IMPERIAL

Data-driven optimization strategies for interconnected process systems

Author

Damien van de Berg

Supervised by

Prof. Nilay Shah

Dr. Ehecatl Antonio del Rio-Chanona

In the Faculty of Engineering submitted in fulfillment of requirements for the degree of **Doctorate of Philosophy**

Department of Chemical Engineering

Imperial College London

2024

Abstract

This thesis explores the application of derivative-free optimization (DFO) and other data-driven approaches to complex, integrated decision-making in enterprise-wide optimization (EWO).

Chapter 1 sets the historical backdrop for optimization and Machine Learning (ML) tools in EWO and exposes their limitations through a value chain coordination example under different game-theoretical settings. Chapter 2 reviews dominant mathematical decision-making frameworks for the operation of chemical process systems. Mathematical optimization is introduced, along with various ways it can be enhanced by ML.

Chapter 3 presents and assesses different types of DFO solvers on process systems engineering (PSE) applications, setting the scene for subsequent work. A convex quadratic trust-region optimizer CUATRO is introduced, specifically designed to address the limitations of existing DFO solvers in PSE, focusing on safe exploration under noisy evaluations and black-box constraints.

Chapter 4 demonstrates how DFO can coordinate complicating variables linking subproblems in EWO, while preserving agent autonomy and privacy. Considered case studies include collaborative model training, facility location, and resource sharing problems. Data-driven coordination outperforms distributed optimization in nonconvex problems with few linking variables.

Chapter 5 shifts focus to hierarchical planning-scheduling-control. A data-driven approach is presented that combines DFO with optimality surrogates to enable the solution of previously intractable tri-level formulations. The proposed method balances tractability and accuracy, offering a viable alternative to heuristic sequential approaches.

Chapter 6 revisits CUATRO and its application to high-dimensional DFO problems. We learn linear subspaces to optimize over, reducing computational cost in the trust region updates. This approach matches the performance of state-of-the-art model-based solvers in over a hundred dimensions, with significant computational speed-up.

This thesis identifies specific scenarios in EWO, where DFO is effectively integrated with mathematical programming, by treating subproblem expressions as black-boxes, and coordinating their optimal outputs with DFO. To conclude, the contributions are discussed alongside promising directions for future research.

Declaration of Originality

I hereby declare that this thesis is the result of my own original work. To the best of my knowledge the work is original and ideas developed in collaboration with others have been appropriately referenced. Any ideas, techniques, or data derived from the work of others are fully acknowledged and appropriately referenced in the text. Portions of this thesis include work that has been previously published in peer-reviewed journals and conferences. Any adopted work has been incorporated with necessary permissions and is appropriately indicated at the beginning of each chapter.

Copyright Declaration

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-NonCommercial 4.0 International Licence (CC BY-NC).

Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work, provided that you credit the author and do not use it, or any derivative works, for a commercial purpose.

Where chapters or sections have been previously published, appropriate permissions have been obtained, and these are clearly indicated at the beginning of the relevant chapters. All other chapters or sections of the thesis are subject to the CC BY-NC licence without additional restrictions.

When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes.

Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

Acknowledgments

This thesis is dedicated to my supervisors and loved ones - family, friends, and partner. Their support and advice have been invaluable, helping me grow and making this journey possible. Thank you:

To Antonio: It was an honor to be among the first students in the group. I appreciate the close support you provided, especially during my first year marked by all kinds of uncertainties. I will fondly remember the technical discussions about Boyd and convex optimization, the extensive MSc and MEng supervision experiences, and your going out of your way to support my extracurricular growth at conferences.

To Nilay: I always looked forward to our meetings, which were true goldmines of incisive technical and strategic insight. Thank you for always making time when I needed it.

To the close-knit PSE group for the strong support that helped smooth over the rougher patches of the journey. I want to especially thank Ilya for his unwavering positive outlook and assistance; Max for the bouldering sessions and philosophical conversations about academia; Zhengang for ensuring mathematical rigor in my work; Akhil, Miguel, and Tom for the much needed levity during challenging times.

To Debora, Inga, Christian, and the BASF team for the warm reception in Ludwigshafen and at retreats, and for providing my first exposure to research in industry, grounding my work in real-world applications.

Most importantly, the PhD experience has been as much about personal as it has been about professional growth. Thank you to my undergrad friends for keeping me sane and reminding me that there is a life outside of academia.

To Canfer, for your endless patience and compassion. I feel truly understood by you. You give me strength and inspire me not only to be a better professional, but a better person.

