two problems are pushed on stack $\mathcal{S}$. This is done before warm-starting the binary variables.

Algorithm 10 warm-start for binary variables

Input: Sets $\tilde{\mathcal{I}}_{\bar{\ell}}$, $\tilde{\mathcal{I}}_{\bar{u}}$ containing the values corresponding to the warm-start, $\mathcal{J}$, $\mathcal{I}_{\bar{\ell}}$, $\mathcal{I}_{\bar{u}}$; Optimal solution z^* of the QP problem;

1. **if** $\mathcal{J} \cap (\tilde{\mathcal{I}}_{\bar{\ell}} \cup \tilde{\mathcal{I}}_{\bar{u}}) \neq \emptyset$ **then** (first branch on the warm-started binary variables)

   1.1. $j \leftarrow \inf(\mathcal{J} \cap (\tilde{\mathcal{I}}_{\bar{\ell}} \cup \tilde{\mathcal{I}}_{\bar{u}}))$;

   1.2. $\mathcal{J} \leftarrow \mathcal{J} \setminus j$, $\mathcal{T}_0 \leftarrow \{\mathcal{I}_{\bar{\ell}} \cup \{j\}, \mathcal{I}_{\bar{u}}, z^*\}$; $\mathcal{T}_1 \leftarrow \{\mathcal{I}_{\bar{\ell}}, \mathcal{I}_{\bar{u}} \cup \{j\}, z^*\}$;

   1.3. **if** $j \in \tilde{\mathcal{I}}_{\bar{\ell}}$ **then** append $\{\mathcal{T}_1, \mathcal{T}_0\}$ to $\mathcal{S}$ **otherwise** append $\{\mathcal{T}_0, \mathcal{T}_1\}$ to $\mathcal{S}$;

   otherwise

   1.4. $j \leftarrow \arg\min_{i \in \mathcal{J}} \left| z_i^* - \frac{1}{2} \right|$;

   1.5. $\mathcal{J} \leftarrow \mathcal{J} \setminus j$, $\mathcal{T}_0 \leftarrow \{\mathcal{I}_{\bar{\ell}} \cup \{j\}, \mathcal{I}_{\bar{u}}, z^*\}$; $\mathcal{T}_1 \leftarrow \{\mathcal{I}_{\bar{\ell}}, \mathcal{I}_{\bar{u}} \cup \{j\}, z^*\}$;

   1.6. **if** $z_j^* \leq \frac{1}{2}$ **then** append $\{\mathcal{T}_1, \mathcal{T}_0\}$ to $\mathcal{S}$ **otherwise** append $\{\mathcal{T}_0, \mathcal{T}_1\}$ to $\mathcal{S}$;

2. **if** $card(\mathcal{J}) = p - 1$ **then** (just once below the root node)

   2.1. **set** $\mathcal{J} \leftarrow \{1, \ldots, p\} \setminus (\tilde{\mathcal{I}}_{\bar{\ell}} \cup \tilde{\mathcal{I}}_{\bar{u}})$;

   2.2. $j \leftarrow \arg\min_{i \in \mathcal{J}} \left| z_i^* - \frac{1}{2} \right|$;

   2.3. $\mathcal{J} \leftarrow \mathcal{J} \setminus j$, $\mathcal{T}_0 \leftarrow \{\tilde{\mathcal{I}}_{\bar{\ell}} \cup \{j\}, \tilde{\mathcal{I}}_{\bar{u}}, z^*\}$; $\mathcal{T}_1 \leftarrow \{\tilde{\mathcal{I}}_{\bar{\ell}}, \tilde{\mathcal{I}}_{\bar{u}} \cup \{j\}, z^*\}$;

   2.4. **if** $z_j^* \leq \frac{1}{2}$ **then** append $\{\mathcal{T}_1, \mathcal{T}_0\}$ to $\mathcal{S}$ **otherwise** append $\{\mathcal{T}_0, \mathcal{T}_1\}$ to $\mathcal{S}$; (pushed at the top of $\mathcal{S}$ with the highest priority)

Output: Two subproblems $\mathcal{T}_1, \mathcal{T}_0$ (in appropriate order) are pushed at the top of the stack $\mathcal{S}$.

ing Step 1 of Algorithm 10, until all the nodes of the tree are explored. We ensure not to solve the nodes leading to the leaves already solved while exploring the tree now from beneath the root node.

We illustrate the flow of Algorithm 9+10 with an example with 3 binary variables, warm-started using the sequence $(1, 0, \star)$, that is the warm-

start $z_1 = 1$, $z_2 = 0$. The flow of B&B is shown in Figure 12, where the number inside the node denotes the order in which the QP relaxation is executed. By following Algorithm 9+10, first the root node is solved and then two problems are pushed on the stack with high priority to the $(1, \star, \star)$ node.

Then, the leaf problem $(1, 0, 0)$ is solved first (QP #2), and $(1, 0, 1)$ immediately after (QP #3). Once these two leaves have been solved, problem $(1, \star, \star)$ is popped from the stack (due to the last-in-first-out method), but not solved (depicted as a dashed circle in Figure 12), because we have already explored the children nodes of $(1, 0, \star)$. Indeed, problems $(1, 0, 0)$ and $(1, 0, 1)$ are ignored when popped again from stack herein. Therefore, problem $(1, 1, \star)$ is popped from the stack and solved (QP #4).

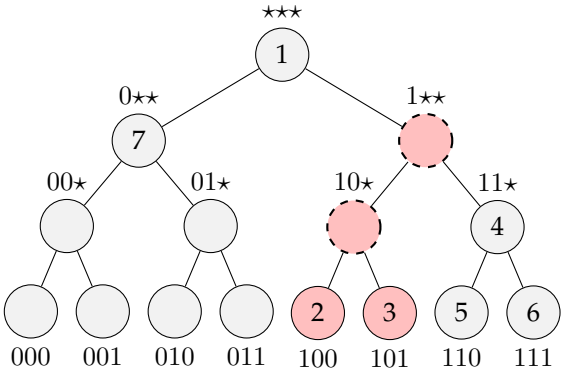


Figure 12: Illustration example with 3 binary variables and $(1, 0, \star)$ as a binary warm-start. The numbers indicate the order in which the QP relaxations are solved, dashed nodes correspond to QP subproblems that are ignored.

We remark that we have saved solving 2 QP relaxations without compromising the optimality of the MIQP solution. In general, using this approach at least $p - 1$ QP relaxations can be saved, even a few more if different branching rules are used. Consider the example shown in Figure 13, in which the binary variable to branch is selected according to the maximum fractional part, and the binary warm-start is $(0, 0, \star)$. In this case, as shown in Figure 13, we save 4 QP subproblems.

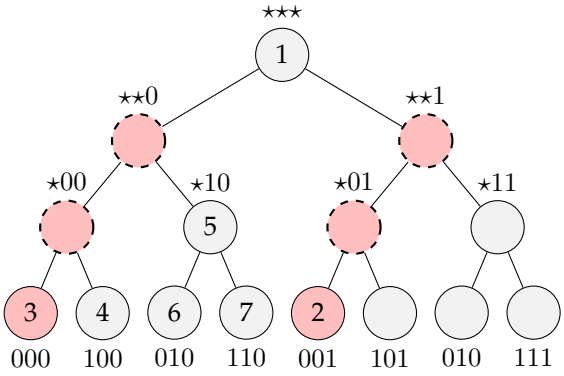


Figure 13: Illustration example with 3 binary variables and $(0, 0, \star)$ as a binary warm-start. The numbers denote the order in which the QP relaxations are solved, dashed nodes correspond to QP subproblems that are ignored.

When solving hybrid MPC in receding horizon fashion, at every sampling instance a sequence z is optimized over the whole horizon of length N, the first computed value of the input is applied to the system, then the horizon window is shifted by one sample time and the procedure repeated. Let $z^* = (v^*_{0|t}, \delta^*_{0|t}, \zeta^*_{0|t}, \ldots, v^*_{N-1|t}, \delta^*_{N-1|t}, \zeta^*_{N-1|t})$ be the optimal solution computed at time t, where v_k, ζ_k, δ_k are the vectors of inputs, real-valued and binary auxiliary variables respectively, corresponding to the optimal trajectory of the MLD model used by hybrid MPC (Bemporad and Morari, 1999a). We can exploit the shifted binary values optimized at the previous step as the binary warm-start, in particular $\delta_{0|t+1} = \delta^*_{1|t}, \ldots, \delta_{N-2|t+1} = \delta^*_{N-1|t}$ (and similarly for binary inputs, if any), and use Algorithm 9 with Steps 2.2.3.1–2.2.3.3 replaced by Algorithm 10. A similar warm-start can be also used to solve MIQP problems arising in moving-horizon estimation and piecewise affine regression, which will be exploited in the following chapter.

3.5.1 Using mid-way approach to warm-start binary variables

The warm-starting for binary variables presented in Section 3.5 is a generic scheme. Due to its flexibility to adapt warm-starting any specified binary variables, the "mid-way" heuristic approach proposed in Section 2.4.2 of Chapter 2, can be considered as a special case of the same framework for solving MIQP to optimality. This can be done by exploiting the information on offer by the "mid-way" heuristic approach.

Specifically, this is done by initializing the sets in Algorithm 10 with the quantities from Algorithm 5 as

$$\tilde{\mathcal{I}}_{\bar{\ell}} \leftarrow \mathcal{K}_{\bar{\ell}},\ \tilde{\mathcal{I}}_{\bar{u}} \leftarrow \mathcal{K}_{\bar{u}},\ \mathcal{J} \leftarrow \mathcal{H}, \tag{3.13a}$$

and similarly initializing B&B Algorithm 3 as

$$V_0 \leftarrow V_H^*, \tag{3.13b}$$

$$\lambda_0, \lambda_{-1} \leftarrow \lambda_R^*,\ \nu \leftarrow \nu_R^*. \tag{3.13c}$$

Here, we warm-start the binary variables with the indices comprised in the sets $\mathcal{K}_{\bar{\ell}}$ and $\mathcal{K}_{\bar{u}}$. Moreover, if the suboptimal integer feasible solution V_H^* is found then it is assigned as the best known integer feasible solution V_0 upon initialization, this helps in pruning the nodes while branching on the variables contained in the set $\mathcal{H}$. In this case, we also utilize the values $z_R^*, \lambda_R^*, \nu_R^*$ and start Algorithm 3 directly from step 2.2.3.1.

We illustrate the flow with an example with 3 binary variables $z_H^* = (0, 0, 1)$ is solution of mid-way heuristic approach, $\mathcal{H} = \{1, 2\}$, $\mathcal{K}_{\bar{\ell}} = \emptyset$, $\mathcal{K}_{\bar{u}} = \{3\}$ that is the warm-start $z_3 = 1$. The flow of B&B is shown in Figure 14, where the number inside the node denotes the order in which the QP relaxation is executed. First, the best known integer feasible solution V_0 is initialized by V_H^*. As z_R^* is available which is equivalent to solution of the root node $(\star, \star, \star)$ (hence the root node is not solved), two problems are pushed on the stack with high priority to the $(\star, \star, 1)$ node.

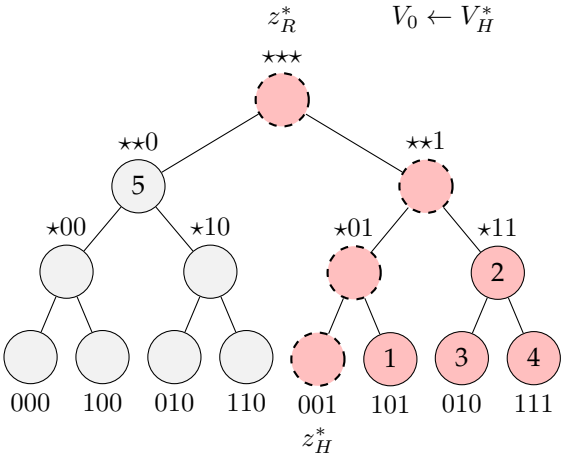


Figure 14: Illustration example with 3 binary variables and $z_H^* = (0, 0, 1)$ as solution of heuristic approach, considered as a binary warm-start, the best know integer feasible solution so far V_0 is initialized as $V_0 \leftarrow V_H^*$, $\mathcal{K}_{\bar{\ell}} = \emptyset$, $\mathcal{K}_{\bar{u}} = \{3\}$, $\mathcal{H} = \{1, 2\}$. The numbers denote the order in which the QP relaxations are solved, dashed nodes correspond to QP subproblems that are ignored.

Then, as warm-start $z_3 = 1$ provided, two problems are pushed on the stack with high priority to the $(\star, 0, 1)$ node, which then is popped from the stack (due to the last-in-first-out method), but not solved (depicted as a dashed circle in Fig. 14), because we have already explored a children node of $(\star, 0, 1)$. Similarly, we have the solution $z_H^* = (0, 0, 1)$, hence it is popped but not solved. Therefore, leaf problem $(1, 0, 1)$ is popped from the stack and solved (QP #1). Then problem $(\star, 1, 1)$ is popped from the stack and solved (QP #2). Indeed, problems $(\star, 0, 1)$, $(0, 0, 1)$ and $(\star, \star, 1)$ are ignored when popped from stack herein.

Similarly, we remark that while using the solution heuristic approach (Section 2.4.1) as upper bound on the cost in B&B setup $V_0 \leftarrow V_H^*$, the values $z_R^*, \lambda_R^*, \nu_R^*$ are also utilized and Algorithm 3 starts directly from step 2.2.3.1 in this case.

3.6 Numerical results

We report numerical experiments carried out on a desktop computer with Intel Core i7-4700MQ CPU 2.40 GHz and 8 GB RAM, using MATLAB R2015a. Algorithms 9, 10 are implemented in interpreted MATLAB code and Algorithm 8 in compiled Embedded MATLAB code.

3.6.1 Hybrid MPC problem

We consider the hybrid MPC problem from (Bemporad and Morari, 1999a, Example 5.1), with the default settings as in `bm99sim.m` which is a part of the Hybrid Toolbox for MATLAB (Bemporad, 2003). This demo considers unit weight on the output of the system and all the other weights are zero. This hybrid MPC problem has been tested for the prediction horizon N from 2 to 15. The performance of Algorithm 8 is compared against commercial solvers GUROBI (Gurobi Optimization, Inc., 2014) and CPLEX IBM, Inc. (2014) with presolvers enabled. The maximum norm of the error in the input trajectories is less than 10^{-4}.

Table 8 report the CPU time taken by the different solvers, where the columns NNLS and NNLS* denote Algorithm 8+9 without warm-starting binary variables and Algorithm 8+9+10 with warm-start of binary variables, respectively. For $N = 10$, the MIQP problem has $n = 40$, $p = 10$ (i.e., 30 continuous variables and 10 binary variables), and $m = 160$ linear inequalities.

3.7 Conclusions

Motivated by embedded hybrid MPC applications, this chapter has proposed a new MIQP solver based on branch and bound that is able to exploit warm-start on binary variables. A numerically efficient and robust QP solver based on an active-set method to solve nonnegative least squares combined with proximal-point iterations is used for solving the QP relaxations, which does not require the Hessian matrix to be positive definite; this is particularly useful in hybrid MPC problems based

Table 8: Hybrid MPC problem: CPU time (ms) per sampling step for different prediction horizons N.

N	NNLS		NNLS*		GUROBI		CPLEX	
	avg	max	avg	max	avg	max	avg	max
2	2.0	2.6	2.0	2.6	1.6	2.0	3.1	6.0
3	3.1	5.1	2.5	4.8	2.7	3.0	6.3	11.5
4	5.3	8.8	3.1	6.9	3.1	3.9	8.9	15.7
5	8.1	16.4	3.9	13.0	3.9	4.8	10.8	15.7
6	13.6	25.5	5.1	18.3	4.7	7.3	11.1	17.1
7	21.7	46.2	6.4	30.2	5.6	9.5	17.5	80.4
8	29.7	71.0	8.1	43.4	7.2	13.2	15.5	80.2
9	49.5	115.9	11.1	69.8	8.8	15.3	22.8	110.6
10	76.2	146.1	14.4	103.2	11.1	17.6	35.1	95.3
11	121.3	254.6	20.6	179.1	13.0	23.9	37.3	102.5
12	155.8	410.8	26.9	263.4	14.9	31.2	61.7	103.7
13	247.6	607.9	35.5	384.9	17.5	36.6	47.3	119.5
14	304.9	893.7	46.3	562.4	21.4	67.4	81.6	150.6
15	484.2	1242.3	61.7	766.9	25.9	109.8	89.9	181.1

on quadratic costs where the quadratic cost is only positive semidefinite due to some variables having zero weight in the MPC cost function. The reported numerical results demonstrate that presented framework for warm-starting binary variables for hybrid MPC problems using the same QP solution algorithm provides better performance in terms of both average and maximum computation time. Moreover, the "mid-way" heuristic approach proposed in Section 2.4.2 of Chapter 2 is demonstrated as a special case of the proposed warm-starting framework. We will utilize the binary warm-start framework presented in this Chapter for the PieceWise Affine (PWA) regression algorithm proposed in the following Chapter.

Chapter 4

Regularized moving-horizon PWA regression using mixed-integer quadratic programming

4.1 Introduction

This chapter presents a novel two-stage regularized moving-horizon algorithm for *PieceWise Affine* (PWA) regression. At the first stage, the training samples are processed iteratively, and a *Mixed-Integer Quadratic-Programming* (MIQP) problem is solved to find the sequence of active modes and the model parameters which best match the training data, within a relatively short time window in the past. According to a moving-horizon strategy, only the last element of the optimal sequence of active modes is kept, and the next sample is processed by shifting forward the estimation horizon. A regularization term on the model parameters is included in the cost of the formulated MIQP problem, to partly take into account also the past training data outside the considered

time horizon. Accelerated Dual Gradient Projection (GPAD) based MIQP solver proposed in Chapter 2 is employed to solve the resulting MIQP problem. At the second stage, linear multi-category discrimination techniques are used to compute a polyhedral partition of the regressor space based on the estimated sequence of active modes.

The proposed PWA regression algorithm is properly adapted for the identification of Linear Parameter-Varying (LPV) models. In addition, the binary warm-starting framework proposed in Chapter 3 is employed for efficient solution of the formulated MIQP problem.

4.1.1 Motivation

PWA systems are heterogeneous systems which exhibit both continuous and discrete dynamics. PWA models are simple and flexible model structures and thanks to their universal approximation properties, any nonlinear function can be modeled with arbitrary accuracy by a PWA map (Breiman, 1993). Furthermore, due to the equivalence between PWA models and several classes of hybrid models (Heemels et al., 2001), available tools for modelling, analysis and control of hybrid systems can be also applied to PWA systems (Bemporad and Morari, 1999a; Bemporad et al., 2000).

PWA regression is an NP-hard problem (Lauer, 2015), where both the regressor space partition and the submodel parameters have to be estimated from a set of training data. Several algorithms/heuristics for PWA regression and for the identification of hybrid systems, have been proposed in the last two decades (see the survey papers (Paoletti et al., 2007; Garulli et al., 2012)). Among these algorithms, we mention the set-membership approaches (Bemporad et al., 2005; Ozay et al., 2015), the sparse-optimization based approaches (Bako, 2011; Ohlsson and Ljung, 2013) and the mixed-integer programming method (Roll et al., 2004). In the latter approach, the estimation of hinging-hyperplane ARX models and piecewise affine Wiener models is formulated as a mixed-integer linear or quadratic programming problem, and then solved through a branch and bound algorithm. As the number of integer variables is pro-

portional to the number of modes and the number of training samples, the approach in (Roll et al., 2004) is limited to medium-scale problems. A study of various mixed-integer programming formulations for piecewise linear functions is carried out in (Vielma et al., 2010). The contributions (Bemporad et al., 2005; Nakada et al., 2005; Juloski et al., 2005; Ferrari-Trecate et al., 2003; Bako et al., 2011; Breschi et al., 2016b,a) fall in the class of cluster-based two-stage methods. At the first stage, the training samples are clustered by assigning each datapoint to a submodel according to a certain criterion and, at the same time, the parameters of the affine submodels are estimated. In the second stage, the polyhedral partition of the regressor space is computed by linear separation techniques. Unlike the mixed-integer programming approach (Roll et al., 2004), which can be solved for the global optimum, suboptimal solutions are obtained by the aforementioned two-stage methods.

4.1.2 Contribution

In this chapter, we propose a regularized moving-horizon PWA regression algorithm, which can be seen as a mix between the mixed-integer programming approach (Roll et al., 2004) and the cluster-based algorithm (Breschi et al., 2016b). At the first stage, a Mixed-Integer Quadratic-Programming (MIQP) problem is formulated to compute both the optimal sequence of active modes within the considered horizon and the parameters of the affine submodels. Moreover, a regularization term is included in the cost of the formulated MIQP problem, to exploit the information from the past training samples outside the considered time window. Thus, the length N_p of the horizon acts as a knob to combine the advantages of the two-stage algorithm (Breschi et al., 2016b) (namely, computational efficiency and iterative processing of the training samples) and the advantages of the mixed-integer programming approach (Roll et al., 2004) (namely, non-decoupled optimization over the active modes and the submodel parameters). According to a moving-horizon strategy, only the active mode at current sampling time is extracted from the computed optimal sequence of active modes, and the

next training sample is processed by shifting forward the estimation horizon. At the second stage, the regressor space is partitioned using computationally efficient multi-class linear separation methods proposed in (Breschi et al., 2016b).

4.1.3 Outline

The organization of this chapter is as follows. Section 4.2 describes the formulation of the PWA regression problem. Section 4.3 contains the proposed identification algorithm. A proper adaptation of the proposed PWA regression algorithm for the identification of Linear Parameter-Varying (LPV) systems is briefly described in Section 4.3.3. The simulation examples on the identification of *PieceWise Affine autoRegressive with eXogenous inputs* (PWARX) dynamical systems and a LPV system are reported in Section 4.4 to show the effectiveness of the proposed approach. The final remarks are given in Section 4.5.