U meng Elteren - Ech kann net a Wierder faasse wéi dankbar ech fir alles sinn. Et gëtt mer ëmmer méi bewosst, dass et net selbstverständlech ass, wéi vil dier mech ënnerstëtzt hutt, sou dass ech mech fir de Gros nëmmen hu missten op d'Schoul konzentréieren. Ech mengen net dass ech et ouni iech sou wäit gepackt hätt.

Contents

Abstract i

Declaration of Originality iii

Copyright Declaration v

Acknowledgments vii

List of Acronyms xv

List of Figures xvii

List of Tables xxiii

1 Overview 1

1.1 Setting the scene: A short history of PSE . . . 1

1.1.1 Process systems as complex networks . . . 1

1.1.2 Process systems engineering before Industry 4.0 . . . 2

1.1.3 The digital revolution and Machine Learning (ML) . . . 3

1.2 Motivation . . . 5

1.3 Contributions and structure . . . 8

1.4 Notation . . . 11

2 Background 13

2.1 Decision-making paradigms . . . 14

2.1.1 The Beer distribution game: an example from supply chain management . . . 14

2.1.2 The Knapsack problem: an example from operations research . . . 15

2.1.3 The linear-quadratic-Gaussian control problem . . . 17

2.1.4 Prevalence of mathematical programming in PSE . . . 19

2.2 Mathematical programming . . . 21

2.2.1 Duality . . . 21

2.2.2 Decomposition . . . 24

2.2.3 Mathematical programming solvers . . . 27

2.3 Machine Learning (ML) . . . 35
2.3.1 Paradigms . . . 35
2.3.2 Recent wave of ML advances . . . 36
2.3.3 ML for optimization . . . 36
2.3.4 How Machine Learning and optimization are used in this thesis . . . 43

3 Derivative-free optimization for process systems engineering applications 45
3.1 Introduction . . . 46
3.1.1 Categorization of derivative-free optimization solvers . . . 47
3.1.2 Previous work in the chemical engineering community . . . 48
3.1.3 Our contribution . . . 49
3.2 Methodology . . . 50
3.2.1 Problem statement . . . 50
3.2.2 Steepest descent via finite difference approximations . . . 50
3.2.3 Py-BOBYQA . . . 52
3.2.4 Nesterov random Search . . . 53
3.2.5 SQSnobfit . . . 54
3.2.6 DIRECT-L randomized . . . 55
3.2.7 Simplex method . . . 55
3.2.8 Bayesian Optimization (BO) . . . 56
3.2.9 Constraint handling . . . 58
3.3 CUATRO . . . 58
3.3.1 Surrogate fitting . . . 59
3.3.2 Explicit constraint handling . . . 60
3.3.3 Surrogate minimization . . . 61
3.4 Case Studies . . . 63
3.4.1 Problem considerations and selection of case studies . . . 63
3.4.2 Constrained Rosenbrock . . . 64
3.4.3 Real-time optimization . . . 64
3.4.4 Model-based design of experiments . . . 66
3.4.5 Self-optimizing reactions . . . 67
3.4.6 Controller synthesis . . . 68
3.4.7 Flowsheet optimization . . . 71
3.5 Results and Discussion . . . 73

3.5.1 Algorithms . . . 73
3.5.2 Performance of DFO algorithms on the case studies . . . 78
3.5.3 Performance of DFO algorithms at increasing levels of stochasticity . . . 87
3.5.4 Constraint Satisfaction . . . 90
3.6 Conclusion . . . 94
3.6.1 Looking Back . . . 95

4 Coordinating value chain planning under organizational considerations 97
4.1 Introduction . . . 98
4.1.1 Decomposition and distributed optimization . . . 99
4.1.2 Organizational considerations and limitations . . . 100
4.1.3 Our contribution . . . 101
4.2 Methodology . . . 102
4.2.1 Problem statement . . . 102
4.2.2 Problem reformulation . . . 102
4.2.3 ADMM by consensus . . . 103
4.2.4 Data-driven coordination . . . 105
4.2.5 Algorithms and software implementation . . . 111
4.2.6 Game-theoretical and other considerations . . . 112
4.3 Case studies . . . 112
4.3.1 Convergence plots . . . 113
4.3.2 Motivating example . . . 113
4.3.3 Collaborative model training . . . 119
4.3.4 Facility location . . . 123
4.3.5 Multi-objective coordination . . . 128
4.4 General observations and discussion . . . 132
4.5 Conclusion . . . 135
4.5.1 Looking back . . . 136

5 Hierarchical planning-scheduling-control 139
5.1 Introduction . . . 140
5.1.1 Relaxation or aggregation techniuqes . . . 141
5.1.2 Decomposition techniques . . . 141
5.1.3 The impact of data-driven techniques . . . 142

5.1.4 Our contribution . . . 142
5.1.5 Related works . . . 144
5.2 Case study . . . 148
5.2.1 Planning . . . 149
5.2.2 Scheduling . . . 149
5.2.3 Control . . . 150
5.2.4 Integration . . . 151
5.3 Methodology . . . 152
5.3.1 Using data to solve a canonical bi-level problem . . . 153
5.3.2 Hierarchical planning-scheduling-control as tri-level formulation . . . 156
5.3.3 Metrics . . . 157
5.3.4 Partitioning into DFO variables and lower-level scheduling-control problems 158
5.3.5 Methodology 1: Derivative-free optimization . . . 161
5.3.6 Methodology 2: Optimality surrogates . . . 162
5.3.7 Methodology 3: Combining both approaches . . . 163
5.3.8 Software and practical considerations . . . 163
5.4 Results and Discussion . . . 164
5.4.1 Methodology 1: Integration into planning via DFO . . . 165
5.4.2 Methodology 2: Integration into planning via surrogates . . . 167
5.4.3 Comparing the DFO and surrogate approaches . . . 171
5.4.4 Methodology 3: Bringing out the best in both approaches . . . 172
5.5 Conclusion . . . 176
5.5.1 Looking back . . . 177

6 Scaling to higher dimensions via trust region optimization in informative linear embeddings 179

6.1 Sampling heuristics for improving trust-region DFO . . . 180
6.1.1 Exploration . . . 180
6.1.2 Constraint satisfaction . . . 183
6.2 Extension to high dimensions . . . 184
6.2.1 Introduction . . . 185
6.2.2 Methodology . . . 186
6.2.3 Case studies . . . 191
6.2.4 Results and discussion . . . 194

6.3 Conclusion . . . 196
6.3.1 Looking back and ahead . . . 196

Conclusions **199**

6.3.2 Retrospective . . . 202
6.3.3 Future work . . . 203

A Supplementary information for Chapter 3 **257**

A.1 Finite difference approximation algorithms . . . 257
A.1.1 Finite Difference Approximation . . . 257
A.1.2 Newton's method via finite differences . . . 258
A.1.3 BFGS via finite differences . . . 259
A.1.4 Adam via finite differences . . . 260
A.2 Gaussian Processes . . . 260
A.3 Algorithm convergence . . . 262
A.4 Self-Optimizing reactor parameters of system . . . 263

B Supplementary information for Chapter 4 **265**

B.1 Distributed optimization . . . 265
B.1.1 Method of Multipliers . . . 265
B.1.2 ADMM . . . 266
B.1.3 ALADIN . . . 267
B.2 Facility Location case study . . . 267
B.2.1 Decomposition . . . 269

C Supplementary information for Chapter 5 **273**

C.1 Optimization formulations . . . 273
C.1.1 Planning formulation . . . 273
C.1.2 Scheduling formulation . . . 275
C.1.3 Control formulation . . . 278
C.2 Supplementary results . . . 279
C.2.1 Integrating scheduling and control . . . 279
C.2.2 Optimality surrogate training samples . . . 280
C.2.3 Surrogate sensitivity analysis on bi-level integrated planning-scheduling problem . . . 280
C.2.4 Qualitative discussion on scaling with relative variable size . . . 286

C.2.5 Benchmarking on perturbed problem instances 286
C.3 Planning profiles for best solutions 286

D Supplementary information for Chapter 6 **293**
D.1 Formulations 293
D.1.1 PID controller tuning 293

List of Acronyms

ADMM Alternating Direction Method of Multipliers

AGI Artificial General Intelligence

AI Artificial Intelligence

AL Active Learning

ADP Approximate Dynamic Programming

BBO Black-Box Optimization

BO Bayesian Optimization

CO Combinatorial Optimization

DFO Derivative-Free Optimization

DP Dynamic Programming

EWO Enterprise-Wide Optimization

GenAI Generative Artificial Intelligence

GP Gaussian processes

IP Integer Programming

iPSC integrated Planning-Scheduling-Control

IL Imitation Learning

KP Knapsack Problem

LLMs Large Language Models

LP Linear Programming

LQG Linear-Quadratic-Gaussian

MAS Multi-Agent System

MDP Markov Decision Process

MILP Mixed-Integer Linear Programming

MINLP Mixed-Integer Nonlinear Programming

MIP Mixed-Integer Programming

ML Machine Learning

MPC Model Predictive Control

NLP Nonlinear Programming

OR Operations Research

PLS Partial Least Squares

PSE Process Systems Engineering

RL Reinforcement Learning

SCM Supply Chain Management

SO Simulation optimization

SOTA State-Of-The-Art

SQP Sequential Quadratic Programming

TSP Travelling Salesman Problem

List of Figures

1.1 Motivating multi-site, multi-product value chain. S1-S3 denote three sites, where raw material RM is converted to intermediates I1, I2, I21, I22, and products P1-P6, before being sold to markets M2-M3. 6