4.2 Problem Formulation

Let us consider a data-generating system in the form

$$y(k) = f_{\mathrm{o}}(x(k)) + e_{\mathrm{o}}(k), \tag{4.1}$$

where $k \in \mathbb{N}$ is the time index, $y(k) \in \mathbb{R}^{n_y}$ is the measured output at time k, $e_{\mathrm{o}}(k) \in \mathbb{R}^{n_y}$ is an additive random noise, $x(k) \in \mathbb{R}^{n_x}$ is the regressor vector which is assumed to take values in a set $\mathcal{X} \subset \mathbb{R}^{n_x}$, and $f_{\mathrm{o}} : \mathcal{X} \to \mathbb{R}^{n_y}$ is an unknown and possible discontinuous function.

In this chapter, we address a PWA regression problem, which amounts at computing a PWA function $f : \mathcal{X} \to \mathbb{R}^{n_y}$ approximating the regression function f_{o} based on a set of N observations of the regressor/output pairs $\{x(k), y(k)\}_{k=1}^{N}$. The PWA vector-valued function f is described as:

$$f(x) = \begin{cases} \Theta_1 \left[\begin{smallmatrix} 1 \\ x \end{smallmatrix}\right] & \text{if } x \in \mathcal{X}_1, \\ \vdots & \\ \Theta_s \left[\begin{smallmatrix} 1 \\ x \end{smallmatrix}\right] & \text{if } x \in \mathcal{X}_s, \end{cases} \tag{4.2}$$

where $s \in \mathbb{N}$ denotes the number of modes, $\Theta_i \in \mathbb{R}^{n_y \times (n_x+1)}$ are parameter matrices, and $\mathcal{X}_i$, with $i = 1, \ldots, s$, are polyhedra ($\mathcal{H}_i x \leq \mathcal{D}_i$) that form a complete polyhedral partition[1] of the regressor space $\mathcal{X}$.

Estimation of the PWA function f in (4.2) thus requires: (i) selecting the number of modes s; (ii) estimating the parameter matrices Θ_i; and (iii) finding the polyhedra $\mathcal{X}_i$ (i.e., the matrices $\mathcal{H}_i$ and $\mathcal{D}_i$) defining the partition of the regressor space $\mathcal{X}$. Tradeoff between data fitting and model complexity should be taken into account while choosing the number of modes s. If the number of modes s is small, then the PWA map f may not be flexible enough to capture the shape of the underlying nonlinear data-generating function f_o (4.1). On the contrary, considering the high number of modes results in a more accurate description of the PWA map f with more degrees of freedom. However, this may cause overfitting and poor generalization to unseen data (not used in the training phase) as the final estimate is sensitive to the noise corrupting the observations, besides increasing the complexity of the estimation procedure and of the resulting PWA model. In the rest of the chapter, we assume that s is fixed by the user, and chosen via cross-calibration. This is done by evaluating the performance of the estimated model for different values of s, on a fresh dataset which is different from the training dataset.

4.3 PWA Regression Algorithm

The developed algorithm for PWA regression consists of the following two stages:

S1. Recursive *estimation* of the model parameters Θ_i and simultaneous *clustering* of the regressors $\{x(k)\}_{k=1}^{N}$.

S2. *Computation of a polyhedral partition* of the regressor space $\mathcal{X}$ using computationally efficient multi-category linear separation methods already available in the literature. This can be computed either offline or online (recursively) and is executed after S1.

[1] A collection $\{\mathcal{X}_i\}_{i=1}^{s}$ is a complete partition of the regressor domain $\mathcal{X}$ if $\bigcup_{i=1}^{s} \mathcal{X}_i = \mathcal{X}$ and $\mathring{\mathcal{X}}_i \cap \mathring{\mathcal{X}}_j = \emptyset, \forall i \neq j$, with $\mathring{\mathcal{X}}_i$ denoting the interior of $\mathcal{X}_i$.

4.3.1 Recursive clustering and parameter estimation

Stage **S1** is carried out through a regularized moving-horizon identification algorithm. The training regressor/output pairs $\{x(k), y(k)\}$ are processed iteratively. At each time sample k, a moving-horizon window of length N_p containing regressor/output pairs from time $k-N_p+1$ to time k is considered. The model parameters Θ_i and the active mode $\sigma(k)$ at time k are estimated simultaneously by solving the mixed-integer programming problem:

$$\min_{\substack{\{\Theta_i\}_{i=1}^{s} \\ \{\delta_i(k-t)\}_{i=1,t=0}^{s,N_p-1}}} \sum_{i=1}^{s}\sum_{t=0}^{N_p-1} \left\|\left(y(k-t)-\Theta_i\begin{bmatrix}1\\ x(k-t)\end{bmatrix}\right)\delta_i(k-t)\right\|^2 \tag{4.3a}$$

$$+\sum_{t=1}^{k-N_p}\left\|y(t)-\Theta_{\sigma(t)}\begin{bmatrix}1\\ x(t)\end{bmatrix}\right\|^2 \tag{4.3b}$$

$$\text{s.t. } \delta_i(k-t)\in\{0,1\}, \sum_{i=1}^{s}\delta_i(k-t)=1,\ t=0,\dots,N_p-1. \tag{4.3c}$$

The active mode $\sigma(k) \in \{1,\dots,s\}$ represents the cluster that the regressor $x(k)$ is assigned to, and it is extracted from the optimizer of problem (4.3), i.e.,

$$\sigma(k) = i^*, \quad \text{with } i^* : \delta_{i^*}(k) = 1. \tag{4.4}$$

According to a moving-horizon estimation strategy, only the active mode $\sigma(k)$ at time k is kept, and the N_p-length time window is shifted forward to process the next pair $\{x(k+1), y(k+1)\}$.

Problem (4.3) aims at searching for the optimal sequence of active modes within the considered time window and the model parameters Θ_i which best match the available observations up to time k. Note that the term (4.3a) aims at finding both the model parameters and the sequence of active modes $\{\sigma(t)\}_{t=k-N_p+1}^{k}$ which best match the observations within the N_p-step time horizon. The term (4.3b) acts as a regularization term on the parameters Θ_i and it takes into account the time history of the observations outside the considered time window. More

specifically, in (4.3b), the sequence of active modes is not optimized from time 1 to time $k - N_p$, but it is fixed to the estimates $\{\sigma(t)\}_{t=1}^{k-N_p}$ obtained from the previous iterations of the moving-horizon estimation algorithm. In-turn, the sequence of active modes is optimized only within the considered time horizon in (4.3a).

Increasing N_p increases the information used to cluster the regressor $x(k)$ and to estimate the model parameters Θ_i. On the one hand, increasing N_p increases the number of binary decision variables δ_i in (4.3). Thus, the length N_p of the horizon provides a trade off between complexity of the optimization problem (4.3), and accuracy in estimating the model parameters Θ_i and in clustering the regressor $x(k)$.

Recursive update of the objective function

Note that, at a first glance, the regularization cost (4.3b) requires to use, and thus to store, the whole time-history of observations up to time $k - N_p$ (i.e., the sequence of regressor/output pairs $\{x(k), y(k)\}_{k=1}^{k-N_p}$). Nevertheless, once a new observation is available at time k, the term (4.3b) can be recursively updated, as described in the following.

Let us rewrite the regularization term (4.3b) as

$$\begin{aligned}
&\sum_{t=1}^{k-N_p} \mathrm{tr}\Big(\big(y(t) - \Theta_{\sigma(t)} \left[\begin{smallmatrix} 1 \\ x(t) \end{smallmatrix}\right]\big) \big(y(t) - \Theta_{\sigma(t)} \left[\begin{smallmatrix} 1 \\ x(t) \end{smallmatrix}\right]\big)' \Big) = \\
&\mathrm{tr}\left(\sum_{t=1}^{k-N_p} \Theta_{\sigma(t)} \left[\begin{smallmatrix} 1 \\ x(t) \end{smallmatrix}\right] \left[\begin{smallmatrix} 1 \\ x(t) \end{smallmatrix}\right]' \Theta'_{\sigma(t)} \right) - \\
&2\mathrm{tr}\left(\sum_{t=1}^{k-N_p} \Theta_{\sigma(t)} \left[\begin{smallmatrix} 1 \\ x(t) \end{smallmatrix}\right] y(t)' \right) + \mathrm{tr}\left(\sum_{t=1}^{k-N_p} y(t) y(t)' \right),
\end{aligned} \tag{4.5}$$

with $\mathrm{tr}(\cdot)$ denoting the matrix trace. Let us now define the matrices

$$H_i(k - N_p) = \sum_{t=1}^{k-N_p} \left[\begin{smallmatrix} 1 \\ x(t) \end{smallmatrix}\right] \left[\begin{smallmatrix} 1 \\ x(t) \end{smallmatrix}\right]' h_i(t), \tag{4.6a}$$

$$F_i(k - N_p) = \sum_{t=1}^{k-N_p} \left[\begin{smallmatrix} 1 \\ x(t) \end{smallmatrix}\right] y(t)' h_i(t), \tag{4.6b}$$

with $h_i(t) = 1$, if $\sigma(t) = i$ or $h_i(t) = 0$, otherwise. Substituting (4.6) into the cost (4.5), we can represent (4.3b) as

$$\sum_{t=1}^{k-N_p} \left\| y(t) - \Theta_{\sigma(t)} \begin{bmatrix} 1 \\ x(t) \end{bmatrix} \right\|^2 = \mathrm{tr}\left(\sum_{i=1}^{s} \Theta_i H_i(k-N_p) \Theta_i' \right) - 2\mathrm{tr}\left(\sum_{i=1}^{s} \Theta_i F_i(k-N_p) \right) + \mathrm{tr}\left(\sum_{t=1}^{k-N_p} y(t) y(t)' \right). \tag{4.7}$$

Note that the matrices $H_i(k-N_p)$ and $F_i(k-N_p)$ can be recursively computed as

$$\begin{aligned} H_i(k-N_p) =& H_i(k-N_p-1) + \\ & \begin{bmatrix} 1 \\ x(k-N_p) \end{bmatrix} \begin{bmatrix} 1 \\ x(k-N_p) \end{bmatrix}' h_i(k-N_p), \qquad (4.8a) \\ F_i(k-N_p) =& F_i(k-N_p-1) + \\ & \begin{bmatrix} 1 \\ x(k-N_p) \end{bmatrix} y(k-N_p)' h_i(k-N_p). \qquad (4.8b) \end{aligned}$$

Thus, processing the observation $\{x(k), y(k)\}$ just requires to update the matrices $H_i(k-N_p-1)$ and $F_i(k-N_p-1)$ through (4.8), with no need to store the time history of observations $\{x(k), y(k)\}_{t=1}^{k-N_p-1}$.

MIQP formulation

To reformulate (4.3) as an MIQP problem, let us define the vector $z_i(k) \in \mathbb{R}^{n_y}$ with $\delta_i(k) \in \{0, 1\}$ as

$$z_i(k) = \left(y(k) - \Theta_i \begin{bmatrix} 1 \\ x(k) \end{bmatrix} \right) \delta_i(k). \tag{4.9a}$$

Note that:

$$z_i(k) = \begin{cases} y(k) - \Theta_i \begin{bmatrix} 1 \\ x(k) \end{bmatrix} & \text{if } \delta_i(k) = 1 \\ 0 & \text{if } \delta_i(k) = 0 \end{cases} \tag{4.9b}$$

Let M and m be an arbitrary large (resp. small) upper (resp. lower) bound of the elements of the vector $y(k) - \Theta_i \begin{bmatrix} 1 \\ x(k) \end{bmatrix}$,

$$m \leq y(k) - \Theta_i \begin{bmatrix} 1 \\ x(k) \end{bmatrix} \leq M. \tag{4.9c}$$

Based on conditions (4.7) and (4.9), problem (4.3) can be equivalently written as the MIQP problem

$$\min_{\substack{\{\Theta_i\}_{i=1}^{s} \\ \{\delta_i(k-t)\}_{i=1,t=0}^{s,N_p-1} \\ \{z_i(k-t)\}_{i=1,t=0}^{s,N_p-1}}} \sum_{t=0}^{N_p-1}\sum_{i=1}^{s} z_i^2(k-t)+ \tag{4.10a}$$

$$\operatorname{tr}\left(\sum_{i=1}^{s}\Theta_i H_i(k-N_p)\Theta_i'\right)-2\operatorname{tr}\left(\sum_{i=1}^{s}\Theta_i F_i(k-N_p)\right), \tag{4.10b}$$

$$\text{s.t. } z_i(k-t) \le M\delta_i(k-t), \tag{4.10c}$$

$$z_i(k-t) \ge m\delta_i(k-t), \tag{4.10d}$$

$$z_i(k-t) \le y(k-t)-\Theta_i\begin{bmatrix}1\\ x(k-t)\end{bmatrix}-m(1-\delta_i(k-t)), \tag{4.10e}$$

$$z_i(k-t) \ge y(k-t)-\Theta_i\begin{bmatrix}1\\ x(k-t)\end{bmatrix}-M(1-\delta_i(k-t)), \tag{4.10f}$$

$$\sum_{i=1}^{s}\delta_i(k-t)=1,\ t=0,\dots,N_p-1, \tag{4.10g}$$

$$\delta_i(k-t)\in\{0,1\},\ i=1,\dots,s, \tag{4.10h}$$

Based on the constructed problem (4.10), different MIQP solvers can be chosen to solve (4.10). For small-scale problems, GPAD-B&B approach introduced in Chapter 2 can be used. The GPAD algorithm (Patrinos and Bemporad, 2014) is very simple as it only needs basic arithmetic computations, which is specialized to solve the Quadratic Programming (QP) relaxations arising in B&B. GUROBI solver can be used to solve medium and large-scale problems.

Summary and iterative refinement

The steps described so far for recursive clustering of the regressors $\{x(k)\}_{k=1}^{N}$ and for model parameters Θ_i estimation are summarized in Algorithm 11. At the beginning of Algorithm 11, a mini-batch identification problem is solved to estimate the sequence of active modes $\sigma(t)$ from time 1 up to time N_p and to assign the regressor $\{x(t)\}_{t=1}^{N_p}$ to the cluster $\{\mathcal{C}_{\sigma(t)}\}_{t=1}^{N_p}$ (stages 2-6). Then, the observations $\{x(k), y(k)\}$ are

processed iteratively. Besides updating the model parameters Θ_i at each time k (stage 7.3), the active mode $\sigma(k)$ is estimated (stages 7.4-7.6) and the regressor $x(k)$ is consequently assigned to cluster $\mathcal{C}_{\sigma(k)}$ (stage 7.7).

It is worth pointing out that, at the first iterations of Algorithm 11 (i.e., for $\bar{k} \ll N$), the observations $\{x(t), y(t)\}_{t=1}^{\bar{k}}$ may be wrongly classified, as the learning phase is based on a "small" set of observations. An error in the classification of the pairs $\{x(t), y(t)\}_{t=1}^{\bar{k}}$ (i.e., an incorrect estimate of the mode sequence $\{\sigma(t)\}_{t=1}^{\bar{k}}$) also influences the estimate of the active modes and of the model parameters Θ_i at the next time samples $k > \bar{k}$, as the regularization cost (4.3b) depends on the estimated sequence $\{\sigma(t)\}_{t=1}^{\bar{k}}$. When working in a batch mode, the effect of the initial classification error may be reduced by running Algorithm 11 multiple times, including in the regularization term (4.3b) also the sequences of active modes estimated at the previous runs. More specifically, at the n_q-th run of Algorithm 11, the following cost is considered instead of (4.3a)-(4.3b):

$$\sum_{i=1}^{s} \sum_{t=0}^{N_p-1} \gamma_1 \left\| \left(y(k-t) - \Theta_i \begin{bmatrix} 1 \\ x(k-t) \end{bmatrix} \right) \delta_i(k-t) \right\|^2 + \tag{4.11a}$$

$$\gamma_2 \sum_{t=1}^{k-N_p} \left\| y(t) - \Theta_{\sigma(t,n_q)} \begin{bmatrix} 1 \\ x(t) \end{bmatrix} \right\|^2 + \tag{4.11b}$$

$$\sum_{q=1}^{n_q-1} \lambda^{n_q-q-1} \sum_{t=1}^{N-N_p} \left\| y(t) - \Theta_{\sigma(t,q)} \begin{bmatrix} 1 \\ x(t) \end{bmatrix} \right\|^2, \tag{4.11c}$$

with $\sigma(t, q)$ ($q = 1, \ldots, n_q$) being the estimate of the active mode at time t obtained at the q-th run of Algorithm 11. Note that (4.11c) is a regularization term based on the past runs of Algorithm 11, while (4.11b) plays the same role of (4.3b), as it regularizes the parameters Θ_i based on the estimate $\{\sigma(t)\}_{t=1}^{k-N_p}$ obtained at the current run of Algorithm 11. A forgetting factor $\lambda \in \mathbb{R} : 0 < \lambda \leq 1$ is also included in (4.11c) to exponentially downweight the estimates $\{\sigma(t, q)\}_{t=1}^{N}$ obtained at past runs of Algorithm 11. The regularization parameters $\gamma_1, \gamma_2 \in \mathbb{R}$ are introduced in (4.11a) and (4.11b), respectively.

Similar to the original regularization term (4.3b), the cost (4.11)

Algorithm 11 Recursive clustering of the regressors and model parameters estimation

Input: Observations sequence $\{x(k), y(k)\}_{k=1}^{N}$; number of modes s; horizon N_p.

1. **let** $H_i(0) \leftarrow 0$, $F_i(0) \leftarrow 0$, $\mathcal{C}_i \leftarrow \emptyset$, $i = 1, \ldots, s$;
2. **let** $k \leftarrow N_p$;
3. **solve** the MIQP problem (4.10);
4. **let** $\{\delta_i^*(t)\}_{i=1,t=1}^{s,N_p}$ be the optimal parameters minimizing (4.10);
5. **for** $t = 1, \ldots, N_p$ **do**
   - 5.1. **let** $i^*(t)$ be the index such that $\delta_{i^*}^*(t) = 1$;
   - 5.2. **let** $\sigma(t) \leftarrow i^*(t)$;
   - 5.3. **let** $\mathcal{C}_{\sigma(t)} \leftarrow \mathcal{C}_{\sigma(t)} \cup \{x(t)\}$;
6. **end for**
7. **for** $k = N_p + 1, \ldots, N$ **do**
   - 7.1. **update** the matrices $H_i(k - N_p)$ and $F_i(k - N_p)$ through (4.8);
   - 7.2. **solve** the MIQP problem (4.10);
   - 7.3. **let** $\Theta_i^*(k)$ be the optimal parameters minimizing (4.10), $i = 1, \ldots, s$;
   - 7.4. **let** $\{\delta_i^*(k - t)\}_{t=0}^{N_p-1}$ be the optimal parameters minimizing (4.10), $i = 1, \ldots, s$;
   - 7.5. **let** i^* be the index such that $\delta_{i^*}^*(k) = 1$;
   - 7.6. **let** $\sigma(k) \leftarrow i^*$;
   - 7.7. **let** $\mathcal{C}_{\sigma(k)} \leftarrow \mathcal{C}_{\sigma(k)} \cup x(k)$;
8. **end for**;

Output: Estimated parameters $\Theta_1^*(N), \ldots, \Theta_s^*(N)$; clusters $\mathcal{C}_1, \ldots, \mathcal{C}_s$; sequence of active modes $\{\sigma(k)\}_{k=1}^{N}$.

can be also recursively updated as a new sample $\{x(k), y(k)\}$ is processed, without the need to store the whole time history of estimates $\{\sigma(k,q)\}_{k=1,q=1}^{N,n_q-1}$ obtained at the previous runs of Algorithm 11. As a matter of fact, the cost (4.11) can be written as

$$\sum_{i=1}^{s}\sum_{t=0}^{N_p-1}\gamma_1\left\|\left(y(k-t)-\Theta_i\left[\begin{smallmatrix}1\\ x(k-t)\end{smallmatrix}\right]\right)\delta_i(k-t)\right\|^2+ \tag{4.12a}$$

$$\gamma_2\left\{\mathrm{tr}\left(\sum_{i=1}^{s}\Theta_i H_i(k-N_p,n_q)\Theta_i'\right)- \tag{4.12b}$$

$$2\mathrm{tr}\left(\sum_{i=1}^{s}\Theta_i F_i(k-N_p,n_q)\right)+ \tag{4.12c}$$

$$\mathrm{tr}\left(\sum_{t=1}^{k-N_p}y(t)y(t)'\right)\right\}+ \tag{4.12d}$$

$$\sum_{q=1}^{n_q-1}\mathrm{tr}\left(\sum_{i=1}^{s}\Theta_i\lambda^{n_q-q-1}H_i(N-N_p,q)\Theta_i'\right)- \tag{4.12e}$$

$$2\sum_{q=1}^{n_q-1}\mathrm{tr}\left(\sum_{i=1}^{s}\Theta_i\lambda^{n_q-q-1}F_i(N-N_p,q)\right)+ \tag{4.12f}$$

$$\sum_{q=1}^{n_q-1}\mathrm{tr}\left(\lambda^{n_q-q-1}\sum_{t=1}^{N-N_p}y(t)y(t)'\right), \tag{4.12g}$$

where $H_i(N-N_p,q)$ and $F_i(N-N_p,q)$ $(q=1,\ldots,n_q)$ are defined similarly to (4.6) and computed based on the estimates $\sigma(t,q)$ computed at the q-th run of Algorithm 11. Specifically:

$$H_i(N-N_p,q)=\sum_{t=1}^{N-N_p}\left[\begin{smallmatrix}1\\ x(t)\end{smallmatrix}\right]\left[\begin{smallmatrix}1\\ x(t)\end{smallmatrix}\right]'h_i(t,q),$$

$$F_i(N-N_p,q)=\sum_{t=1}^{N-N_p}\left[\begin{smallmatrix}1\\ x(t)\end{smallmatrix}\right]y(t)'h_i(t,q),$$

with $h_i(t,q)=0$, if $\sigma(t,q)=i$ or $h_i(t,q)=1$, otherwise.

When the pair $\{x(k),y(k)\}$ is processed, the matrices $H_i(k-N_p,n_q)$ and $F_i(k-N_p,n_q)$ in (4.12b)-(4.12c) can be recursively updated

through (4.8), while only the matrices $H_i(N - N_p, q)$ and $F_i(N - N_p, q)$ (with $q = 1, \dots, n_q - 1$) are needed to construct the terms (4.12e)-(4.12f).