1.2 Examples of supply chain decision-making under organizational constraints 7

2.1 Beer distribution game schematic 14

2.2 Constraint-Variable matrix for subproblem interactions 25

3.1 The four basic simplex operations: reflect, shrink, contract and expand. 55

3.2 Illustration of Gaussian processes (GP) regression and expected improvement acquisition function of a sampled function. 57

3.3 Two-dimensional search space convergence plot for the best methods (3.3b) from the starting point towards the optimum for the a) real-time optimization case study, and the b) self-optimizing reactions case study. 69

3.4 The CryoMan cascade refrigeration cycle as implemented in Aspen HYSYS [10]. . 72

3.5 Two-dimensional solution space convergence on the deterministic constrained Rosenbrock function from starting point $[-0.5, 1.5]^\intercal$ 74

3.6 Convergence plots showing 10 best function evaluation versus number of function evaluation runs for all methods on the constrained Rosenbrock test function without and with stochasticity. 80

3.7 Convergence plots showing 10 best function evaluation versus number of function evaluation runs for all methods on the real-time optimization case study without and with stochasticity. 81

3.8 Convergence plots showing 10 best function evaluation versus number of function evaluations runs for all methods on the self-optimizing reactions case study without and with stochasticity 82

3.9 Convergence plots showing 10 best function evaluation versus number of function evaluations runs for all methods on the deterministic model-based design of experiments case study 83

3.10 State (3.10a) and control (3.10b) trajectories corresponding to the initial and the median optimal DIRECT, CUATRO global and CUATRO local PI controller gains for the deterministic PI controller tuning case study. Evolution of the state (3.10c) and control (3.10d) trajectories corresponding to all sampled median CUATRO local PI controller tunings from the initial towards the best. 84

3.11 Convergence plots showing 10 best function evaluation versus number of function evaluation runs for all methods on the PI controller tuning case study without and with stochasticity. The asterisk in the label for Snobfit denotes that not all of its 10 runs were successful. 85

3.12 State and control trajectories corresponding to the initial and best DIRECT, CUATRO global, and CUATRO local controller gains for the deterministic quadratic feedback regulator case study. 86

3.13 Convergence plots showing 10 best function evaluation versus number of function evaluations runs for all methods on the stochastic quadratic control law synthesis case study. The asterisk in the label for Snobfit denotes that not all of its 10 runs were successful. 87

3.14 Convergence plots showing the 10 best function evaluation versus number of function evaluation runs for all methods on the deterministic flowsheeting case study 87

3.15 Stochasticity convergence plots for the best methods per noisy case study: For each of the six stochasticity levels, the difference is shown between the best optimum found in a) 100, b) 50, c) 50, d) 100, or e) 100 function evaluations for each of the 20 runs and the best optimum found among all methods for said stochasticity level. The stochasticity levels refer to the noise standard deviations of the normally distributed noise added to each stochastic case study as discussed in Section 3.4 . . 89

3.16 Constrained Rosenbrock convergence and proportion of constraint violation plots for PyBOBYQA, SQSnobfit, DIRECT, CUATROg and Bayesian opt. without and with sample average approximation at the same total evaluation budget. 93

4.1 We can solve an equivalent centralized formulation either by ADMM or data-driven coordination (⇓). In either case, a coordinator (□) iteratively sends an updated proposed set of shared variables $\mathbf{z}^k$ (↓) to all coordinating agents . The coordinating agents (△) then optimize their private objective according to (*) or (**) for ADMM and data-driven coordination respectively. In ADMM, the subproblems return the optimal local copies of the proposed shared variables $\mathbf{z}_a^{k+1}$, whereas in our approach they return the optimal subproblem objective evaluations y_a^{k+1} (↑). In both frameworks, the penalty parameter ρ determines the extent to which the deviation between the suggested $\mathbf{z}^k$ and optimal set of private variables $\mathbf{z}_a^{k+1}$ is penalized. ADMM's proximal term also includes scaled dual variables λ_a^k, which serve as 'integral error terms' between $\mathbf{z}$ and $\mathbf{z}_a$. In the coordination step, ADMM updates the proposed set of shared variables $\mathbf{z}^{k+1}$ by averaging its local copies $\mathbf{z}_a^{k+1}$, while our data-driven framework uses derivative-free optimization (DFO) to determine $\mathbf{z}^{k+1}$ based on shared variable $\mathbf{z}^{k+1}$ to the sum of optimal subproblem evaluation y_a^{k+1} data. In this case, we only show 2 coordinating agents, but this scheme can be generalized to any number of coordinating agents. 105