4.3.2 Construction of the state partition

The partition $\{\mathcal{X}_i\}_{i=1}^s$ of the regressor space $\mathcal{X}$ can be found along with the estimation of the model parameters $\{\Theta_i\}_{i=1}^s$ and the sequence of active modes $\{\sigma(k)\}_{k=1}^N$. This is done by separating the computed clusters $\{\mathcal{C}_i\}_{i=1}^s$ using linear multicategory discrimination.

In the following subsection, we briefly describe the algorithms recently presented in (Breschi et al., 2016b), which are suited both for offline and online (i.e., recursive) computation of the state partition.

Linear multicategory discrimination: problem formulation

According to the formulation introduced in (Bennett and Mangasarian, 1994), the linear multicategory discrimination problem is tackled by searching for a convex piecewise affine separator function $\phi : \mathbb{R}^{n_x} \to \mathbb{R}$ discriminating between the clusters $\mathcal{C}_1, \dots, \mathcal{C}_s$. The separator function ϕ is defined as

$$\phi(x) = \max_{i=1,\dots,s} \left(\begin{bmatrix} x' & -1 \end{bmatrix} \begin{bmatrix} \omega^i \\ \gamma^i \end{bmatrix} \right), \tag{4.14}$$

where $\omega^i \in \mathbb{R}^{n_x}$ and $\gamma^i \in \mathbb{R}$ are the parameters to be computed. Let m_i denote the cardinality of the cluster $\mathcal{C}_i$ and let $M_i \in \mathbb{R}^{m_i \times n_x}$, for $i = 1, \dots, s$, which is obtained by stacking the regressors $x(k)'$ belonging to $\mathcal{C}_i$ in its rows.

If the clusters $\{\mathcal{C}_i\}_{i=1}^s$ are linearly separable, then the separator function ϕ satisfies the following conditions:

$$\begin{bmatrix} M_i & -\mathbf{1}_{m_i} \end{bmatrix} \begin{bmatrix} \omega^i \\ \gamma^i \end{bmatrix} \geq \begin{bmatrix} M_i & -\mathbf{1}_{m_i} \end{bmatrix} \begin{bmatrix} \omega^j \\ \gamma^j \end{bmatrix} + \mathbf{1}_{m_i}, \qquad i, j = 1, \dots, s,\ i \neq j, \tag{4.15}$$

where $\mathbf{1}_{m_i}$ is an m_i-dimensional vector of ones.

The piecewise-affine separator ϕ thus satisfies the conditions:

$$\begin{cases} \phi(x) = [x' \quad -1] \begin{bmatrix} \omega^i \\ \gamma^i \end{bmatrix}, \ \forall x \in \mathcal{C}_i, \ \ i = 1, \dots, s \\ \phi(x) \geq [x' \quad -1] \begin{bmatrix} \omega^j \\ \gamma^j \end{bmatrix} + 1, \ \forall x \in \mathcal{C}_i, \ i \neq j \end{cases} \tag{4.16}$$

From (4.16), the polyhedra $\{\mathcal{X}_i\}_{i=1}^s$ are defined as

$$\mathcal{X}_i = \left\{ x \in \mathbb{R}^{n_x} : [x' \quad -1] \begin{bmatrix} \omega^i - \omega^j \\ \gamma^i - \gamma^j \end{bmatrix} \geq 1, \ j = 1, \dots, s, \ j \neq i \right\}.$$

Off-line multicategory discrimination

The parameters $\{\omega^i, \gamma^i\}_{i=1}^s$ are calculated by solving the optimization problem (Breschi et al., 2016b) (instead of solving a *robust linear programming* (RLP) problem as in (Bennett and Mangasarian, 1994)),

$$\begin{aligned} &\min_{\xi} \frac{\kappa}{2} \sum_{i=1}^{s} \left(\|\omega^i\|_2^2 + (\gamma^i)^2 \right) + \\ &\sum_{i=1}^{s} \sum_{\substack{j=1 \\ j \neq i}}^{s} \frac{1}{m_i} \left\| \left(\begin{bmatrix} M_i & -\mathbf{1}_{m_i} \end{bmatrix} \begin{bmatrix} \omega^j - \omega^i \\ \gamma^j - \gamma^i \end{bmatrix} + \mathbf{1}_{m_i} \right)_+ \right\|_2^2, \end{aligned} \tag{4.17}$$

with $\xi = \begin{bmatrix} (\omega^1)' & \dots & (\omega^s)' & \gamma^1 & \dots & \gamma^s \end{bmatrix}'$. Problem (4.17) minimizes the averaged squared 2-norm of the violation of the inequalities in (4.15). The regularization parameter $\kappa > 0$ guarantees that the objective function (4.17) is strongly convex. Problem (4.17) is then solved through the *Regularized Piecewise-Smooth Newton* (RPSN) method explained in (Breschi et al., 2016b) and originally proposed in (Bemporad et al., 2015).

Recursive multicategory discrimination

An online approach can be used, either in place of the off-line approach or to refine the partition ϕ online based on streaming data. A recursive approach using on-line convex programming can be used to solve (4.17).

Considering the data-points $x \in \mathbb{R}^{n_x}$ as random vectors and let us assume that there exists an oracle function $i : \mathbb{R}^{n_x} :\to \{1, \dots, s\}$ that assigns the corresponding mode $i(x) \in \{1, \dots, s\}$ to a given $x \in \mathbb{R}^{n_x}$. By

definition, function i describes the clusters in the data-point space $\mathbb{R}^{n_x}$. Let us also assume that the following probabilities

$$\pi_i = \text{Prob}[i(x) = i] = \int_{\mathbb{R}^{n_x}} \delta(i, i(x)) p(x) dx,$$

are known for all $i = 1, \ldots, s$, where $\delta(i, j) = 1$ if $i = j$, zero otherwise. Problem (4.17) can be the generalized as the following convex regularized stochastic optimization problem

$$\xi^* = \min_{\xi} E_{x \in \mathbb{R}^{n_x}} \left[\ell(x, \xi) \right] + \frac{\kappa}{2} \|\xi\|_2^2 \tag{4.18}$$

$$\ell(x, \xi) = \sum_{\substack{j=1 \\ j \neq i(x)}}^{s} \frac{1}{\pi_{i(x)}} \left(x'(\omega^j - \omega^{i(x)}) - \gamma^j + \gamma^{i(x)} + 1 \right)_+^2,$$

where $E_x[\cdot]$ denotes the expected value w.r.t. x. Problem (4.18) aims at violating the least, on average over x, the condition in (4.15) for $i = i(x)$. The coefficients π_i can be estimated offline from a data subset, specifically $\pi_i = \frac{m_i}{N}$, and can be updated iteratively. Nevertheless, numerical experiments have shown that uniform coefficients $\pi = \frac{1}{s}$ work equally well. Problem (4.18) can be solved online using convex optimization algorithm, *Averaged Stochastic Gradient Descent* (ASGD) method described in (Breschi et al., 2016b).

4.3.3 Regularized Moving-Horizon PWA Regression for LPV System Identification

The proposed regularized moving-horizon *PieceWise Affine* (PWA) regression algorithm is employed for the identification of *Linear Parameter-Varying* (LPV) models. Specifically, the scheduling-variable space is partitioned into polyhedral regions, where each region is assigned to a PWA function describing the local affine dependence of the LPV model coefficients on the scheduling variable. Both the local affine functions and the partition of the scheduling variable domain are directly estimated from data, by properly adapting the regularized moving-horizon algorithm for PWA regression proposed in the former section. See (Mejari et al., 2018b) for further details.

4.4 Simulation Examples

The performance of the proposed PWA regression algorithm is shown via identification of *PieceWise Affine autoRegressive with eXogenous input* (PWARX) systems in this section. The output sequence used for training is corrupted by a zero mean white Gaussian noise process e_{o}. The *Signal-to-Noise Ratio* (SNR) index

$$\mathrm{SNR} = 10 \log \frac{\sum_{t=1}^{N} (y(t) - e_{\mathrm{o}}(t))^2}{\sum_{t=1}^{N} (e_{\mathrm{o}}(t))^2}, \tag{4.19}$$

quantifies the effect of the measurement noise on the output. The quality of the identified models is assessed on a noiseless validation dataset (not used for training) through the *Best Fit Rate* (BFR) index

$$\mathrm{BFR} = \max \left\{ 1 - \sqrt{\frac{\sum_{k=1}^{N_{\mathrm{val}}} \left(y(k) - \hat{y}(k)\right)^2}{\sum_{k=1}^{N_{\mathrm{val}}} \left(y(k) - \bar{y}\right)^2}}, 0 \right\}, \tag{4.20}$$

with N_{val} being the length of the validation set and $\hat{y}$ being the estimated model output and $\bar{y}$ the sample mean of the output signal. All the simulations are carried out using a desktop computer with MATLAB R2015a, Intel Core i7-4700MQ CPU with 2.40 GHz and 8 GB of RAM.

4.4.1 Identification of SISO PWARX system

As a first example, we consider a *single-input single-output* (SISO) PWARX system for the data generation, described by the difference equation

$$\begin{aligned} y(k) =& 0.8y(k-1) + 0.4u(k-1) - 0.1 + \max\left\{-0.3y(k-1)\right. \\ & \left. +0.6u(k-1) + 0.3, 0\right\} + e_{\mathrm{o}}(k), \end{aligned}$$

with $\bar{s} = 2$ modes, based on the possible combinations generated by the sign of the "max" operator. To gather the data, the system is excited by an input $u(k)$ which is chosen to be white noise with uniform distribution $\mathcal{U}(-1, 1)$ and length $N = 1000$, $e_{\mathrm{o}}(k) \in \mathbb{R}$ is a zero-mean white Gaussian noise with variance $\sigma_e^2 = 6.25 \cdot 10^{-4}$. This corresponds to a SNR on the output channel equal to 20 dB.

Table 9: Best Fit Rate on the validation dataset for SISO PWARX system.

runs(n_q)	GUROBI	miqpGPAD
1	0.86	0.86
2	0.90	0.89
3	0.92	0.90

For identification, Algorithm 11 is run for 3 iterations (i.e., $n_q = 3$) with $s = \bar{s} = 2$ and with prediction horizon $N_p = 5$, forgetting factor $\lambda = 0.01$, $\gamma_1 = 10,\ \gamma_2 = 1$. The resultant MIQP problem (4.10) consists of 26 variables out of which 10 are binary, 40 inequality and 5 equality constraints. The MIQP problem is solved with the GPAD-B&B algorithm proposed in Chapter 2, denoted by miqpGPAD and the performance is compared with the commercial solver GUROBI. In the second stage, off-line multicategory discrimination algorithm (section 4.3.2) is executed for the partitioning of the regressor space, with parameter $\kappa = 10^{-5}$. The BFR on the noise-free validation dataset of length $N_{\text{val}} = 300$ are summarized in Table (9). The mean time taken to process a single training sample by miqpGPAD is 0.13 sec with the feasibility tolerance $\epsilon_G = 10^{-3}$, optimality tolerance $\epsilon_V = 10^{-3}$, infeasibility detection tolerance $\epsilon_I = 10^{-3}$, whereas GUROBI with default settings takes 0.09 sec. The method miqpGPAD makes a trade off between the execution time and quality of solution, by selecting appropriate tolerance values. It is simple library-free solver, yet rendered comparable performance with respect to the commercial solver for the given problem.

The effect of increasing the prediction horizon N_p on the BFR is shown in Figure 15, for $n_q = 1, 2$ and 3 runs respectively.

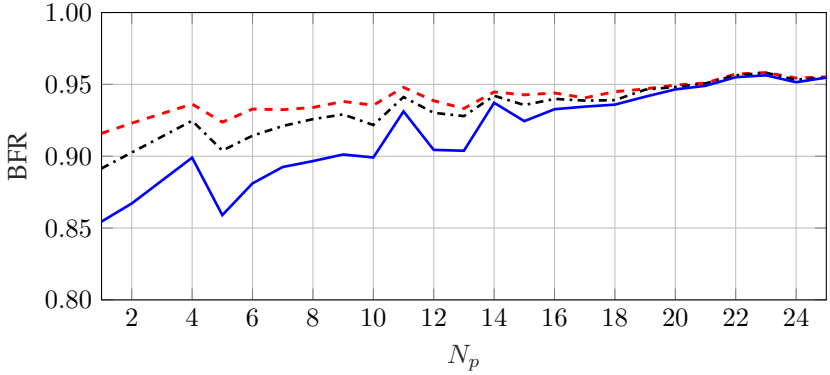


Figure 15: BFR vs N_p : $n_q = 1$, $n_q = 2$, $n_q = 3$.

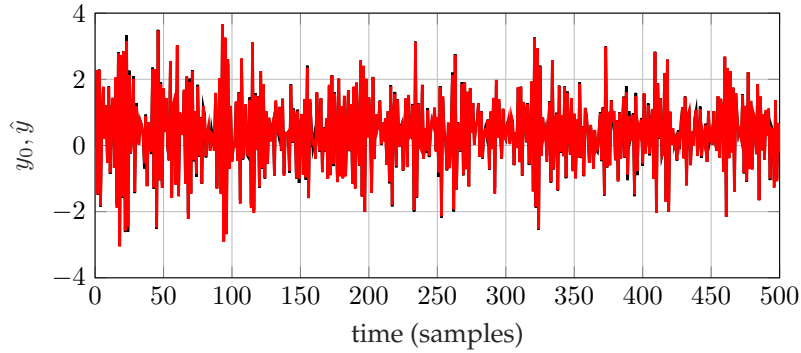
4.4.2 Identification of MIMO PWARX system

The second example is a Multiple-Input Multiple-Output (MIMO) PWARX data generating system (Breschi et al., 2016a), given by

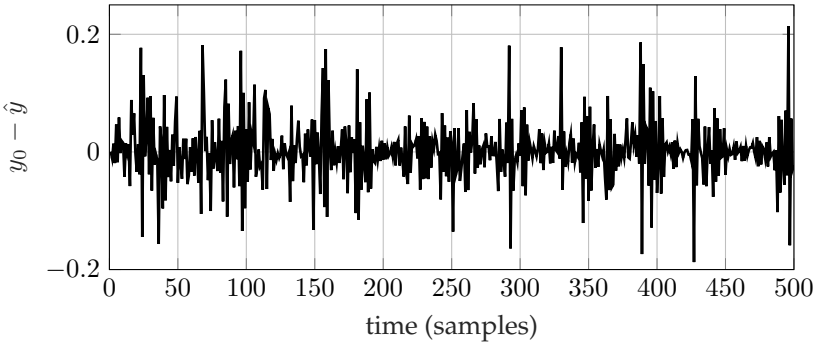
$$\begin{aligned}\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} = &\begin{bmatrix} -0.83 & 0.20 \\ 0.30 & -0.52 \end{bmatrix} \begin{bmatrix} y_1(k-1) \\ y_2(k-1) \end{bmatrix} + \begin{bmatrix} -0.34 & 0.45 \\ -0.30 & 0.24 \end{bmatrix} \begin{bmatrix} u_1(k-1) \\ u_2(k-1) \end{bmatrix} \\ &+ \begin{bmatrix} 0.20 \\ 0.15 \end{bmatrix} + \max \left\{ \begin{bmatrix} 0.20 & -0.90 \\ 0.10 & -0.42 \end{bmatrix} \begin{bmatrix} y_1(k-1) \\ y_2(k-1) \end{bmatrix} \right. \\ &\left. + \begin{bmatrix} 0.42 & 0.20 \\ 0.50 & 0.64 \end{bmatrix} \begin{bmatrix} u_1(k-1) \\ u_2(k-1) \end{bmatrix} + \begin{bmatrix} 0.40 \\ 0.30 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\} + e_{\mathrm{o}}(k), \end{aligned} \tag{4.21}$$

which is described by $\bar{s} = 4$ modes, based on the possible combinations generated by the sign of the max operator. The data is gathered by applying an input sequence $u(k)$ of length $N = 3000$, which is a white noise process with uniform distribution in the range $[-2 \;\; 2] \times [-2 \;\; 2]$. The noise signal $e_{\mathrm{o}} \in \mathbb{R}^2$ is a white Gaussian noise with covariance matrix $\Lambda_e = \begin{bmatrix} 2.5 \cdot 10^{-3} & 0 \\ 0 & 2.5 \cdot 10^{-3} \end{bmatrix}$. This corresponds to the SNR of 27 dB for the first output channel and 26 db for the second.

For the identification of system (4.21), Algorithm 11 is run with a prediction horizon $N_p = 10$, $s = \bar{s} = 4$ and with $n_q = 1$ i.e., for one iteration. This if followed by the off-line multicategory discrimination algorithm for partitioning of the regressor space, where algorithm (reported in section 4.3.2) is run with parameter $\kappa = 10^{-5}$. The obtained identification



(a) true (black) vs estimated (red)



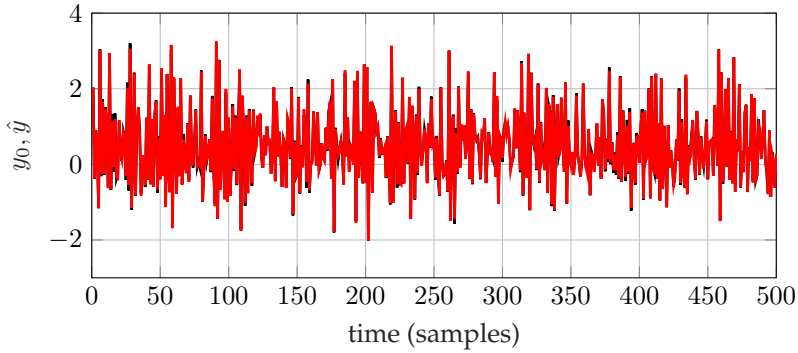
(b) Simulation error

Figure 16: Validation results for identification of MIMO-PWARX system, Output channel-1.

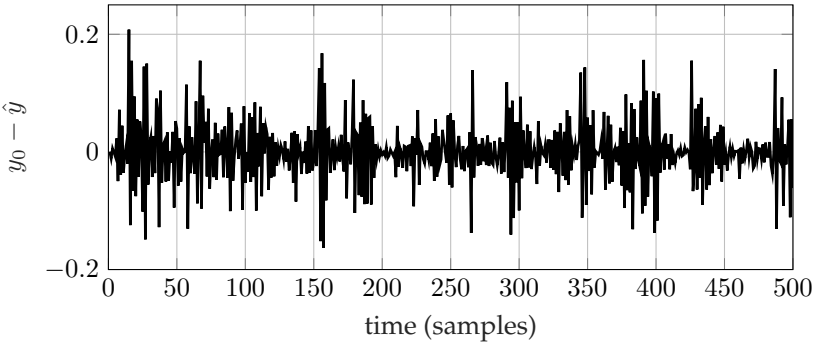
results in terms of BFR are reported in Table (10), achieved by using a noise-free validation dataset of length $N_{\text{val}} = 500$. The actual validation output y_{o} generated by the system, the simulated output $\hat{y}$ and the error $y_{\text{o}} - \hat{y}$ for the output channels 1 and 2 are shown in Figure 16, 17 respectively. The results show a good match between the actual output and the output simulated by the estimated model.

4.4.3 Identification of SISO LPV system

Next, we consider the identification of Linear Parameter Varying (LPV) models as a PieceWise Affine (PWA) regression problem. We consider a



(a) true (black) vs estimated (red)



(b) Output channel-2: Simulation error

Figure 17: Validation results for identification of MIMO-PWARX system, Output channel-2.

Table 10: Best Fit Rates on the validation dataset for MIMO PWARX system (4.21).

Output channel	BFR
BFR1	0.95
BFR2	0.94

single-input and single-output (SISO)-LPV ARX data generating system:

$$y(k)=a_1^{\circ}(p(k))y(k-1)+a_2^{\circ}(p(k))y(k-2)+b_1^{\circ}(p(k))u(k-1)+e(k).$$

The p-dependent coefficients $a_1^o(p(k))$, $a_2^o(p(k)$ and $b_1^o(p(k))$ are described by the nonlinear functions:

$$a_1^o(p(k)) = \begin{cases} -0.5, & \text{if } p(k) > 0.5 \\ -p(k), & \text{if } -0.5 \leq p(k) \leq 0.5 \\ 0.5, & \text{if } p(k) < -0.5 \end{cases}$$

$$a_2^o(p(k)) = p^3(k), \;\; b_1^o(p(k)) = \sin(\pi p(k)).$$

A training data and a validation dataset of length $N = 1000$ and $N_{\text{val}} = 2000$, respectively, are generated. The training input u and the scheduling signal p are considered as independent white-noise processes with uniform distribution $\mathcal{U}(-1, 1)$. The standard deviation of the noise e is 0.05, which corresponds to SNR of 20 dB. A PWA model with $s = 6$ modes is considered with prediction horizon $T = 6$. Each MIQP subproblem, solved with GPAD-B&B, contains 36 binary and 72 continuous variables, 144 inequality and 6 equality constraints. In the second stage, off-line multicategory discrimination algorithm is executed for partitioning the scheduling space, and the estimated model parameters are refined based on the computed partition, using simple least-squares for each sub-model.

Let $(z^*_{0|t}, \ldots, z^*_{T-1|t})$ be the optimal solution computed for the binary variables at time t, with T as horizon length. We can exploit the shifted binary values optimized at the previous step as the binary warm-start, in particular $z_{0|t+1} = z^*_{1|t}, \ldots, z_{T-2|t+1} = z^*_{T-1|t}$ by exploiting the framework proposed Section 3.5 of Chapter 3.

The Best fit rate calculated is 95% and the average time to solve the MIQP problem are shown in Table 11, where miqpGPAD refers to Algorithm 3 (from Chapter 2), miqpGPAD* refers to Algorithms 3 (from Chapter 2) properly coupled with Algorithm 10 (form Chapter 3). The results demonstrate the effectiveness of the warm-start approach,

Table 11: Average CPU time for training samples N=1000.

Solver	Time (ms)
GUROBI	259.38
miqpGPAD	125.06
miqpGPAD*	82.35

rendering better performance using the same QP solving algorithm.

4.5 Conclusion

A novel moving-horizon algorithm for *PieceWise Affine* regression has been described in this chapter. The proposed method combines the following advantages: (i) simultaneous choice of the model parameters and of the optimal sequence of active modes within a relatively short time horizon, and (ii) computational efficiency and iterative processing of the training samples. These features make the proposed approach particularly suitable for online applications, where a model needs to be updated once new data becomes available, without the need of storing past data history. The proposed algorithm is also employed for the identification of *Linear Parameter-Varying* (LPV) models. The binary warm-start framework proposed in Chapter 3 is exploited for efficient solution of the formulated MIQP problem, by properly adapting it for the GPAD-based MIQP solver presented in Chapter 2. The numerical results clearly demonstrate the utility of the proposed framework.

Chapter 5

Energy Disaggregation using Embedded Binary Quadratic Programming

5.1 Introduction

Extracting the power consumption profiles of individual household electric appliances from aggregated power measurements, commonly referred in the literature as *non-intrusive load monitoring* or *energy disaggregation*, is essential for consumers to become more aware of their energy usage habits, and for utilities and policy makers to design customized energy demand management strategies.

This chapter proposes new approaches for energy disaggregation using integer programming that are particularly suitable for embedded implementation on smart energy meters for a typical household where the number of appliances are moderate, exploiting the combinatorial solver proposed in Chapter 2. Moreover, the regularized moving-horizon PWA regression algorithm proposed in Chapter 4 is utilized in order to identify the dynamic model for each appliance.

Specifically, the problem of energy disaggregation is formulated in terms of three different *binary quadratic programming* (BQP) problems, derived from switching models describing the behaviour of the single appliances. The solution of the formulated BQP problems is com-

puted through a customized solver using *branch and bound* combined with *accelerated dual gradient projection* algorithm for solving quadratic programming (QP) relaxations. The proposed solver is library-free, the main computationally-intensive operations can be performed offline only once, while only basic arithmetic operations are performed online, as soon as new aggregate data become available. This makes the proposed energy disaggregation algorithms particularly suitable for real-time embedded implementation directly on commercial smart meters, due to its limited memory and CPU requirements.

5.1.1 Motivation

Retrieving residential power consumption at the single-appliance level is a valuable source of information for (i) consumers to make them more aware of their energy usage habits, receive automated feedback and personalized recommendations for energy savings, detect malfunctioning or the presence of energy-inefficient devices, as well as for (ii) utilities and policy makers to segment the market, facilitate forecasting, design and assess the impact of customized energy demand management strategies (Armel et al., 2013).

Energy monitoring or appliance load monitoring (ALM) methods are employed to gauge the energy consumption of the individual appliance/device of whole building/premise. Two possible approaches can be followed to acquire information on end-use energy consumption:

- *Intrusive load monitoring* (ILM). This is a *hardware* strategy, which consists in using smart appliances connected to a home network or attaching a measurement device to each appliance in the house.

- *Non-intrusive load monitoring* (NILM). This is a *software* strategy, which consists in using data-processing algorithms (commonly referred in the literature as energy disaggregation methods) to decompose the aggregate power demand gathered from a single-point smart meter into the individual contribution of each appliance.

The advantages of the NILM approach with respect to ILM are clearly the reduction of intrusiveness into consumers' houses and lower costs for installation, maintenance and replacement of the monitoring system.

Since the pioneering work by Hart on NILM (Hart, 1992), several algorithms for energy disaggregation have been developed in the last dec-

ades (see the survey papers (Zeifman and Roth, 2011; Zoha et al., 2012; Faustine et al., 2017) for a literature overview). State-of-the-art NILM methods can be classified into two main categories: pattern recognition (Baranski and Voss, 2004; Srinivasan et al., 2006; Ducange et al., 2014; Tabatabaei et al., 2017; Cominola et al., 2017; Gillis and Morsi, 2017; Singhal et al., 2018) and optimization-based approaches (Suzuki et al., 2008; Figueiredo et al., 2013; Piga et al., 2016; Rahimpour et al., 2017; Bhotto et al., 2017; Wittmann et al., 2018). Algorithms belonging to the first class try to extract features (*e.g.*, active and reactive power, harmonics, high-frequency wavelets, etc.) from the aggregate consumption, and then classify events based on the extracted features. Algorithms belonging to the second class formulate energy disaggregation as an optimization problem and find the set of devices (and their configuration) that best fits the measured aggregate power. As simply looking for the best fit of the aggregate power might lead to poor results in the case of appliances with similar power rating levels, data pre-processing and additional constraints are typically needed.

We focus particularly on the NILM techniques based on integer programming. An integer programming based approach is presented in (Suzuki et al., 2008) for small-scale problems, and for the cases where some appliance has multiple modes. One of the challenge for such approaches is to distinguish between appliances with similar or overlapping load signature. Aided linear integer programming (ALIP) based approach presented in (Bhotto et al., 2017) is with correction based on a state diagram, median filtering, and linear-programming-based refinement. Mixed-integer linear programming (MILP) based NILM approach presented in (Wittmann et al., 2018) introduces a new set of integer linear constraints. However, these integer programming based approaches for energy disaggregation available in literature (Suzuki et al., 2008; Bhotto et al., 2017; Wittmann et al., 2018) are implemented using "desktop" computers running software packages like CPLEX (IBM, Inc., 2014), MOSEK (MOSEK ApS, 2015), and GUROBI (Gurobi Optimization, Inc., 2014). Consequently, these algorithms are not tailored to a real-time embedded implementation, characterized by restrictions on available memory, computing power, and time.

Recently, embedded control community have been doing notable efforts to tailor mixed-integer quadratic programming (MIQP) solvers for embedded applications. Several solution methods have been presented using the Branch and Bound (B&B) algorithm (Floudas, 1995) combined with QP solving algorithms including interior-point method (Frick

et al., 2015), active-set for solving nonnegative least square (NNLS) (Bemporad, 2015b), accelerated dual gradient projection (GPAD) presented in Chapter 2, Alternating Direction Method of Multipliers (ADMM) algorithm and the operator splitting quadratic program (OSQP) solver presented in Chapter 2, and active-set for solving NNLS with proximal-point iterations presented in Chapter 3.

5.1.2 Contribution

This chapter presents novel optimization-based approaches tailored for embedded implementation such as smart energy meters, for a typical household where the number of devices are moderate. We provide approaches to overcome the problem of distinguishing between different devices with similar operating modes (or combinations of them). Specifically, the contribution is twofold: First, we present three algorithms for energy disaggregation based on *binary quadratic programming* (BQP). The first two algorithms only require to have a prior knowledge on the power rating levels of each appliance at each operating mode, while the third approach employs switching dynamical models to describe the typical power consumption profiles of each appliance. These models can be estimated from disaggregated data gathered over a short intrusive period, using techniques for identification of hybrid dynamical systems, such as moving-horizon methods (Ferrari-Trecate et al., 2002) or clustering-based approaches (Ferrari-Trecate et al., 2003; Breschi et al., 2016b). Particularly, we employ the moving-horizon algorithm proposed in Chapter 4.

Secondly, we propose a customized solver for the formulated BQP energy disaggregation problems. This solver, named bqpGPAD, combines *branch and bound* (B&B) with *Accelerated Dual Gradient Projection*[1] (GPAD), which is a special case of the Mixed-Integer Quadratic Programming (MIQP) solver presented in Chapter 2. In the proposed BQP formulations, the main computationally intensive operations, such as matrix factorizations and preconditioning, can be performed offline only once, while only basic arithmetic operations are performed online, as soon as a new data become available. As a result, the proposed energy disaggregation algorithms are particularly suitable for real-time embedded implementation in commercial smart meters.

[1] Accelerated dual gradient projection, originally introduced in (Patrinos and Bemporad, 2014) for embedded model predictive control, is the Nesterov's accelerated method (Nesterov, 1983) applied to dual quadratic programming problems.

Finally, the proposed disaggregation algorithms are tested against the AMPds benchmark dataset (Makonin et al., 2013), which contains energy consumption recorded over a year in a single house located in Canada at one-minute time resolution. For a comparative analysis, two types of energy disaggregation problems are considered specifically for appliances with two operating modes, and further with multiple operating modes. The numerical results are compared with convex optimization-based approach presented in (Piga et al., 2016). A comparison in terms of CPU time between the developed bqpGPAD solver and the commercial software package for mixed-integer optimization GUROBI (Gurobi Optimization, Inc., 2014) and CPLEX (IBM, Inc., 2014) is provided.

5.1.3 Outline

The outline of this chapter is as follows. The formulation of energy disaggregation problem is presented in Section 5.2. The proposed disaggregation approaches are described in Section 5.3. This is followed by a customized BQP solver using GPAD-based Branch and Bound presented in Section 5.4. The proposed algorithms are validated against a standard benchmark dataset and the numerical results are reported in Section 5.5. The conclusion and final remarks are given in Section 5.6.

5.2 Problem setting

5.2.1 Single appliance modelling

Consider a household with n different electric appliances which draw power from the main electric line. Under the modelling assumption that each appliance has $s \in \mathbb{N}$ operating modes[2], we denote with $P_i^{(j)}$ the power rating (in Watts) of the i-th appliance at the j-th operating mode. Let us introduce the binary variable $\delta_i^{(j)}(t) \in \{0, 1\}$ associated to the j-th operating mode of the i-th appliance at time t. Specifically, $\delta_i^{(j)}(t) = 1$ if the i-th appliance is at mode j at time t; $\delta_i^{(j)}(t) = 0$ otherwise. Hence, the

[2]The disaggregation approach discussed in the chapter can be applied to appliances with different number of operating modes. However, we consider the case where all the appliances have the same number s of operating modes just for simplicity of notation.

power consumed by the i-th appliance at time t can be represented as

$$y_i(t) = \begin{bmatrix} P_i^{(1)}(t) & \cdots & P_i^{(s)}(t) \end{bmatrix} \begin{bmatrix} \delta_i^{(1)}(t) \\ \vdots \\ \delta_i^{(s)}(t) \end{bmatrix} + e_i(t), \tag{5.1a}$$

$$\delta_i^{(j)}(t) \in \{0,1\},\ i = 1,\dots,n,\ j = 1,\dots,s, \tag{5.1b}$$

$$\sum_{j=1}^{s} \delta_i^{(j)}(t) = 1,\ i = 1,\dots,n, \tag{5.1c}$$

where $e_i(t)$ represents an intrinsic modeling error, and the constraint (5.1c) is due the fact that each device can operate in one and only one mode at each time t. Note that $P_i^{(j)}$ can be time varying to take into account transient behaviour.

Let mode $j = 1$ represent the OFF state for each device, with associated power rate $P_i^{(1)} = 0$ W. It can be easily shown that as $P_i^{(1)} = 0$, the power consumption model (5.1) can be equivalently represented as

$$y_i(t) = \begin{bmatrix} P_i^{(2)}(t) & \cdots & P_i^{(s)}(t) \end{bmatrix} \begin{bmatrix} \delta_i^{(2)}(t) \\ \vdots \\ \delta_i^{(s)}(t) \end{bmatrix} + e_i(t), \tag{5.2a}$$

$$\delta_i^{(j)}(t) \in \{0,1\},\ i = 1,\dots,n,\ j = 2,\dots,s, \tag{5.2b}$$

$$\sum_{j=2}^{s} \delta_i^{(j)}(t) \leq 1,\ i = 1,\dots,n. \tag{5.2c}$$

This allows us to reduce the number of binary variables in the optimization problems formulated in the following sections.

5.2.2 Energy disaggregation problem

Given the aggregated power $y(t)$:

$$y(t) = \sum_{i=1}^{n} y_i(t) + e(t), \tag{5.3}$$

the objective of energy disaggregation is to extract the power consumption profiles $y_i(t)$ of each appliance. In (5.3), $e(t)$ takes into account unmodelled devices and measurement noise on the aggregated power reading.

5.3 Disaggregation Algorithms

In this section, we present three approaches for energy disaggregation based on *Binary Quadratic Programming* (BQP). The first two approaches make use of switching static models to describe the power consumption profiles of each appliance, while the third one is based on switching linear autoregressive models.

5.3.1 Approach A1: Binary Quadratic Programming

A simple approach to reconstruct the power demand $y_i(t)$ of each appliance from the aggregated power readings $y(t)$ over a timing window of length $T \in \mathbb{N}$ is to compute the time-varying model parameters $\delta_i^{(2)}(t), \ldots, \delta_i^{(s)}(t)$ (with $i = 1, \ldots, n$ and $t = 1, \ldots, T$) by solving the binary least-square problem

$$\min_{\{\delta_i^{(2)}(t), \ldots, \delta_i^{(s)}(t)\}_{i,t=1}^{n,T}} \quad \sum_{t=1}^{T} \left(y(t) - \sum_{i=1}^{n} \hat{y}_i(t, \delta_i(t)) \right)^2 \tag{5.4a}$$

$$\text{s.t.} \quad \sum_{j=2}^{s} \delta_i^{(j)}(t) \leq 1, \; i = 1, \ldots, n \tag{5.4b}$$

$$\delta_i^{(j)}(t) \in \{0, 1\}, \; i = 1, \ldots, n, \; j = 2, \ldots, s, \tag{5.4c}$$

with

$$\delta_i(t) = [\delta_i^{(2)}(t) \;\; \cdots \;\; \delta_i^{(s)}(t)], \tag{5.4d}$$

and

$$\hat{y}_i(t, \delta_i(t)) = \begin{bmatrix} P_i^{(2)} & \cdots & P_i^{(s)} \end{bmatrix} \begin{bmatrix} \delta_i^{(2)}(t) \\ \vdots \\ \delta_i^{(s)}(t) \end{bmatrix}. \tag{5.4e}$$

A rough knowledge of the terms $P_i^{(j)}$ (i.e., the steady-state power demand of each appliance at each operating mode) is supposed to be available. For instance, $P_i^{(j)}$ can be evaluated from "snapshots" of energy use patters through k-means clustering (Likas et al., 2003) or by simple visual inspection.

The main limitation in problem (5.4) is that appliances with similar power ratings (or similar combinations of them[3]) may not be distinguished, leading to poor disaggregation results. In order to overcome this limitation, in the following paragraph we propose a modification of problem (5.4) by exploiting the time-domain information that electric appliances rarely change their operating mode over time.

5.3.2 Approach A2: Regularized Binary Quadratic Programming

In order to overcome the problem due to the presence of different devices with similar operating modes (or combinations of them), a regularization term can be added to (5.4) to penalize the change of the binary state $\delta_i(t)$ over time. This leads to

$$\min_{\{\delta_i^{(2)}(t),\dots,\delta_i^{(s)}(t)\}_{i,t=1}^{n,T}} \sum_{t=1}^{T}\left(y(t)-\sum_{i=1}^{n}\hat{y}_i(t,\delta_i)\right)^2 + \tag{5.5a}$$

$$+\sum_{i=1}^{n}\sum_{t=1}^{T}\left\|\begin{bmatrix}\gamma_i^{(2)}\\ \vdots \\ \gamma_i^{(s)}\end{bmatrix}\odot\begin{bmatrix}\delta_i^{(2)}(t)-\delta_i^{(2)}(t-1)\\ \vdots \\ \delta_i^{(s)}(t)-\delta_i^{(s)}(t-1)\end{bmatrix}\right\|_2^2 \tag{5.5b}$$

$$\text{s.t.} \quad \sum_{j=2}^{s}\delta_i^{(j)}(t)\le 1,\ i=1,\dots,n, \tag{5.5c}$$

$$\delta_i^{(j)}(t)\in\{0,1\},\ i=1,\dots,n,\ j=2,\dots,s, \tag{5.5d}$$

where $\delta_i^{(j)}(0)$ is a guess of an the initial state, and $\gamma_i^{(j)}\in\mathbb{R}$ are positive weighting parameters chosen through cross-validation and penalizing the time variation of the binary variables $\delta_i^{(j)}$. Note that, since the parameters $\gamma_i^{(j)}$ depend on the index i, appliances which rarely changes their

[3]Suppose that the appliances A, B and C consume 600 W, 400 W and 210 W, respectively, in one of their operating points. Then, the behaviour of appliance A can be confused with the combination of appliances B and C.

operating modes over time should be associated to high-value penalization parameters $\gamma_i^{(j)}$. Penalizing the norm of the difference between two consecutive parameters $\delta_i^{(j)}(t-1)$ and $\delta_i^{(j)}(t)$ enforces the binary $\delta_i^{(j)}$ to be piecewise constant over time. This is a reasonable assumption for many electric devices, provided that data are observed at small sampling time (say, less than 1 minute).

5.3.3 Approach A3: Binary Quadratic Programming with S-AR models

Both methods described in the previous paragraph are based on (switching) static models, and thus only information on the steady-state behaviour is exploited. An alternative approach to accurately distinguish appliances with similar consumption ratings consists in exploiting information coming from transient behaviour. This requires to describe the power consumption patterns of individual appliances in terms of switching dynamical models, instead of static one.

The approach described in this paragraph employs *Switching Auto-Regressive* (S-AR) models to describe the power consumption profile $y_i(t)$ of the single appliance. More specifically, $y_i(t)$ is given by

$$y_i(t) = \begin{cases} \Theta_{i,1}^{'} \begin{bmatrix} 1 \\ x_i(t) \end{bmatrix} & \text{if } \delta_i^{(1)}(t) = 1, \\ \vdots & \\ \Theta_{i,s}^{'} \begin{bmatrix} 1 \\ x_i(t) \end{bmatrix} & \text{if } \delta_i^{(s)}(t) = 1, \end{cases} \tag{5.6a}$$

where $x_i(t)$ denotes the regressor vector containing the past values of the outputs, i.e.,

$$x_i(t) = \left[y_i(t-1), \ldots, y_i(t-n_\mathrm{a}) \right]^{'}, \tag{5.6b}$$

and $\Theta_{i,j}$ is a set of parameters describing the behavior of the i-th appliance at the j-th operating mode. Using disaggregated training data collected over a short intrusive period, the parameters $\Theta_{i,j}$ can be estimated using available algorithms for identification of hybrid dynamical systems. In this work, moving-horizon approach proposed in Chapter 4 will be used.

Once the model parameters $\Theta_{i,j}$ are estimated, the energy disaggregation problem can be tackled by solving iteratively and at each time t, the

binary quadratic program:

$$\min_{\{\delta_i^{(j)}(t)\}_{i,j=1}^{n,s}} \left\| y(t) - \sum_{i=1}^{n}\sum_{j=1}^{s} \Theta_{i,j}^{'} \begin{bmatrix} 1 \\ \hat{x}_i(t) \end{bmatrix} \delta_i^{(j)}(t) \right\|_2^2, \tag{5.7a}$$

$$\text{s.t. } \delta_i^{(j)}(t) \in \{0,1\}, \ \sum_{j=1}^{s} \delta_i^{(j)}(t) = 1. \tag{5.7b}$$

$\hat{x}_i(t)$ is the estimated regressor vector[4] defined as

$$\hat{x}_i(t) = \left[\hat{y}_i(t-1), \ldots, \hat{y}_i(t-n_{\mathrm{a}})\right]^{'},$$

and $\hat{y}_i(t)$ is the estimate of the disaggregated power for the i-th appliance given by

$$\hat{y}_i(t) = \Theta_{i,j^*}^{'} \begin{bmatrix} 1 \\ \hat{x}_i(t) \end{bmatrix}, \tag{5.8}$$

where j^* is extracted from the solution of problem (5.7) and it represents the active operative mode of the i-th appliance at time t, i.e.,

$$j^* : \delta_i^{(j^*)}(t) = 1.$$

The estimated disaggregated power $\hat{y}_i(t)$ is then used at time $t+1$ to construct the regressor $\hat{x}_i(t+1)$. Because of the iterative nature of this third disaggregation approach, including a term in (5.7) penalizing the time variation of the binary state $\delta_i(t)$ is not straightforward.

5.4 Solving BQP problems using GPAD-based Branch and Bound

In this section, we describe a customized GPAD-based branch and bound algorithm (named bqpGPAD) to solve the BQP energy disaggregation problems formulated in the previous section. This solver is a special case of the approach proposed in Chapter 2 for embedded *mixed-integer quadratic programming*. For brevity, we describe the numerical algorithm focusing only on problem (5.4). The same algorithm can be easily adapted to problems (5.5) and (5.7).

[4] The true regressor $x_i(t)$ in (5.6b) can not be constructed as it depends on disaggregated power consumption y_i. This information is not available at the disaggregation stage.

In order to rewrite problem (5.4) in terms of standard BQP formulation, let us introduce the binary state vector $\delta(t) \in \{0,1\}^{(s-1)\cdot n}$ by concatenating the state vectors $\delta_i(t)$ (for $i = 1, \ldots, n$) defined in (5.4d) and characterizing the operating mode of the i-the device at time t, i.e.,

$$\delta(t) = [\delta_1'(t) \ \cdots \ \delta_n'(t)]'. \tag{5.9}$$

Let $\bar{\delta}$ denote the binary vector constructed by stacking $\delta(t)$ for $t = 1, \ldots, T$, i.e.,

$$\bar{\delta} = [\delta^{'}(1) \ \cdots \ \delta^{'}(T)]^{'},$$

with $\bar{\delta} \in \{0,1\}^p$, where $p = (s-1) \cdot n \cdot T$.

Problem (5.4) can be then rewritten as a standard BQP problem

$$\begin{aligned} \min_{\bar{\delta}} \quad & V(\bar{\delta}) \triangleq \frac{1}{2}\bar{\delta}^{'} Q \bar{\delta} + c^{'}\bar{\delta} && \text{(5.10a)} \\ \text{s.t.} \quad & A\bar{\delta} \leq u && \text{(5.10b)} \\ & \bar{\delta}_i \in \{\bar{\ell}_i, \bar{u}_i\}, \ i = 1, \ldots, p && \text{(5.10c)} \end{aligned}$$

where the Hessian $Q \in \mathbb{R}^{p \times p}$ and the vector $c \in \mathbb{R}^p$ are properly defined based on the quadratic cost (5.4a), $\bar{\ell}_i = 0$, $\bar{u}_i = 1$, $u = \mathbf{1}_T$, and $A \in \mathbb{R}^{T \times p}$ is given by

$$\begin{aligned} A\bar{\delta} &= \text{diag}(\bar{\delta}') \\ &= \begin{bmatrix} \delta(1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \delta(T) \end{bmatrix} \\ &= \underbrace{(I_T \otimes \mathbf{1}'_{(s-1)\cdot n})}_{A} \cdot \bar{\delta}. \end{aligned}$$

In the following sections, we show how to solve the BQP problem (5.10) through the GPAD-based B&B numerical solver bqpGPAD. For completeness, we recall the framework presented in Chapter 2: First, in Section 5.4.1, we refer the GPAD algorithm from Section 2.2 for solving the relaxed QP problems arising in branch and bound. Then, in Section 5.4.2, we describe how to employ GPAD in the B&B framework, based on the ideas presented in Section 2.3.

5.4.1 Solving relaxed QP problems

In order to relax the BQP problem (5.10), binary constraints (5.10c) are replaced by

$$\bar{\ell}_i \leq \bar{\delta}_i \leq \bar{u}_i,\ i = 1, \ldots, p, \tag{5.11}$$

leading to the following relaxed QP problem

$$\begin{aligned} \min_{\bar{\delta}} \quad & V(\bar{\delta}) \triangleq \frac{1}{2}\bar{\delta}' Q \bar{\delta} + c' \bar{\delta} && \text{(5.12a)} \\ \text{s.t.} \quad & A\bar{\delta} \leq u, && \text{(5.12b)} \\ & \bar{\ell} \leq \bar{\delta} \leq \bar{u}, && \text{(5.12c)} \end{aligned}$$

with $\bar{\ell} = \mathbf{0}_p$ and $\bar{u} = \mathbf{1}_p$.

The solution of the QP problem (5.12) is computed through the GPAD method outlined in Algorithm 12. The main idea behind the GPAD algorithm, originally proposed in (Patrinos and Bemporad, 2014), is to extend the formulation of Nesterov's fast gradient method (Nesterov, 1983) on the dual QP problem:

$$\begin{aligned} \max_{\lambda} \ & \Psi(\lambda) \triangleq -\frac{1}{2}\lambda' \mathcal{A} Q^{-1} \mathcal{A}' \lambda - d'\lambda - \frac{1}{2} c' Q^{-1} c \\ \text{s.t.} \quad & \lambda \geq 0 \end{aligned} \tag{5.13a}$$

where

$$\mathcal{A} = \begin{bmatrix} I \\ -I \\ A \end{bmatrix},\ \mathcal{B} = \begin{bmatrix} \bar{u} \\ -\bar{\ell} \\ u \end{bmatrix},\ d = \mathcal{B} + \mathcal{A} Q^{-1} c, \tag{5.13b}$$

and $\lambda \in \mathbb{R}^{2p+T}$ is the dual vector, which is decomposed as

$$\lambda = \begin{bmatrix} \lambda_{\bar{u}} \\ \lambda_{\bar{\ell}} \\ \lambda_u \end{bmatrix}, \quad \lambda_{\bar{u}}, \lambda_{\bar{\ell}} \in \mathbb{R}^p, \quad \lambda_u \in \mathbb{R}^T. \tag{5.13c}$$

At Step 2 of Algorithm 12, the dual vector λ is first initialized. At each iteration k, the extrapolation factor $\beta^{(k)}$ is computed (Step 5) and used in Step 6, where Nesterov's acceleration is applied to the dual vector λ. Finally, λ is updated through projected gradient (Steps 7 - 9). The algorithm is iterated until the primal feasibility criterion

$$\bar{s}_j^{(k)} \leq \frac{1}{L}\epsilon_G,\ \forall j = 1, \ldots, 2p + T \tag{5.14a}$$

and the optimality criterion

$$-\bar{w}^{(k)'} \bar{s}^{(k)} \leq \frac{1}{L}\epsilon_V \tag{5.14b}$$

are satisfied simultaneously, where $\epsilon_G > 0$ is a feasibility tolerance, and $\epsilon_V \geq 0$ is an optimality tolerance. Condition (5.14b) is derived from the relation of duality gap $V(\bar{\delta}^{(k)}) - V^* \leq V(\bar{\delta}^{(k)}) - \Psi(\bar{w}^{(k)}) = -\bar{w}^{(k)'}\bar{s}^{(k)} \leq \frac{1}{L}\epsilon_V$ (see (Patrinos and Bemporad, 2014) for details).

The algorithm returns the optimal dual vector λ^* and the retrieved primal solution $\bar{\delta}^*$ and the optimal cost V^*.

Algorithm 12 Accelerated dual gradient projection method to solve relaxed QP problem (5.12).

Input: matrices Q, $\mathcal{A}$; vectors c, $\mathcal{B}$;

$H \leftarrow \mathcal{A}Q^{-1}\mathcal{A}'$, $L \leftarrow \lambda_{max}(H)$, $\mathcal{A}_L \leftarrow \frac{1}{L}\mathcal{A}$, $\mathcal{B}_L \leftarrow \frac{1}{L}\mathcal{B}$;
$\lambda_0, \lambda_{-1} \leftarrow 0$;
$k \leftarrow 0$;
repeat
  $\beta^{(k)} \leftarrow \max\left\{\frac{k-1}{k+2}, 0\right\}$;
  $\bar{w}^{(k)} \leftarrow \lambda^{(k)} + \beta^{(k)}\left(\lambda^{(k)} - \lambda^{(k-1)}\right)$;
  $\bar{\delta}^{(k)} \leftarrow -Q^{-1}\mathcal{A}'\bar{w}^{(k)} - Q^{-1}c$;
  $\bar{s}^{(k)} \leftarrow \left(\mathcal{A}_L\bar{\delta}^{(k)} - \mathcal{B}_L\right)$;
  $\lambda^{(k+1)} \leftarrow \max\{\bar{w}^{(k)} + \bar{s}^{(k)}, 0\}$;
  $k \leftarrow k + 1$;
until convergence;
$\lambda^* \leftarrow \bar{w}^{(k)}$, $\bar{\delta}^* \leftarrow \bar{\delta}^{(k)}$, $V^* \leftarrow V(\bar{\delta}^*)$;

Output: dual solution λ^*, primal solution $\bar{\delta}^*$, optimal cost V^*.

Preconditioning

It is well known that performance of the first-order optimization methods is sensitive to the problem of conditioning. In order to address this issue and to better conditioning the QP problem (5.12), we adopt the following Jacobi diagonal scaling (Bertsekas, 2009)

$$\mathcal{A}_j \leftarrow M_j\mathcal{A}_j, \quad \mathcal{B}_j \leftarrow M_j\mathcal{B}_j, \quad j = 1, \ldots, 2p + T, \tag{5.15}$$

with $M_j \triangleq \frac{1}{\sqrt{\mathcal{A}_j Q^{-1}\mathcal{A}_j'}}$.

Restart

Accelerated gradient methods exhibit non-monotonic behavior during iterations, and ripples in the sequence of cost function are often observed during numerical iterations. Convergence of the method can be

improved by using restarting heuristics. In this work, we apply the gradient-based adaptive restart ideas of (O' Donoghue and Candès, 2015; Giselsson and Boyd, 2014) to the dual problem (5.13). Specifically, the sequence of scalars β_k in Step 5 of Algorithm 12 is restarted from 0 whenever the following condition

$$-L\bar{s}^{(k)'}\left(\lambda^{(k+1)}-\lambda^{(k)}\right)>0, \tag{5.16}$$

is satisfied. Note that evaluating condition (5.16) is computationally inexpensive as it depends on already computed quantities.

Infeasibility detection

During the execution of B&B (discussed in Section 5.4.2), QP relaxation with some of the integer combination may be infeasible, a frequently occurring case for B&B. In this case, the dual cost $\Psi(\lambda^{(k)})$ tends to $+\infty$.

The criteria to detect infeasibility are reported in Algorithm 13, which applies Farkas Lemma (Rockafellar, 1970, p. 201) to detect infeasibility of QP problem (5.12), within a tolerance $\epsilon_I > 0$ (see Lemma 1 in Section 2.2.4).

Algorithm 13 Infeasibility detection

1: $\alpha_k \leftarrow \|\bar{w}^{(k)}\|_\infty$;
2: **if** $\|\mathcal{A}'\bar{w}^{(k)}\|_\infty \leq \epsilon_I \alpha_k$ **and** $d'\bar{w}^{(k)} < -\epsilon_I\alpha_k$ **then**
3: **stop** (primal problem is infeasible)
4: **end if**

Early stopping criterion

Termination criteria based on dual cost is particularly useful for QP solving algorithms within the B&B setting. Let V_0 be the best known integer-feasible cost, which acts as an upper bound for the optimal cost V^*. Since $V(\bar{\delta}^{(k)}) \geq \Psi(\lambda^{(k)})$, the GPAD Algorithm 12 can be stopped prematurely if the condition

$$\Psi(\lambda^{(k)}) \geq V_0, \tag{5.17}$$

is satisfied. Indeed, if condition (5.17) holds, we can prune that particular branch of the binary tree, without the need to further iterate Algorithm 12 until convergence. This part is also discussed in the following section describing the B&B algorithm.

5.4.2 Branch and Bound for BQP problems

This section summarizes the B&B algorithm (equipped with the GPAD method described in Section 5.4.1) used to solve the BQP problem (5.10).

The B&B algorithm proceeds in a binary tree fashion, where each node of the tree represent a relaxed QP problem. The root node (origin of a tree) is represented by the relaxed QP problem (5.12), where all the $\{1,\ldots,p\}$ integrality constraints in (5.10c) are replaced by (5.11). The tree evolves by branching operations, that pick a variable $\bar{\delta}_i$ and change the corresponding relaxed inequality $\bar{\ell}_i \leq \bar{\delta}_i \leq \bar{u}_i$ into two equalities $\bar{\delta}_i = \bar{\ell}_i$ and $\bar{\delta}_i = \bar{u}_i$, thus creating two children nodes. This is continued till all binary variables $\bar{\delta}_i$, with $i = \{1,\ldots,p\}$, are branched upon in the same fashion.

Hence, execution of B&B relies on the solution of a relaxed QP subproblem at each node, where the integrality constraints (5.10c) are relaxed by

$$\bar{\delta}_{\mathcal{I}_{\bar{u}}} = \bar{u}_{\mathcal{I}_{\bar{u}}}, \tag{5.18a}$$
$$\bar{\delta}_{\mathcal{I}_{\bar{\ell}}} = \bar{\ell}_{\mathcal{I}_{\bar{\ell}}}, \tag{5.18b}$$
$$\bar{\ell}_{\mathcal{J}} \leq \bar{\delta}_{\mathcal{J}} \leq \bar{u}_{\mathcal{J}}, \tag{5.18c}$$

with $\mathcal{I}_{\bar{u}}$, $\mathcal{I}_{\bar{\ell}}$ and $\mathcal{J}$ being disjoint sets of integers (i.e., $\mathcal{I}_{\bar{u}} \cap \mathcal{I}_{\bar{\ell}} \cap \mathcal{J} = \emptyset$) such that $\mathcal{I}_{\bar{u}} \cup \mathcal{I}_{\bar{\ell}} \cup \mathcal{J} = \{1,\ldots,p\}$. The sets $\mathcal{I}_{\bar{u}}$, $\mathcal{I}_{\bar{\ell}}$ and $\mathcal{J}$ are uniquely defined during the branching operations. Thus, the following general QP problem should be solved at each node of the binary tree:

$$\min_{\bar{\delta}} \quad V(\bar{\delta}) \triangleq \frac{1}{2}\bar{\delta}' Q \bar{\delta} + c' \bar{\delta} \tag{5.19a}$$
$$\text{s.t.} \quad A\bar{\delta} \leq u \tag{5.19b}$$
$$\bar{\ell}_{\mathcal{J}} \leq \bar{\delta}_{\mathcal{J}} \leq \bar{u}_{\mathcal{J}}, \tag{5.19c}$$
$$\bar{\delta}_{\mathcal{I}_{\bar{u}}} = \bar{u}_{\mathcal{I}_{\bar{u}}}, \;\; \bar{\delta}_{\mathcal{I}_{\bar{\ell}}} = \bar{\ell}_{\mathcal{I}_{\bar{\ell}}}. \tag{5.19d}$$

Algorithm 14 outlines the B&B solver for the BQP problem (5.10) in which the relaxed QP subproblems (5.19) are solved using the GPAD algorithm discussed in Section 5.4.1.

At Step 1, the root node is created by relaxing all integrality constraints (5.10c) into (5.11), that is equivalent to $\mathcal{I}_{\bar{\ell}} = \mathcal{I}_{\bar{u}} = \emptyset$, and $\mathcal{J} = \{1,\ldots,p\}$ in (5.18). The tuple $\mathcal{T}$ holds the sets of indices $\mathcal{I}_{\bar{\ell}}, \mathcal{I}_{\bar{u}}$ which uniquely identify the QP subproblems of form (5.19). The set $\mathcal{S}$ stacks the tuples $\mathcal{T}$ in a specific order of pending relaxations to be solved

Algorithm 14 BQP solver based on GPAD to solve (5.10)

Input: BQP problem matrices/vectors Q, A, u, $\bar{\ell}$, $\bar{u}$; feasibility tolerance $\epsilon \geq 0$.

1. **set** $\mathcal{J} \leftarrow \{1, \ldots, p\}$; $\mathcal{I}_{\bar{\ell}} \leftarrow \emptyset$; $\mathcal{I}_{\bar{u}} \leftarrow \emptyset$; $\mathcal{T} \leftarrow \{\mathcal{I}_{\bar{\ell}}, \mathcal{I}_{\bar{u}}\}$; $\mathcal{S} \leftarrow \{\mathcal{T}\}$; $V_0 \leftarrow +\infty$;
2. **while** $\mathcal{S} \neq \emptyset$ **do**:
   - 2.1. $\{\mathcal{I}_{\bar{\ell}}, \mathcal{I}_{\bar{u}}\} \leftarrow$ last element $\mathcal{T}$ of $\mathcal{S}$; $\mathcal{S} \leftarrow \mathcal{S} \setminus \{\mathcal{T}\}$;
   - 2.2. **solve** (5.19) using Algorithm 12, under early stopping criterion (5.17);
   - 2.3. **if** the outputs $\bar{\delta}^*$ and V^* are returned by Algorithm 12, **then**
     - 2.3.1. **if** $\mathcal{I}_{\bar{\ell}} \cup \mathcal{I}_{\bar{u}} = \{1, \ldots, p\}$ **or** $\bar{\delta}_i^* \in \{\bar{\ell}_i, \bar{u}_i\}, \forall i \in \mathcal{J}$ **then** $V_0 \leftarrow V^*$, $\zeta^* \leftarrow \bar{\delta}^*$; **otherwise**
       - 2.2.3.1. $j \leftarrow \arg\min_{i \in \mathcal{J}} \left| \bar{\delta}_i^* - \frac{\bar{\ell}_i + \bar{u}_i}{2} \right|$;
       - 2.2.3.2. $\mathcal{J} \leftarrow \mathcal{J} \setminus j$;
       - 2.2.3.3. $\mathcal{T}_0 \leftarrow \{\mathcal{I}_{\bar{\ell}} \cup \{j\}, \mathcal{I}_{\bar{u}}\}$; $\mathcal{T}_1 \leftarrow \{\mathcal{I}_{\bar{\ell}}, \mathcal{I}_{\bar{u}} \cup \{j\}\}$;
       - 2.2.3.4. **if** $\bar{\delta}_j^* \leq \frac{\bar{\ell}_i + \bar{u}_i}{2}$ **then** append $\{\mathcal{T}_1, \mathcal{T}_0\}$ to $\mathcal{S}$ **otherwise** append $\{\mathcal{T}_0, \mathcal{T}_1\}$ to $\mathcal{S}$;
3. **if** $V_0 = +\infty$ **then** (5.10) is infeasible, **otherwise** $\mathcal{V}^* \leftarrow V_0$;
4. **end**.

Output: Solution ζ^* of the BQP problem (5.10), optimal cost $\mathcal{V}^*$, or infeasibility status.

by Algorithm 12. The quantity V_0 is the best known integer-feasible cost, which acts as an upper bound of the optimal cost of the BQP problem (5.10). The algorithm is iterated until the stack $\mathcal{S}$ of pending relaxations is empty.

At Step 2.1, the tuple $\mathcal{T}$ from the top of the stack $\mathcal{S}$ is popped and the corresponding relaxed QP problem (5.19) is solved at Step 2.2, under the additional early stopping criterion (5.17). If a feasible solution if found, then Step 2.3.1 checks whether all integrality constraints are satisfied and accordingly updates the best known integer-feasible cost V_0

and corresponding optimizer ζ^*. Conversely, branching is executed at Steps 2.2.3.1–2.2.3.4. Specifically, the index j is picked from the set of relaxed constraints $\mathcal{J}$ by considering the relaxed variable $\bar{\delta}_j^*$ with largest fractional part. The selected constraint j is moved from the set of relaxed inequality constraints $\mathcal{J}$ to the set of equality constraints $\mathcal{I}_{\bar{\ell}}, \mathcal{I}_{\bar{u}}$, thus creating two new relaxed QP subproblems identified by $\mathcal{T}_0$ and $\mathcal{T}_1$. These two new problems represented by $\mathcal{T}_0$ and $\mathcal{T}_1$ are pushed onto the stack $\mathcal{S}$, with priority given based on the distance between the relaxed solution $\bar{\delta}_j^*$ and the bounds $\bar{\ell}_j$ and $\bar{u}_j$.

Once stack $\mathcal{S}$ becomes empty (namely, no further QP relaxations are remaining to be solved) then, Step 3 checks if V_0 is still $+\infty$, and in this case the BQP problem (5.10) is reported infeasible. Otherwise, the optimizer ζ^* and its corresponding optimal solution $\mathcal{V}^*$ is returned.

Exploiting the fixed structure of dual QP relaxations

As described in Algorithm 14, B&B progresses by picking a binary variables from the set $\mathcal{J}$ of relaxed constraints in (5.18c), and moving them to the sets $\mathcal{I}_{\bar{u}}$ and $\mathcal{I}_{\bar{\ell}}$ of equality constraints as in (5.18a) and (5.18b), respectively. This change from one QP relaxation to another only affects the constraints (5.18a)-(5.18c), which simply map to the following modified constraints in the dual QP problem (5.13):

$$\lambda_{\bar{u},\mathcal{I}_{\bar{u}}} = \bar{w}_{\bar{u},\mathcal{I}_{\bar{u}}} + \bar{s}_{\bar{u},\mathcal{I}_{\bar{u}}}, \lambda_{\bar{\ell},\mathcal{I}_{\bar{u}}} = 0, \tag{5.20a}$$

$$\lambda_{\bar{\ell},\mathcal{I}_{\bar{\ell}}} = \bar{w}_{\bar{\ell},\mathcal{I}_{\bar{\ell}}} + \bar{s}_{\bar{\ell},\mathcal{I}_{\bar{\ell}}}, \quad \lambda_{\bar{u},\mathcal{I}_{\bar{\ell}}} = 0, \tag{5.20b}$$

$$\lambda_{\bar{u},\mathcal{J}} \geq 0, \qquad \lambda_{\bar{\ell},\mathcal{J}} \geq 0, \tag{5.20c}$$

with $\lambda_{\bar{u}}$ and $\lambda_{\bar{\ell}}$ represent the decomposition of the dual vector λ in (5.13c). Similarly, $\bar{w}_{\bar{u}}$ and $\bar{w}_{\bar{\ell}}$ (resp., $\bar{s}_{\bar{u}}$ and $\bar{s}_{\bar{\ell}}$) represent the decomposition of the vector $\bar{w}$ in Algorithm 12, Step 6 (resp., $\bar{s}$ in Algorithm 12, Step 8). This leads to a minor change in Algorithm 12. Specifically, in the update of dual vector λ at Step 9, the components of the vectors $\lambda_{\bar{u}}$ and $\lambda_{\bar{\ell}}$ with indexes in the sets $\mathcal{I}_{\bar{u}}$ and $\mathcal{I}_{\bar{\ell}}$ are updated as in (5.20a) - (5.20b). This is a very attractive feature of the GPAD-based B&B solver bqpGPAD, as the same QP structure (matrices and preconditioning) can be computed just once for the initial relaxation at the root node and maintained unaltered throughout the execution of the B&B Algorithm 14.

Computation complexity

Since the binary state vector $\delta(t)$ belongs to $\{0,1\}^{(s-1)n}$, the complexity of the BQP problem (5.10) (in terms of possible combinations of the

binary optimization variables) is $2^p = 2^{T(s-1)n}$. However, at a given time t, each device can operate in one and only one mode (including the OFF state). This is characterized by (5.4b), which reduces the complexity of the BQP problem to s^{Tn}.

It is worth emphasising that, for a fixed length T of timing window where disaggregation is performed, the matrix Q (and its inverse), the Hessian $\mathcal{A}Q^{-1}\mathcal{A}'$ and the constraint matrices $\mathcal{A}$ and b in problem (5.13) can be computed offline only once, as they do not depend on the current measurement $y(t)$. Thus, preconditioning (5.15) can be performed offline only once, and the remaining online operations in Algorithm 12 require only very basic arithmetic operations. This significantly reduces the online computational burden.

5.5 Experimental case study

The effectiveness of the three energy disaggregation algorithms presented in this chapter is demonstrated against the *AMPds* benchmark dataset (Makonin et al., 2013), which contains energy consumption records for individual appliances available in a household located in Vancouver, Canada. Measurements are taken at one-minute time interval for an entire year (from April 1, 2012 to March 31, 2013).

For this experimental study, we consider the aggregate consumption given by the sum of the power consumption readings of the following devices: clothes dryer (CDE), fridge (FGE), dish washer (DWE), heat pump (HPE). Moreover, in order to test the robustness of the disaggregation approaches, a fictitious white noise error $e(t)$, with zero-mean Gaussian distribution and standard deviation 4 W, is added to the aggregate power $y(t)$.

The S-AR models (5.6) used in Approach **A3** to describe the consumption pattern of each appliance are estimated from training data gathered over 500-min intrusive period. Specifically, measurements at day 19 (April 19, 2012) are used to build models for fridge, dish washer, and heat pump, while the behaviour of the clothes dryer is modelled based on data taken at day 38 (May 8, 2012). Switching-AutoRegressive models (5.6) with $s = 3$ modes and dynamical model order $n_a = 2$ are estimated using the moving-horizon identification algorithm proposed in Chapter 4[5]. Once the S-AR model parameters $\Theta_{i,j}$ are estimated, the

[5] An horizon length $N_p = 5$ is used.

Table 12: Power rating levels used in Approaches **A1** and **A2**.

Device	2-modes	3-modes
clothes dryer	[0 4700] W	[0 260 4700] W
fridge	[0 128] W	[0 128 200] W
dish washer	[0 800] W	[0 120 800] W
heat pump	[0 1900] W	[0 39 1900] W

disaggregation problem (5.7) is formulated.

As far as Approach **A1** and **A2** is concerned, the power rating levels $P_i^{(j)}$ (eq. (5.1a)) are chosen via simple visual inspection, based on the same dataset used to estimate the S-AR models in **A3**. 2-mode and 3-mode static models are considered. The corresponding power ratings are reported in Table 12. The $\gamma_i^{(j)}$ hyper-parameters employed in the regularized Approach **A2** (see problem (5.5)) are chosen through cross validation, based on the same disaggregated data gathered during the intrusive period.

5.5.1 Performance Evaluation measures

Disaggregation results are validated based on data measured from June 1 to June 30. The available individual power consumption readings are utilized only as ground-truth data to assess the quality of the results, which is quantified in terms of the following metrics (Piga et al., 2016).

In order to measure the quantity of energy consumed by the i-th appliance as a fraction of the total energy consumption, we define the *Energy Fraction Indices* for both the estimated energy; and true or actual energy. For the estimated energy, the *Estimated Energy Fraction Index* (EEFI) is defined as,

$$\hat{h}_i = \frac{\sum_{t=1}^{T} \hat{y}_i(t)}{\sum_{i=1}^{N}\sum_{t=1}^{T} \hat{y}_i(t)}.$$

An equivalent index for the true or actual energy consumed is,

$$h_i = \frac{\sum_{t=1}^{T} y_i(t)}{\sum_{i=1}^{N}\sum_{t=1}^{T} y_i(t)},$$

which is denoted by the *Actual Energy Fraction Index* (AEFI).

The normalized error between the true and the estimated power consumption of the i-th appliance is given by the *Relative Square Error* (RSE),

$$\mathrm{RSE}_i = \frac{\sum_{t=1}^{T} \left(y_i(t) - \hat{y}_i(t)\right)^2}{\sum_{t=1}^{T} y_i^2(t)}.$$

The R^2 coefficient, defined for the i-th appliance as,

$$R_i^2 = 1 - \frac{\sum_{t=1}^{T} \left(y_i(t) - \hat{y}_i(t)\right)^2}{\sum_{t=1}^{T} \left(y_i(t) - \bar{y}_i\right)^2},$$

where $\bar{y}_i = \frac{1}{T}\sum_{t=1}^{T} y_i(t)$. The RSE and R^2 coefficient gives a measure of the accuracy of the estimated power trends with reference to the actual power trends over specified time interval. Low value of RSE or high value of R^2 coefficient is desired for any disaggregation technique, which will provide better quality estimates of the device specific trends or the EEFI $\hat{h}_i$ to the consumer. This in turn provides vital information for potential reduction in usage of some appliances during the off-peak hours.

5.5.2 Numerical Results

All the numerical experiments are carried out on a desktop computer with Intel Core i7-4700MQ CPU with 2.40 GHz and 8 GB of RAM, using MATLAB R2015a. Algorithm 14 is implemented in *interpreted MATLAB* code, while Algorithms 12 and 13 are implemented in *Embedded MATLAB* code and compiled.

For all approaches **A1**, **A2**, and **A3**, the tolerance values used in the termination criteria (5.14) for Algorithm 12 are set to $\epsilon_G = 10^{-6}$ and $\epsilon_V = 10^{-6}$, while the tolerance value ϵ_I used in Algorithm 13 for infeasibility detection is 10^{-2}. Furthermore, a regularization term $10^{-3}I$ is added to the Hessian of the formulated BQP problems in order to guarantee its strictly positive definiteness.

In order to reduce the computational complexity in solving the regularized BQP (5.5), the available dataset is split into disjoint sets of length $T = 2$ samples. Problem BQP (5.5) is then solved for each subset, by setting $\delta_i^{(j)}(0)$ equal to the estimate $\hat{\delta}_i^{(j)}(T)$ achieved by processing the previous subset. Note that such a splitting introduces suboptimality with respect to using the entire dataset. Similarly, since the estimated regressor $\hat{x}_i(t)$ should be constructed iteratively at every sampling time t, problem (5.7) is solved at each time t for a timing-window of length $T = 1$.

Table 13: Actual Energy Fraction Indices h_i and Estimated Energy Fraction Indices $\hat{h}_i$ obtained by Approach **A1** and **A2** using 2-mode models.

Device	Approach **A1**	Approach **A2**	True values
	$\hat{h}_i$	$\hat{h}_i$	$\hat{h}_i$
CDE	35.4	35.6	31.3
FGE	23.0	23.3	21.3
DWE	7.7	6.6	5.1
HPE	33.9	34.5	42.3

Table 14: Actual Energy Fraction Indices h_i and Estimated Energy Fraction Indices $\hat{h}_i$ obtained by Approach **A1**, **A2**, and **A3** using 3-mode models, LASSO approach is from Piga et al. (2016).

Device	Approach **A1**	Approach **A2**	Approach **A3**	LASSO	True values
	$\hat{h}_i$	$\hat{h}_i$	$\hat{h}_i$	$\hat{h}_i$	h_i
CDE	31.7	31.9	31.9	30.7	31.3
FGE	14.8	19.3	19.7	22.0	21.3
DWE	11.5	6.3	6.9	4.0	5.1
HPE	42.0	42.5	41.5	43.3	42.3

The EEFIs $\hat{h}_i$ obtained by the proposed disaggregation algorithms are reported in Table 13 and 14. More specifically, Table 13 provides the EEFIs $\hat{h}_i$ obtained by Approach **A1** and **A2**, when the appliances are described using 2-mode static models (namely, only ON/OFF modes). Table 14 gives the EEFIs obtained using 3-mode models, both static (in Approach **A1** and **A2**) and dynamic (in Approach **A3**). The RSE and the R^2 coefficients are provided in Table 15 and 16.The results obtained by the LASSO-based algorithm (Piga et al., 2016) are also reported in the same tables for comparison. For the sake of visual representation of the experimental results, Figure 18, 20, 19, and 21 show a portion of the disaggregated power consumption profiles obtained using Approach **A2** and **A3**, respectively. The true disaggregated profiles are also plotted to

assess the quality of the obtained results.

A comparison between the results obtained with 2-mode and 3-mode models (namely, Table 13 *vs* 14, and similarly Table 15 *vs* 16) shows that increasing the complexity of the models of the single appliances provides a better estimate of the disaggregated consumptions, at the price of increasing the computational load. However, it is interesting to note that when the BQP formulation is employed (i.e., Approach **A1**), models with 2 modes leads to more accurate results (lower RSE and higher R^2 coefficients) than models with 3 modes. This can be explained since, when 3-mode models are used, the power consumption levels for fridge and dishwasher are $[0 \ \ 128 \ \ 200]$ W and $[0 \ \ 120 \ \ 800]$ W, respectively (see Table 12). Thus, these two appliances behave similarly when they operate in the second mode, making them difficult to be distinguished. This is reflected in an underestimate of the energy fraction index $\hat{h}_i$ for the fridge and in an overestimate of the energy fraction index for the dishwasher (see Table 14). It is to be noted that the problem formulation from 2-modes towards 3-modes increases the number of binary variables in the BQP problem, resulting in increased computation complexity and computation time. Hence, the number of modes s acts as knob for a trade-off between modelling accuracy and computation load.

It can be also observed that, as expected, adding regularization penalizing the mode transition w.r.t. time (i.e., Approach **A2**) or exploiting transient information on appliances' consumption profiles (*i.e.*, Approach **A3**) outperforms the Binary Quadratic Programming formulation (*i.e.*, Approach **A1**). Furthermore, more accurate results than the LASSO-based approach Piga et al. (2016) are achieved. We highlight that static model Approach **A2** with 3-modes gives comparable results as **A3** with dynamic models as observed from Table 14 and 16, as well as from Figure 18, 20, 19, and 21.

Moreover, we remark that for Approach **A2**, increasing the length T of the timing windows where disaggregation is performed also increases the accuracy of the disaggregated results, at the cost of larger computation load. In contrast, as Approach **A1** does not consider any regularization term, the length T of the timing window has no effect on the accuracy of results (i.e., independent disaggregation problems can be solved at each time t setting $T = 1$).

The average and maximum CPU times required to solve the formulated BQP problems (5.4), (5.5) and (5.7) for a fixed window size T are reported in Table 17, where it can be observed that, in the worst case,

Table 15: RSE_i and R_i^2 coefficients obtained by Approach **A1** and **A2** using 2-mode models.

Device	Approach **A1**		Approach **A2**	
	RSE_i	R_i^2	RSE_i	R_i^2
CDE	0.3	99.7	0.3	99.7
FGE	19.2	71.0	16.4	75.2
DWE	33.3	62.2	11.6	88.2
HPE	2.8	97.0	2.4	97.3

Table 16: RSE_i and R_i^2 coefficients obtained by Approach **A1**, **A2** and **A3** using 3-mode models, LASSO approach is from Piga et al. (2016).

Device	Approach **A1**		Approach **A2**		Approach **A3**		LASSO	
	RSE_i	R_i^2	RSE_i	R_i^2	RSE_i	R_i^2	RSE_i	R_i^2
CDE	0.3	99.7	0.2	99.8	0.1	99.9	0.8	99.2
FGE	36.6	44.7	14.7	77.7	15.2	76.5	24.2	63.3
DWE	38.0	61.1	19.2	80.4	12.4	87.0	28.2	71.4
HPE	4.0	95.6	3.5	96.2	0.5	99.4	2.7	97.1

Table 17: Average and maximum CPU time (in milliseconds) taken to solve the disaggregation problem for a given window size T.

Approach	T	p	bqpGPAD		GUROBI		CPLEX	
			avg	max	avg	max	avg	max
A1 (2-modes)	2	8	11.0	47.0	2.5	31.0	4.2	47.0
A1 (3-modes)	2	16	31.4	203.0	4.1	31.0	7.4	172.0
A2 (2-modes)	2	8	6.0	46.0	2.8	32.0	3.9	47.0
A2 (3-modes)	2	16	16.0	375.0	3.3	32.0	6.7	141.0
A3 (3-modes)	1	12	41.4	188.0	2.9	32.0	13.2	110.0

the disaggregation problem is solved in an order of magnitude of hundreds of milliseconds. For the sake of comparison, the CPU time required by commercial software packages GUROBI (Gurobi Optimization, Inc., 2014) and CPLEX (IBM, Inc., 2014) is also reported. Although GUROBI and CPLEX provide better performance in terms of average and maximum CPU time, we remark that the developed bqpGPAD solver, specifically tailored for the formulated BQP energy disaggregation problems, is library-free and only very basic arithmetic operations should be performed online. These features make it suitable for real-time embedded implementation on smart meters.

5.6 Conclusion

Three algorithms for energy disaggregation based on *Binary Quadratic Programming* (BQP) have been presented. The first algorithm employs switching static models to describe the typical consumption profiles of individual appliances. However, it might not properly handle situations in which appliances have similar power rating levels. This limitation is overcome by the second and the third approach. Specifically, the second disaggregation algorithm exploits the additional information that each electric appliance rarely changes its operating mode over time, and translates this information into the BQP formulation as a penalty in the mode transitions. Instead, the third disaggregation algorithm exploits transient information on the appliances' consumption profiles by using switching dynamical models.

The formulated BQP problems are solved through a customized solver that uses branch and bound coupled with *Accelerated Dual Gradient Projection* (GPAD). The proposed BQP formulations enable to perform computationally intensive operations (preconditioning and matrix factorization) offline, while only very basic arithmetic operations are performed online. This makes the proposed disaggregation approaches appealing for embedded implementation, characterized by high-frequency (> 1 Hz) data acquisition in real-time, and by limited memory availability, directly on smart meters. Only compressed information on end-use consumptions (such as alerts for possible faults, statistics on daily consumption) can be thus transmitted at a lower frequency (*e.g*, daily), thus avoiding network constraints.

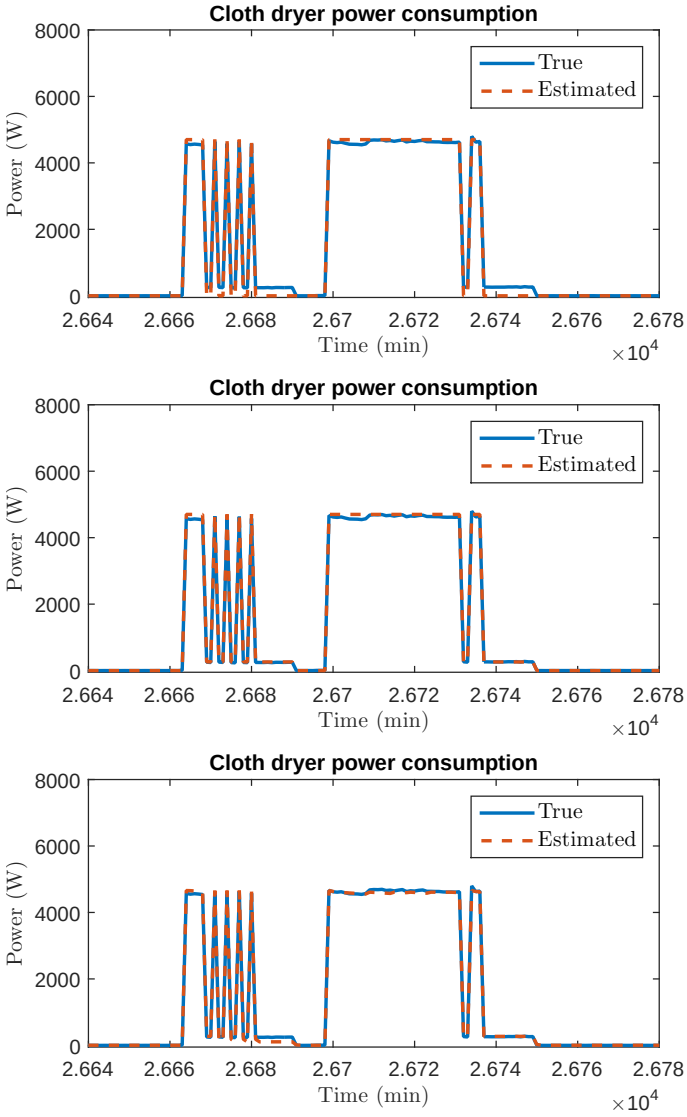


Figure 18: Trends of disaggregated cloth dryer power with Approach A2 for 2-modes (top panel), 3-modes (middle panel), and A3 (bottom panel), where estimated values are calculated using bqpGPAD.

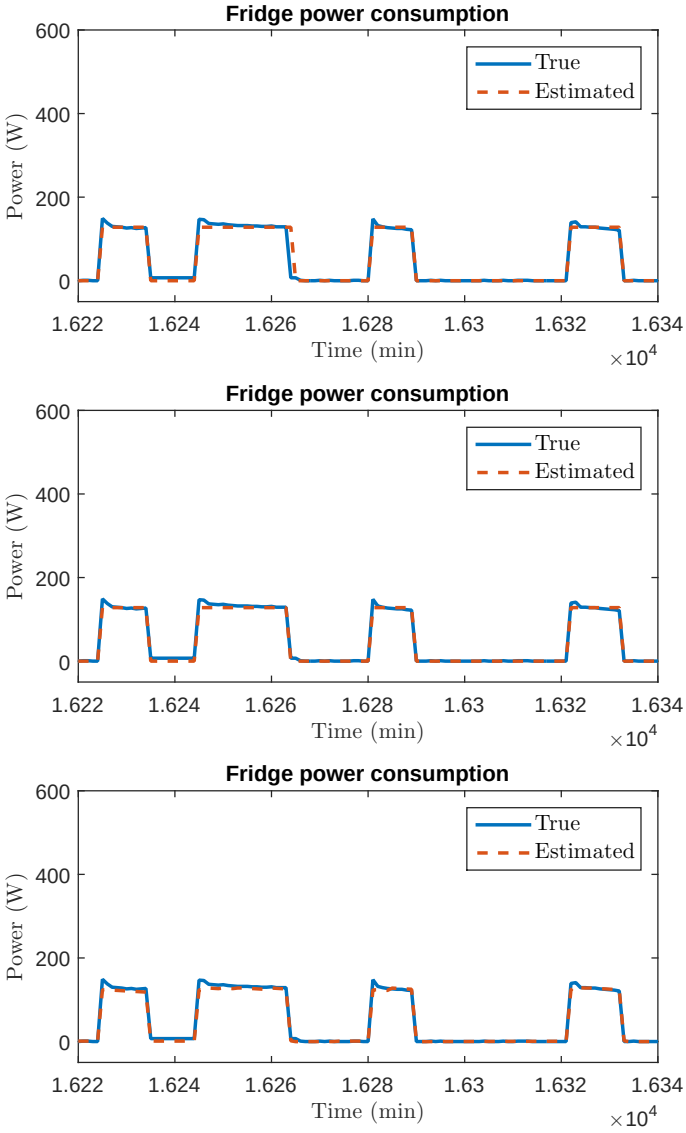


Figure 19: Trends of disaggregated fridge power with Approach A2 for 2-modes (top panel)), 3-modes (middle panel), and A3 (bottom panel), where estimated values are calculated using bqpGPAD.

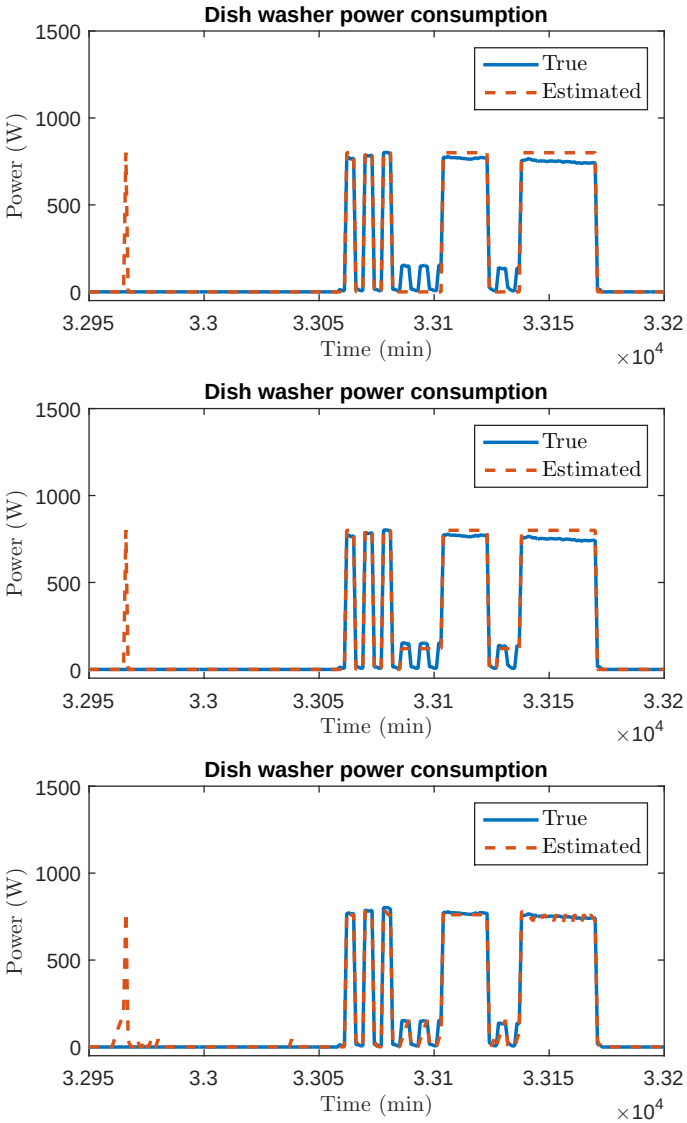


Figure 20: Trends of disaggregated dish washer power with Approach A2 for 2-modes (top panel)), 3-modes (middle panel), and A3 (bottom panel), where estimated values are calculated using bqpGPAD.

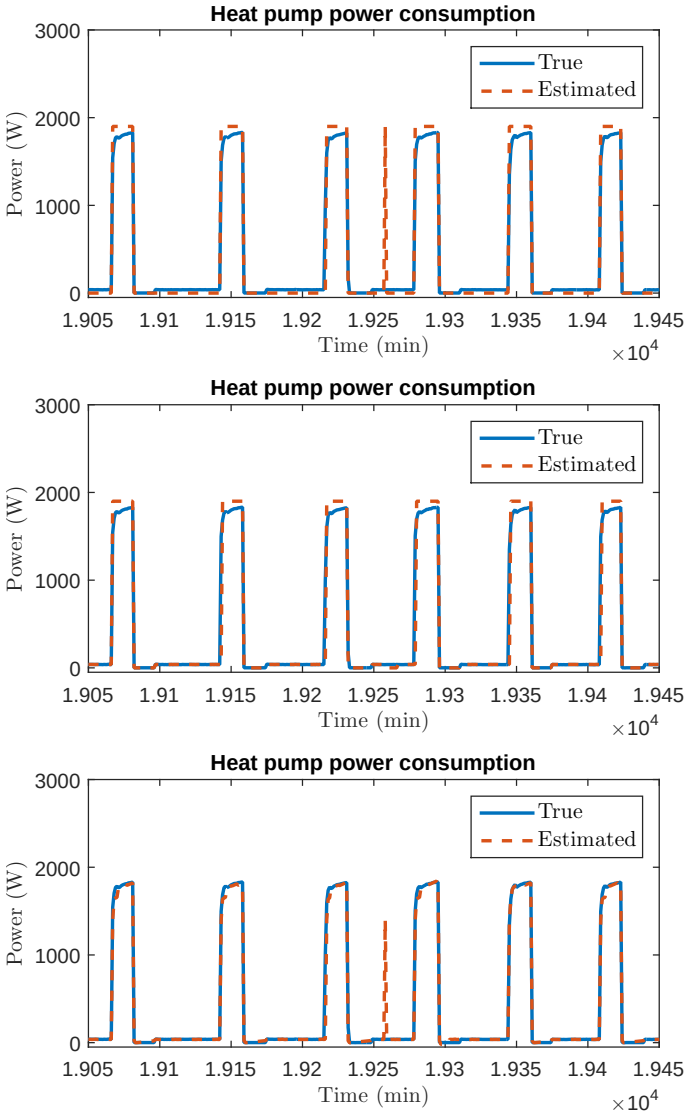


Figure 21: Trends of disaggregated heat pump power with Approach A2 for 2-modes (top panel)), 3-modes (middle panel), and A3 (bottom panel), where estimated values are calculated using bqpGPAD.

Chapter 6

Conclusions and outlook

6.1 Conclusions

In this thesis we introduced new algorithms for Mixed-Integer Quadratic Programming (MIQP) problems tailored to solving small-scale problems such as those that typically arise in embedded applications. The algorithmic simplicity and suitability of embedded implementation is exploited by proposing the *Accelerated Dual Gradient Projection* (GPAD) method combined with Branch and Bound (B&B) in order to solve the MIQP problem to optimality in Chapter 2. Particularly, an existing GPAD algorithm is specialized to complement the requirements of solving QP subproblems arising within the B&B framework. In addition, the GPAD algorithm is used for exploring the possibility to find an integer feasible solution using two different heuristic methods. Further, the idea proposed for using fixed QP matrix structures is employed for the *Alternating Direction Method of Multipliers* (ADMM) algorithm (Boyd et al., 2011) and ADMM based on the *Operator Splitting Quadratic Program* (OSQP) solver (Stellato et al., 2017). Such a fixed structure approach is a very attractive feature utilized in the B&B setup, as the same QP structure including matrices and preconditioning can be computed just once for initial relaxation at the root node and maintained unaltered throughout the B&B algorithm. Hence, the subsequent QP relaxations requires only very basic arithmetic operations to be performed online. An advantage of the MIQP solver based on OSQP solver is that it can handle positive semidefinite Hessian matrix. Whereas, the GPAD based approach has an

advantage of employing dual cost based pruning for premature termination of QP solver.

In spite of their simple coding, the proposed approaches turn out to be quite effective. In particular, the heuristic method can often provide solutions close to optimality with limited computation effort, which can further be used to provide an upper bound on the optimal cost when solving the MIQP to optimality. The performance of the proposed approaches is comparable with state-of-the-art MIQP solvers for small-scale problems, such as those that arise in embedded applications. It is well-known that the performance of first-order methods is highly dependent on the problem conditioning. However, the main motivation to specialize them for solving MIQP problems is to have a simple code requiring only basic arithmetic operations, that makes it suitable for real-time embedded implementation and for ease of compliance with standard production certification tools.

The MIQP solvers based on GPAD and ADMM algorithms presented in Chapter 2 are very simple to code, but they are limited to solving problems with positive definite Hessian matrix. Typically, hybrid Model Predictive Control (MPC) problems based on quadratic costs result in positive semidefinite Hessian matrix. While regularizing the cost function would improve numerical robustness, it would bias the solution away from optimality. In addition to the OSQP based MIQP solver presented in Chapter 2, this limitation is addressed in Chapter 3, where a numerically efficient and robust QP solver based on an active-set method to solve Nonnegative Least Squares (NNLS) combined with proximal-point iterations is proposed, which does not require the Hessian matrix to be positive definite. A generic framework proposed to warm-start binary variables is especially useful in hybrid MPC and moving-horizon estimation, where a good initial guess for the binary variables is available by shifting the optimal solution computed at the previous sample step. Moreover, the "mid-way" heuristic approach proposed in Section 2.4.2 of Chapter 2 is demonstrated as a special case of the warm-start framework proposed for binary variables. The reported numerical results demonstrate that the presented framework for warm-starting binary variables for hybrid MPC problems using the same QP solution algorithm provides better performance in terms of both average and maximum computation time. The same framework was utilized subsequently in Chapter 4.

Chapter 4 and Chapter 5 described novel solution algorithms for estimation problems. Specifically, Chapter 4 proposed a new *PieceWise Affine* (PWA) regression algorithm with the following advantages: *(i)* sim-

ultaneous choice of the model parameters and of the optimal sequence of active modes within a relatively short time horizon and *(ii)* computational efficiency and iterative processing of the training samples, making it suitable for recursive (online) application, where a model should be updated when a new data sample becomes available, without the need to store all past data history. The proposed algorithm is also properly adapted for the identification of *Linear Parameter-Varying* (LPV) models. The binary warm-start proposed in Chapter 3 is combined with GPAD-based MIQP solver presented in Chapter 2, and demonstrated to be effective which renders better solver performance for the formulated MIQP problem using moving-horizon PWA regression algorithm.

Chapter 5 proposed novel Binary Quadratic Programming (BQP) based approaches for energy disaggregation tailored for smart energy meters in a typical household with moderate number of electrical appliances. To address the issue of appliances with similar load signatures, we used an additional regularization term which penalizes the change of appliance state over time. Furthermore, we considered a supervised approach which characterizes the consumption profiles of individual appliance by Switched Affine AutoRegressive (S-AR) models identified by using regularized moving-horizon PWA regression algorithm proposed in Chapter 4. The formulated BQP problems are solved using a custom BQP solver based on GPAD, which is a special case of the MIQP solver introduced in Chapter 2. The proposed energy disaggregation algorithms were tested against a benchmark dataset. A detailed comparative study for appliances with minimal or multiple operating modes, its effects on the problem complexity, computation time and accuracy of the results were presented. This analysis provides a useful means to select appropriate method for online disaggregation, e.g., for selecting number of modes per appliance, static/dynamic models. The approach used to solve the formulated combinatorial problems is library-free, very simple to code, which makes it particularly suitable for embedded implementation on smart meters, with comparable results to state-of-the-art commercial solvers.

6.2 Outlook of possible future directions

An outlook of possible future directions from the contributions described throughout this thesis is as follows.

Hardware implementation of MIQP solving approaches presented

in Chapter 2 and Chapter 3 can be done using low-cost embedded platforms or field programmable gate arrays (FPGA) which enables to solve the QP subproblems in parallel. Furthermore, the solvers can be implemented using universal numbers (UNUM) (Gustafson, 2015), in order to potentially reduce memory requirements for a given problem as compared to the IEEE-754 floating point numbers, which can be particularly beneficial for embedded applications.

QP solution algorithms GPAD, ADMM and NNLS are implemented in Embedded MATLAB code and compiled. Moreover, the branch and bound (B&B) algorithm is implemented in interpreted MATLAB code (for GPAD, ADMM and NNLS) and in Python (for OSQP). Implementation of QP solving algorithms and a B&B algorithm in C can further improve the computation time.

The heuristic approaches introduced in Chapter 2 can be analyzed further in order to guarantee convergence. The warm-start strategy for binary variables introduced in Chapter 3 can be combined with techniques such as "outside-first" tree exploration (Bemporad et al., 1999b), special-ordered sets (SOS) (Beale and Tomlin, 1970), for efficient solution of mixed-integer problems encountered in control and estimation of hybrid systems. The performance of the proposed MIQP approaches in Chapter 2 and Chapter 3 can be tested on optimal control problems for hybrid systems, formulated using techniques such as disjunctive programming. Adaptability of other techniques such as branch and bound with cutting planes, or branch and cut for MIQP solvers tailored to embedded platforms can be explored. Another research direction can be to explore combination of machine learning based techniques with branch and bound (B&B) for the solution of MIQP problems, see recent survey paper (Lodi and Zarpellon, 2017).

The PWA regression algorithm presented in Chapter 4 can be extended to solve the MIQP problem as well as to compute the partition of the polyhedral regions online. Moreover, an area that can be explored is, extensions of the proposed method to other PWA structures such as output-error PWA and Box-Jenkins PWA models. The energy disaggregation approaches in Chapter 5 can be extended to incorporate information on user behavior, and on appliance operating cycle to enhance the disaggregation performance, and extensive validation of the approaches on embedded hardware can be carried out.

Appendix A

Proof of Theorem 1

This proof extends the proof reported in (Bemporad, 2018) to the case of bilateral inequality constraints. First, by defining

$$u \triangleq \mathcal{L}z + \mathcal{L}^{-T}(c - \epsilon z_k) \tag{A.1}$$

we complete the squares in (3.1a) by substituting $z = \mathcal{L}^{-1}u - (Q + \epsilon I)^{-1}(c - \epsilon z_k)$ and recast (3.1) into the equivalent constrained Least Distance Problem (LDP)

$$\begin{aligned} \min_{u} \quad & \frac{1}{2}\|u\|^2 && \text{(A.2a)} \\ \text{s.t.} \quad & \mathcal{M}u \leq d^k && \text{(A.2b)} \\ & Nu = f^k && \text{(A.2c)} \end{aligned}$$

where $\mathcal{M} \triangleq \left[\begin{smallmatrix} -M \\ M \end{smallmatrix}\right]$, $d^k \triangleq \left[\begin{smallmatrix} d_\ell^k \\ d_u^k \end{smallmatrix}\right]$.

i) Assume the optimal residual $r^* = 0$ in (3.8b). Let $\nu_+^* \triangleq \max\{\nu^*, 0\}$, $\nu_-^* \triangleq \max\{-\nu^*, 0\}$ be the positive and negative parts of ν^*, $\nu^* = \nu_+^* - \nu_-^*$, $\nu_+^*, \nu_-^* \geq 0$. Then, by (3.8b) we get

$$\begin{aligned} \mathcal{M}'y^* + N'\nu_+^* - N'\nu_-^* &= 0 \\ (d^k)'y^* + (f^k)'\nu_+^* - (f^k)'\nu_-^* &= -\gamma \\ y^*, \nu_+^*, \nu_-^* &\geq 0 \end{aligned} \tag{A.3}$$

where $y^* \triangleq \left[\begin{smallmatrix} y^*_\ell \\ y^*_u \end{smallmatrix}\right]$. By Farkas's Lemma (Rockafellar, 1970, p. 201), for any $\gamma > 0$ (A.3) is equivalent to infeasibility of

$$\begin{aligned} \mathcal{M}u &\le d \\ Nu &\le f \\ -Nu &\le -f \end{aligned} \tag{A.4}$$

which is obviously equivalent to (A.2b)–(A.2c). Therefore the LDP problem (A.2) does not admit a solution, and consequently (3.1).

ii) Consider the KKT conditions for problem (3.7)

$$-\left[\begin{smallmatrix} \mathcal{M} & \beta d^k \\ N & \beta f^k \end{smallmatrix}\right] \left[\begin{smallmatrix} -\mathcal{M}'y^* - N'\nu^* \\ -\beta((d^k)'y^* + (f^k)'\nu^* + \gamma) \end{smallmatrix}\right] - \left[\begin{smallmatrix} I \\ 0 \end{smallmatrix}\right] w^* = 0 \tag{A.5a}$$

$$(y^*)'w^* = 0 \tag{A.5b}$$

$$\nu^* \text{ free}, \; w^* \ge 0, y^* \ge 0 \tag{A.5c}$$

where $w^* \triangleq \left[\begin{smallmatrix} w^*_\ell \\ w^*_u \end{smallmatrix}\right]$ is the optimal dual variable for problem (3.7). From (A.5a) we get

$$-\left[\begin{smallmatrix} \mathcal{M} & \beta d^k \end{smallmatrix}\right] r^* - w^* = 0 \tag{A.6a}$$

$$-\left[\begin{smallmatrix} N & \beta f^k \end{smallmatrix}\right] r^* = 0 \tag{A.6b}$$

and hence the condition $r^* \neq 0$, (A.5a)–(A.5b) and (A.6) imply that

$$\begin{aligned} 0 \;&<\; (r^*)'r^* = (r^*)' \left[\begin{smallmatrix} -\mathcal{M}' \\ -\beta(d^k)' \end{smallmatrix}\right] y^* + (r^*)' \left[\begin{smallmatrix} N' \\ \beta(f^k)' \end{smallmatrix}\right] \nu^* - \gamma\beta r^*_{n+1} \\ &=\; (w^*)'y^* - \gamma\beta r^*_{n+1} = -\gamma\beta r^*_{n+1}, \end{aligned}$$

i.e., $r^*_{n+1} = -\beta((d^k)'y^* + (f^k)'\nu^* + \gamma) < 0$. By letting

$$u^* \triangleq -\frac{1}{\beta r^*_{n+1}} r^*_{\{1,\ldots,n\}} = -\frac{\mathcal{M}'y^* + N'\nu^*}{\beta^2(\gamma + (d^k)'y^* + (f^k)'\nu^*)}, \tag{A.7}$$

from (A.5c) and (A.6a) we obtain

$$0 \le w^* = -\left[\begin{smallmatrix} \mathcal{M} & \beta d^k \end{smallmatrix}\right] r^* = -r^*_{n+1} \left[\begin{smallmatrix} \beta\mathcal{M} & \beta d^k \end{smallmatrix}\right] \left[\begin{smallmatrix} \frac{r^*_{\{1,\ldots,n\}}}{\beta r^*_{n+1}} \\ 1 \end{smallmatrix}\right]$$

and hence $-\mathcal{M}u^* + d^k \ge 0$, or equivalently u^* satisfies (A.2b). Moreover,

$$Nu^* - f = -N\frac{\mathcal{M}'y^* + N'\nu^*}{\beta^2(\gamma + (d^k)'y^* + (f^k)'\nu^*)} - f^k = 0$$

iff $0 = N\mathcal{M}'y^* + NN'\nu^* + \beta^2(\gamma f^k + f^k(d^k)'y^* + f^k(f^k)'\nu^*) = (NN' + \beta^2 f^k(f^k)')\nu^* + (N\mathcal{M}' + \beta^2 f^k(d^k)')y^* + \beta^2\gamma f$, or equivalently iff

$$\nu^* = -\begin{bmatrix} N' \\ \beta(f^k)' \end{bmatrix}^{\#} \left(\begin{bmatrix} \mathcal{M}' \\ \beta(d^k)' \end{bmatrix} y^* + \begin{bmatrix} 0 \\ \beta\gamma \end{bmatrix} \right). \tag{A.8}$$

Since by (Bemporad, 2015a, Lemma 1) condition (A.8) is always satisfied at optimality of (3.7), we have proved that u^* also satisfies (A.2c) and therefore is a feasible candidate to solve (A.2). It remains to prove that u^* is also optimal for (A.2). To this end, consider the remaining KKT conditions of optimality for problem (A.2)

$$\begin{aligned} u^* + \mathcal{M}'\lambda^* + N'\mu^* &= 0 && \text{(A.9a)} \\ (\lambda^*)'(\mathcal{M}u^* - d^k) &= 0 && \text{(A.9b)} \\ \lambda^* \geq 0,\ \nu^* &\text{ free.} && \text{(A.9c)} \end{aligned}$$

Let

$$\lambda^* \triangleq \begin{bmatrix} \lambda_\ell^* \\ \lambda_u^* \end{bmatrix} = -\frac{1}{\beta r_{n+1}^*} y^*,\ \mu^* \triangleq -\frac{1}{\beta r_{n+1}^*}\nu^*. \tag{A.10}$$

By negativity of r_{n+1}^* and nonnegativity of y^* we get $\lambda^* \geq 0$. Moreover, by recalling (A.7), we get

$$u^* = \frac{1}{\beta r_{n+1}^*}(\mathcal{M}'y^* + N'\nu^*) = -\mathcal{M}'\lambda^* - N'\nu^*$$

so that also (A.9a) is satisfied. To prove (A.9b) we observe that $(\lambda^*)'(\mathcal{M}u^* - d^k) = -\frac{1}{\beta r_{n+1}^*}(\lambda^*)'(\mathcal{M}r_{\{1,\dots,n\}}^* + \beta d^k r_{n+1}^*) = \frac{1}{\beta^2(r_{n+1}^*)^2}(y^*)'\begin{bmatrix} \mathcal{M} & \beta d^k \end{bmatrix} r^* = -\frac{1}{\beta^2(r_{n+1}^*)^2}(y^*)'w^* = 0$ because of (A.6a) and (A.5b). In conclusion, u^* is the optimal solution of problem (A.2), and hence the vector z^* defined in (3.9) solves (3.1a)–(3.1c). As problems (A.2) and (3.5) only differ for the transformation (A.1) of primal variables, by (A.9) (λ^*, μ^*) is also the optimal dual pair for (3.5). This can be also directly inspected by substituting $u^* = \mathcal{L}z^* + \mathcal{L}^{-T}(c - \epsilon z_k)$ in (A.9a) and left-multiplying by $\mathcal{L}'$, and in (A.9b). ■

References

T. Achterberg and T. Berthold. Improving the feasibility pump. *Discrete Optimization*, 4(1):77–86, 2007. 12

A. Alessio and A. Bemporad. A survey on explicit model predictive control. In D. R. L. Magni, F. Allgower, editor, *Nonlinear Model Predictive Control: Towards New Challenging Applications*, volume 384 of *Lecture Notes in Control and Information Sciences*, pages 345–369, Berlin Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-01093-4. 6

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, July 2000. 4

K. C. Armel, A. Gupta, G. Shrimali, and A. Albert. Is disaggregation the holy grail of energy efficiency? The case of electricity. *Energy Policy*, 52:213–234, 2013. 102

D. Axehill and A. Hansson. A mixed integer dual quadratic programming algorithm tailored for MPC. In *Proc. 45th IEEE Conference on Decision and Control*, pages 5693–5698, San Diego, CA, USA, 2006. 2, 9, 11

D. Axehill and A. Hansson. A dual gradient projection quadratic programming algorithm tailored for mixed integer predictive control. Technical Report LiTH-ISY-R-2833, Department of Electrical Engineering, Linköping University, Sweden, 2008. 11

L. Bako. Identification of switched linear systems via sparse optimization. *Automatica*, 47(4):668–677, 2011. 80

L. Bako, K. Boukharouba, E. Duviella, and S. Lecoeuche. A recursive identification algorithm for switched linear/affine models. *Nonlinear Analysis: Hybrid Systems*, 5(2):242–253, 2011. 81

E. Balas. Duality in discrete programming: II. The quadratic case. *Management Science*, 16(1):14–32, 1969. 2

G. Banjac, P. Goulart, B. Stellato, and S. Boyd. Infeasibility detection in the alternating direction method of multipliers for convex optimization. *optimization-online.org*, 2017. URL http://www.optimization-online.org/DB_HTML/2017/06/6058.html. 41

M. Baranski and J. Voss. Genetic algorithm for pattern detection in nialm systems. In *Proc. IEEE International Conference on Systems, Man and Cybernetics*, volume 4, pages 3462–3468, The Hague, Netherlands, Oct 2004. 103

E. Beale and R. Small. Mixed integer programming by a branch and bound technique. In *Proceedings of the IFIP Congress*, volume 2, pages 450–451, 1965. 10

E. M. L. Beale and J. A. Tomlin. Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. In *Proc.Fifth International Conference on Operational Research*, pages 447–454, 1970. 132

A. Bemporad. *Hybrid Toolbox – User's Guide*. Dec. 2003. http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox. 6, 7, 77

A. Bemporad. Efficient conversion of mixed logical dynamical systems into an equivalent piecewise affine form. *IEEE Transactions on Automatic Control*, 49(5): 832–838, May 2004. 8

A. Bemporad. A multiparametric quadratic programming algorithm with polyhedral computations based on nonnegative least squares. *IEEE Trans. Automatic Control*, 60(11):2892–2903, 2015a. 6, 135

A. Bemporad. Solving mixed-integer quadratic programs via nonnegative least squares. In *Proc. 5th IFAC Conference on Nonlinear Model Predictive Control*, pages 73–79, Seville, Spain, 2015b. 11, 12, 60, 63, 104

A. Bemporad. A quadratic programming algorithm based on nonnegative least squares with applications to embedded model predictive control. *IEEE Trans. Automatic Control*, 61(4):1111–1116, April 2016. 10, 11, 60, 63

A. Bemporad. A numerically stable solver for positive semidefinite quadratic programs based on nonnegative least squares. *IEEE Trans. Automatic Control*, 63(2):525–531, Feb 2018. 10, 60, 62, 63, 64, 67, 133

A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999a. 3, 4, 5, 6, 8, 9, 59, 60, 61, 74, 77, 80

A. Bemporad and M. Morari. Robust model predictive control: A survey. In A. Garulli and A. Tesi, editors, *Robustness in identification and control*, pages 207–226, London, 1999b. Springer London. 6

A. Bemporad and V. V. Naik. A numerically robust mixed-integer quadratic programming solver for embedded hybrid model predictive control. In *Proc. 6th IFAC Conference on Nonlinear Model Predictive Control*, pages 412–417, Madison, Wisconsin, USA, 2018.

A. Bemporad, D. Mignone, and M. Morari. Moving horizon estimation for hybrid systems and fault detection. In *Proc. American Control Conference*, pages 2471–2475, Chicago, IL, June 1999a. 3, 60

A. Bemporad, D. Mignone, and M. Morari. An efficient branch and bound algorithm for state estimation and control of hybrid systems. In *Proc. European Control Conference (ECC)*, pages 557–562, Aug 1999b. 132

A. Bemporad, G. Ferrari-Trecate, and M. Morari. Observability and controllability of piecewise affine and hybrid systems. *IEEE transactions on automatic control*, 45(10):1864–1876, 2000. 5, 8, 80

A. Bemporad, J. Roll, and L. Ljung. Identification of hybrid systems via mixed-integer programming. In *Proc. 40th IEEE Conference on Decision and Control*, volume 1, pages 786–792, Dec 2001. 3, 8

A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002. 6

A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino. A bounded-error approach to piecewise affine system identification. *IEEE Transactions on Automatic Control*, 50(10):1567–1580, 2005. 80, 81

A. Bemporad, D. Bernardini, and P. Patrinos. A convex feasibility approach to anytime model predictive control. Technical report, IMT Institute for Advanced Studies, Lucca, Feb. 2015. http://arxiv.org/abs/1502.07974. 92

K. Bennett and O. Mangasarian. Multicategory discrimination via linear programming. *Optimization Methods and Software*, 3:27–39, 1994. 91, 92

L. Bertacco, M. Fischetti, and A. Lodi. A feasibility pump heuristic for general mixed-integer problems. *Discrete Optimization*, 4(1):63–76, 2007. 12

D. Bertsekas. *Convex Optimization Theory*. Athena Scientific, 2009. 28, 113

M. Z. A. Bhotto, S. Makonin, and I. V. Baji. Load disaggregation based on aided linear integer programming. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 64(7):792–796, July 2017. ISSN 1549-7747. https://doi.org/10.1109/TCSII.2016.2603479. 9, 103

D. Bienstock. Computational study of a family of mixed-integer quadratic programming problems. *Mathematical Programming*, 74(2):121–140, Aug 1996. 9

M. Bierlaire, P. Toint, and D. Tuyttens. On iterative algorithms for linear ls problems with bound constraints. *Linear Algebra and Its Applications*, 143:111–143, 1991. 44

R. E. Bixby. A brief history of linear and mixed-integer programming computation. *Documenta Mathematica*, pages 107–121, 2010. 10

F. Borrelli, M. Baotić, A. Bemporad, and M. Morari. Dynamic programming for constrained optimal control of discrete-time linear hybrid systems. *Automatica*, 41(10):1709–1721, Oct. 2005. 4, 6

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011. 25, 38, 40, 41, 129

L. Breiman. Hinging hyperplanes for regression, classification, and function approximation. *IEEE Transactions on Information Theory*, 39(3):999–1013, 1993. 8, 80

V. Breschi, A. Bemporad, and D. Piga. Identification of hybrid and linear parameter varying models via recursive piecewise affine regression and discrimination. In *Proc. European Control Conference*, pages 2632–2637, 2016a. 81, 96

V. Breschi, D. Piga, and A. Bemporad. Piecewise affine regression via recursive multiple least squares and multicategory discrimination. *Automatica*, 73:155–162, 2016b. 81, 82, 91, 92, 93, 104

C. G. Cassandras, D. L. Pepyne, and Y. Wardi. Optimal control of a class of hybrid systems. *IEEE Trans. Automatic Control*, 46(3):398–415, March 2001. 4

J. P. S. Catalão, H. M. I. Pousinho, and V. M. F. Mendes. Scheduling of head-dependent cascaded hydro systems: Mixed-integer quadratic programming approach. *Energy Conversion and Management*, 51(3):524–530, 2010. 4

G. Cimini, A. Bemporad, and D. Bernardini. ODYS QP Solver. ODYS S.r.l. (`https://odys.it/qp`), Sept 2017. 11

A. Cominola, M. Giuliani, D. Piga, A. Castelletti, and A. E. Rizzoli. A hybrid signature-based iterative disaggregation algorithm for non-intrusive load monitoring. *Applied energy*, 185:331–344, 2017. 103

R. J. Dakin. A tree-search algorithm for mixed integer programming problems. *The Computer Journal*, 8(3):250–255, 1965. 10

T. Davis. *Direct Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2006. https://doi.org/10.1137/1.9780898718881. 41

A. Domahidi and J. Jerez. FORCES Professional. embotech GmbH (http://embotech.com/FORCES-Pro), July 2014. 11

A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones. Efficient interior point methods for multistage problems arising in receding horizon control. In *Proc. 51st IEEE Conference on Decision and Control*, pages 668–674, Dec 2012. 11

P. Ducange, F. Marcelloni, and M. Antonelli. A novel approach based on finite-state machines with fuzzy transitions for nonintrusive home appliance monitoring. *IEEE Transactions on Industrial Informatics*, 10(2):1185–1197, 2014. 103

M. A. Duran and I. E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36(3):307–339, Oct 1986. 9

Fair Isaac Corporation. *FICO Xpress Optimization Suite*, 2015. http://www.fico.com/. 4, 7

A. Faustine, N. H. Mvungi, S. Kaijage, and K. Michael. A survey on non-intrusive load monitoring methodies and techniques for energy disaggregation problem. *arXiv preprint arXiv:1703.00785*, 2017. 103

G. Ferrari-Trecate, D. Mignone, and M. Morari. Moving horizon estimation for hybrid systems. *IEEE Trans. Automatic Control*, 47(10):1663–1676, 2002. 3, 60, 104

G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari. A clustering technique for the identification of piecewise affine systems. *Automatica*, 39(2):205–217, 2003. 81, 104

H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014. 10

H. Ferreau, S. Almér, R. Verschueren, M. Diehl, D. Frick, A. Domahidi, J. Jerez, G. Stathopoulos, and C. Jones. Embedded optimization methods for industrial automatic control. In *Proc. 20th IFAC World Congress*, pages 13194–13209, Toulouse, France, 2017. 11

M. Figueiredo, B. Ribeiro, and A. de Almeida. On the regularization parameter selection for sparse code learning in electrical source separation. In *Adaptive and Natural Computing Algorithms*, pages 277–286. Springer, 2013. 103

M. Fischetti, F. Glover, and A. Lodi. The feasibility pump. *Mathematical Programming*, 104(1):91–104, 2005. 12

R. Fletcher and S. Leyffer. Solving mixed integer nonlinear programs by outer approximation. *Mathematical Programming*, 66(1):327–349, Aug 1994. 9

R. Fletcher and S. Leyffer. Numerical experience with lower bounds for MIQP branch-and-bound. *SIAM J. Optim.*, 8(2):604–616, May 1998. 2, 9, 11

C. A. Floudas. *Nonlinear and Mixed-Integer Optimization*. Oxford University Press, 1995. 11, 103

D. Frick, A. Domahidi, and M. Morari. Embedded optimization for mixed logical dynamical systems. *Computers & Chemical Engineering*, 72:21–33, 2015. 11, 103

D. Frick, J. L. Jerez, A. Domahidi, A. Georghiou, and M. Morari. Low-complexity iterative method for hybrid MPC. *ArXiv e-prints*, 2016. 12

A. Garulli, S. Paoletti, and A. Vicino. A survey on switched and piecewise affine system identification. In *Proc. 16th IFAC Symposium on System Identification*, pages 344–355, Brussels, Belgium, 2012. 8, 80

A. M. Geoffrion. Generalized benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260, Oct 1972. 9

J. M. Gillis and W. G. Morsi. Non-intrusive load monitoring using semi-supervised machine learning and wavelet design. *IEEE Transactions on Smart Grid*, 8(6):2648–2655, 2017. 103

P. Giselsson. Improved fast dual gradient methods for embedded model predictive control. In *Proc. 19th IFAC World Congress*, pages 2303–2309, 2014. 24

P. Giselsson and S. Boyd. Monotonicity and restart in fast gradient methods. In *Proc. 53rd IEEE Conference on Decision and Control*, pages 5058–5063, Dec 2014. 29, 114

R. Goebel, R. G. Sanfelice, and A. R. Teel. Hybrid dynamical systems. *IEEE Control Systems Magazine*, 29(2):28–93, April 2009. 4

R. E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bull. Amer. Math. Soc.*, 64(5):275–278, 09 1958. 9

R. E. Gomory. An algorithm for integer solutions to linear programs. *Recent advances in mathematical programming*, 64:269–302, 1963. 9

I. E. Grossmann. Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and Engineering*, 3(3):227–252, Sep 2002. 9

O. K. Gupta and A. Ravindran. Branch and bound experiments in convex nonlinear integer programming. *Manage. Sci.*, 31(12):1533–1546, Dec. 1985. 9

Gurobi Optimization, Inc. *Gurobi Optimizer Reference Manual*, 2014. URL http://www.gurobi.com. 4, 7, 43, 77, 103, 105, 124

J. L. Gustafson. *The End of Error: Unum Computing*. CRC Press, 2015. 132

G. W. Hart. Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 80 (12):1870–1891, 1992. 8, 102

W. Heemels, B. De Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091, 2001. 5, 8, 80

M. Herceg, M. Kvasnica, C. N. Jones, and M. Morari. Multi-parametric toolbox 3.0. In *Proc. European Control Conference*, pages 502–510, July 2013. 5, 6, 7

IBM, Inc. *IBM ILOG CPLEX Optimization Studio 12.6 – User Manual*, 2014. 4, 7, 77, 103, 105, 124

D. D. Ingole, D. N. Sonawane, V. V. Naik, D. L. Ginoya, and V. Patki. Implementation of model predictive control for closed loop control of anesthesia. In *Mobile Communication and Power Engineering*, pages 242–248, Berlin, Heidelberg, 2013a. Springer Berlin Heidelberg.

D. D. Ingole, D. N. Sonawane, V. V. Naik, D. L. Ginoya, and V. V. Patki. Linear model predictive controller for closed-loop control of intravenous anesthesia with time delay. *International Journal on Control System and Instrumentation*, 4 (1):8–15, 2013b.

J. L. Jerez, P. J. Goulart, S. Richter, G. A. Constantinides, E. C. Kerrigan, and M. Morari. Embedded predictive control on an FPGA using the fast gradient method. In *Proc. European Control Conference*, pages 3614–3620, 2013. 24

J. L. Jerez, P. J. Goulart, S. Richter, G. A. Constantinides, E. C. Kerrigan, and M. Morari. Embedded online optimization for model predictive control at megahertz rates. *IEEE Transactions on Automatic Control*, 59(12):3238–3251, Dec 2014. 24

A. L. Juloski, S. Weiland, and W. Heemels. A Bayesian approach to identification of hybrid systems. *IEEE Transactions on Automatic Control*, 50(10):1520–1533, 2005. 81

M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, editors. *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. Springer, Heidelberg, 2010. 10

M. Kögel and R. Findeisen. A fast gradient method for embedded linear predictive control. In *Proc. 18th IFAC World Congress*, pages 1362–1367, 2011. 24

M. Kvasnica, P. Grieder, M. Baotić, and M. Morari. Multi-parametric toolbox (MPT). In *Proc. International Workshop on Hybrid Systems: Computation and Control*, pages 448–462, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. 6, 7

A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960. 9

F. Lauer. On the complexity of piecewise affine system identification. *Automatica*, 62:148–153, 2015. 80

E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719, 1966. 10

R. Lazimy. Improved algorithm for mixed-integer quadratic programs and a computational study. *Mathematical Programming*, 32(1):100–113, May 1985. 2, 9

A. Likas, N. Vlassis, and J. Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003. 108

B. Lincoln and A. Rantzer. Optimizing linear system switching. In *Proc. 40th IEEE Conference on Decision and Control*, volume 3, pages 2063–2068, Dec 2001. 4

J. T. Linderoth and M. W. P. Savelsbergh. A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing*, 11(2):173–187, Feb. 1999. 9, 10

J. D. Little, K. G. Murty, D. W. Sweeney, and C. Karel. An algorithm for the traveling salesman problem. *Operations research*, 11(6):972–989, 1963. 10

L. Ljung. System identification toolbox. *The Matlab users guide*, 1988. 52

A. Lodi and G. Zarpellon. On learning and branching: a survey. *TOP*, 25(2): 207–236, Jul 2017. 132

J. Löfberg. YALMIP : a toolbox for modeling and optimization in MATLAB. In *Proc. IEEE International Conference on Robotics and Automation*, pages 284–289, Sept 2004. 6, 7

S. Makonin, F. Popowich, L. Bartram, B. Gill, and I. Bajic. AMPds: a public dataset for load disaggregation and eco-feedback research. In *Proc. Electrical Power and Energy Conference*, pages 1–6, 2013. 105, 118

J. Mattingley and S. Boyd. Cvxgen: a code generator for embedded convex optimization. *Optimization and Engineering*, 13(1):1–27, Mar 2012. 10

D. Mayne, J. Rawlings, C. Rao, and P. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000. 6

M. Mejari, V. V. Naik, D. Piga, and A. Bemporad. Energy disaggregation using piecewise affine regression and binary quadratic programming. In *Proc. 57th IEEE Conference on Decision and Control*, Miami Beach, FL, USA, 2018a.

M. Mejari, V. V. Naik, D. Piga, and A. Bemporad. Regularized moving-horizon PWA regression for LPV system identification. In *Proc. 18th IFAC Symposium on System Identification*, pages 1092–1097, Stockholm, Sweden, 2018b. 47, 93

D. Mellinger, A. Kushleyev, and V. Kumar. Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In *Proc. IEEE International Conference on Robotics and Automation*, pages 477–483, Saint Paul, MN, USA, 2012. 3

D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79–102, 2016. 9, 10

MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 7.1 (Revision 28)*, 2015. `http://docs.mosek.com/7.1/toolbox/index.html`. 4, 7, 103

V. V. Naik and A. Bemporad. Embedded mixed-integer quadratic optimization using first-order methods. In *Proc. 4th European Conference on Computational Optimization*, Leuven, Belgium, 2016.

V. V. Naik and A. Bemporad. Embedded mixed-integer quadratic optimization using accelerated dual gradient projection. In *Proc. 20th IFAC World Congress*, pages 10723–10728, Toulouse, France, 2017.

V. V. Naik and A. Bemporad. Mixed-integer quadratic optimization based on accelerated dual gradient projection for embedded applications. Technical report, 2018.

V. V. Naik, D. Sonawane, D. D. Ingole, D. L. Ginoya, and V. V. Patki. Design and implementation of proportional integral observer based linear model predictive controller. *International Journal on Control System and Instrumentation*, 4(1): 23–30, 2013a.

V. V. Naik, D. N. Sonawane, D. D. Ingole, and D. Ginoya. Model predictive control of DC servomotor using active set method. In *Proc. IEEE International Conference on Control Applications*, pages 820–825, Aug 2013b.

V. V. Naik, D. N. Sonawane, D. D. Ingole, D. L. Ginoya, and N. S. Girme. Design and implementation of interior-point method based linear model predictive controller. In *Mobile Communication and Power Engineering*, pages 255–261, Berlin, Heidelberg, 2013c. Springer Berlin Heidelberg.

V. V. Naik, M. Mejari, D. Piga, and A. Bemporad. Regularized moving-horizon piecewise affine regression using mixed-integer quadratic programming. In *Proc. 25th Mediterranean Conference on Control and Automation*, pages 1349–1354, Valletta, Malta, 2017.

V. V. Naik, M. Mejari, D. Piga, and A. Bemporad. Energy disaggregation using embedded binary quadratic programming. Submitted for publication, 2018.

H. Nakada, K. Takaba, and T. Katayama. Identification of piecewise affine systems based on statistical clustering technique. *Automatica*, 41(5):905–913, 2005. 81

Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Sov. Math. Doklady*, 27(2):372–376, 1983. 24, 104, 112

Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*, volume 87. Springer, 2004. 24

B. O' Donoghue and E. Candès. Adaptive restart for accelerated gradient schemes. *Foundations of Computational Mathematics*, 15(3):715–732, 2015. 29, 114

B. O' Donoghue, E. Chu, N. Parikh, and S. Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *J. Optim. Theory Appl.*, 169 (3):1042–1068, June 2016. ISSN 0022-3239. 29

H. Ohlsson and L. Ljung. Identification of switched linear regression models using sum-of-norms regularization. *Automatica*, 49(4):1045–1050, 2013. 52, 80

N. Ozay, C. Lagoa, and M. Sznaier. Set membership identification of switched linear systems with known number of subsystems. *Automatica*, 51:180–191, 2015. 80

S. Paoletti, A. L. Juloski, G. Ferrari-Trecate, and R. Vidal. Identification of hybrid systems a tutorial. *European journal of control*, 13(2):242–260, 2007. 8, 80

L. G. Papageorgiou and E. S. Fraga. A mixed integer quadratic programming formulation for the economic dispatch of generators with prohibited operating zones. *Electric Power Systems Research*, 77(10):1292–1296, 2007. 3

P. Patrinos and A. Bemporad. An accelerated dual gradient-projection algorithm for embedded linear model predictive control. *IEEE Trans. Automatic Control*, 59(1):18–33, 2014. 11, 24, 25, 27, 28, 87, 104, 112, 113

P. Patrinos, A. Guiggiani, and A. Bemporad. Fixed-point dual gradient projection for embedded model predictive control. In *Proc. European Control Conference*, pages 3602–3607, Zurich, Switzerland, 2013. 24

D. Piga and R. Tóth. An SDP approach for ℓ_0-minimization: Application to ARX model segmentation. *Automatica*, 49(12):3646–3653, 2013. 50, 51

D. Piga, A. Cominola, M. Giuliani, A. Castelletti, and A. E. Rizzoli. Sparse optimization for automated energy end use disaggregation. *IEEE Transactions on Control Systems Technology*, 24(3):1044–1051, 2016. xv, 103, 105, 119, 121, 122, 123

M. Propato and J. G. Uber. Booster system design using mixed-integer quadratic programming. *Journal of Water Resources Planning and Management*, 130(4):348–352, 2004. 3

A. Raghunathan and S. Di Cairano. Infeasibility detection in alternating direction method of multipliers for convex quadratic programs. In *Proc. 53rd IEEE Conference on Decision and Control*, pages 5819–5824, Dec 2014. 29, 39

A. Rahimpour, H. Qi, D. Fugate, and T. Kuruganti. Non-intrusive energy disaggregation using non-negative matrix factorization with sum-to-k constraint. *IEEE Transactions on Power Systems*, 32(6):4430–4441, 2017. 103

C. V. Rao, S. J. Wright, and J. B. Rawlings. Application of interior-point methods to model predictive control. *Journal of optimization theory and applications*, 99(3): 723–757, 1998. 6

S. Richter, M. Morari, and C. N. Jones. Towards computational complexity certification for constrained mpc based on lagrange relaxation and the fast gradient method. In *Proc. 50th IEEE Conference on Decision and Control and European Control Conference*, pages 5223–5229, Dec 2011. 24

R. Rockafellar. *Convex Analysis*. Princeton University Press, 1970. 30, 114, 134

J. Roll, A. Bemporad, and L. Ljung. Identification of piecewise affine systems via mixed-integer programming. *Automatica*, 40(1):37–50, 2004. 3, 8, 80, 81

M. Rubagotti, P. Patrinos, A. Guiggiani, and A. Bemporad. Real-time model predictive control based on dual gradient projection: Theory and fixed-point FPGA implementation. *International Journal of Robust and Nonlinear Control*, 26 (15):3292–3310, 2016. 24

A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986. 9

H. D. Sherali and P. J. Driscoll. Evolution and state-of-the-art in integer programming. *Journal of Computational and Applied Mathematics*, 124(1):319–340, 2000. 9

V. Singhal, J. Maggu, and A. Majumdar. Simultaneous detection of multiple appliances from smart-meter measurements via multi-label consistent deep dictionary learning and deep transform learning. *IEEE Transactions on Smart Grid*, 2018. 103

D. Srinivasan, W. Ng, and A. Liew. Neural-network-based signature recognition for harmonic source identification. *IEEE Transactions on Power Delivery*, 21(1): 398–405, 2006. 103

B. Stellato and G. Banjac. OSQP: An operator splitting solver for quadratic programs. *GitHub*, 2017. URL `https://github.com/oxfordcontrol/osqp`. 46

B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: An operator splitting solver for quadratic programs. *ArXiv e-prints*, Nov. 2017. 11, 25, 40, 129

B. Stellato, V. V. Naik, A. Bemporad, P. Goulart, and S. Boyd. Embedded mixed-integer quadratic optimization using the OSQP solver. In *Proc. European Control Conference*, pages 1536–1541, Limassol, Cyprus, 2018.

R. A. Stubbs and S. Mehrotra. A branch-and-cut method for 0-1 mixed convex programming. *Mathematical Programming*, 86(3):515–532, Dec 1999. 9

K. Suzuki, S. Inagaki, T. Suzuki, H. Nakamura, and K. Ito. Nonintrusive appliance load monitoring based on integer programming. In *Proc. SICE Annual Conference*, pages 2742–2747, Japan, 2008. 9, 103

S. M. Tabatabaei, S. Dick, and W. Xu. Toward non-intrusive load monitoring via multi-label classification. *IEEE Transactions on Smart Grid*, 8(1):26–40, 2017. 103

R. Takapoui, N. Moehle, S. Boyd, and A. Bemporad. A simple effective heuristic for embedded mixed-integer quadratic programming. In *Proc. American Control Conference*, pages 5619–5625, Boston, MA, USA, 2016. 12, 34, 40, 42, 43

Z. Tian, W. Wu, and B. Zhang. A mixed integer quadratic programming model for topology identification in distribution network. *IEEE Trans. Power Systems*, 31(1):823–824, 2016. 4

F. Torrisi and A. Bemporad. HYSDEL — A tool for generating computational hybrid models. *IEEE Trans. Control Systems Technology*, 12(2):235–249, Mar. 2004. 5, 7, 62

F. Ullmann. FiOrdOs: A Matlab Toolbox for C-Code Generation for First Order Methods. Master's thesis, ETH Zurich, Switzerland, July 2011. 11

R. Vanderbei. Symmetric quasi-definite matrices. *SIAM Journal on Optimization*, 5(1):100–113, 1995. https://doi.org/10.1137/0805005. 41

J. P. Vielma, S. Ahmed, and G. L. Nemhauser. Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions. *Operations Research*, 58:303–315, 2010. 81

H. Witsenhausen. A class of hybrid-state continuous-time dynamic systems. *IEEE Trans. Automatic Control*, 11(2):161–167, April 1966. 4

F. M. Wittmann, J. C. Lpez, and M. J. Rider. Nonintrusive load monitoring algorithm using mixed-integer linear programming. *IEEE Transactions on Consumer Electronics*, 64(2):180–187, May 2018. 9, 103

S. J. Wright. Applying new optimization algorithms to model predictive control. In *AIChE Symposium Series*, pages 147–155. Citeseer, 1997. 6

X. Xu and P. J. Antsaklis. Results and perspectives on computational methods for optimal control of switched systems. In O. Maler and A. Pnueli, editors, *Hybrid Systems: Computation and Control*, pages 540–555, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. 4

G. Y. Yang, G. Hovland, R. Majumder, and Z. Y. Dong. TCSC allocation based on line flow based equations via mixed-integer programming. *IEEE Trans. Power Systems*, 22(4):2262–2269, 2007. 4

M. Zeifman and K. Roth. Nonintrusive appliance load monitoring: Review and outlook. *IEEE Transactions on Consumer Electronics*, 57(1):76–84, 2011. 103

A. Zoha, A. Gluhak, M. Imran, and S. Rajasegarar. Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey. *Sensors*, 12(12): 16838–16866, 2012. 103

Unless otherwise expressly stated, all original material of whatever nature created by Vihangkumar Vinaykumar Naik and included in this thesis, is licensed under a Creative Commons Attribution Non-commercial Share Alike 2.5 Italy License.

Check creativecommons.org/licenses/by-nc-sa/2.5/it/ for the legal code of the full license.

Ask the author about other uses.