4.2 Classification of considered algorithms with associated problem formulations in parentheses: We can decide to solve our equivalent centralised formulation either via distributed optimization, namely ADMM, or via our proposed data-driven coordination framework. Within our framework, we can choose any derivative-free optimization (DFO) algorithm. We choose DIRECT-L as a direct DFO algorithm. Among model-based DFO algorithms, we consider Py-BOBYQA and CUATRO as quadratic trust region frameworks and GPyOpt as Bayesian optimization. We also distinguish between ADMM_CUATRO and ALADIN_CUATRO in the way quadratic surrogates are formulated. Each CUATRO version also has the choice of explicit constraint satisfaction 108

4.3 'Black-box' objective $F(z)$ of the motivating example as a function of the shared variable z. 115

4.4 Convergence plots for the motivating example starting at $z = 4.5$ with $\rho = 1,000$. For DIRECT-L and GPyOpt, the median best evaluation with shaded min-max range is given over 5 runs 116

4.5 Motivating example: Best function evaluation versus number of function evaluations convergence plots under different configurations. For DIRECT-L and GPyOpt, the median best evaluation with shaded min-max range is given over 5 runs 117

4.6 Truncated regression case study: Best function evaluation versus evaluation budget and convergence versus evaluation budget plots at a fixed number of agents ($n_a = 2$), and an increasing number of shared variables ($n_z = 2, 10, 50$). For DIRECT-L and GPyOpt, the median best evaluation with shaded min-max range is given over 5 runs122

4.7 Truncated regression case study: Best function evaluation versus evaluation budget and convergence versus evaluation budget plots at an increasing number of agents ($n_a = 2, 4, 8$), and a fixed number of shared variables ($n_z = 6$). For DIRECT-L and GPyOpt, the median best evaluation with shaded min-max range is given over 5 runs124

4.8 Facility location convergence plots when increasing the number of processing facilities ($n_p = 1, 2$) and shared variable dimensionality ($n_z = 3, 6$) between two agents ($n_a = 2$). For DIRECT-L and GPyOpt, the median best evaluation with shaded min-max range is given over 5 runs 126

4.9 Facility location convergence plots when increasing the number of processing facilities ($n_p = 1, 2$) and shared variable dimensionality ($n_z = 5, 10$) between four agents ($n_a = 4$). For DIRECT-L and GPyOpt, the median best evaluation with shaded min-max range is given over 5 runs. 127

4.10 Facility location convergence plots with two agents ($n_a = 2$) and two processing facilities ($n_p = 2$ and $n_z = 6$) when increasing the number of supplier and customer nodes per subproblem from two to five ($n_s = n_c = 2, 3, 4, 5$). For DIRECT-L and GPyOpt, the median best evaluation with shaded min-max range is given over 5 runs129

4.11 Multi-objective coordination case study: best function evaluation versus evaluation budget and convergence versus evaluation budget plots for the convex and nonconvex variations with an increasing number of shared variables ($n_z = 10, 25$). For DIRECT-L and GPyOpt, the median best evaluation with shaded min-max range is given over 5 runs 133

5.1 Hierarchical decision-making architectures: In the sequential approach, higher-level solutions feed into lower levels as setpoints. Each level solves their own optimization problem to meet these setpoints without explicit considerations of the lower levels. The monolithic approach resembles the sequential approach, but feasibility in lower levels is satisfied via explicit incorporation of their constraints. The hierarchical approach could be viewed as a multi-level leader-follower game. A single 'tri-level' optimization problem is solved, where the planning is subject to the optimal solution of the scheduling level, and the scheduling is constrained on the optimal solution of the control level. 143

5.2 Outline of the hierarchical A) planning, B) scheduling, C) and control levels in the multi-site, multi-product case study. A) shows the planning state network denoting the materials ($\circ$) as well as processing ($\rightarrow$), transport ($\rightarrow$), and sales ($\rightarrow$) across the three sites ($\square$). Scheduling and control are only relevant to the third site. B) shows the state network of the scheduling problem that includes an additional intermediate material. The production of I3 and P1-P4 needs to be scheduled in batches on two machines in seven event time points. C) Each batch production is then subject to an optimal control problem. 148

5.3 Linking variables between the planning, scheduling, and control levels. Each planning problem feeds the planning targets for its 12 timesteps to separate scheduling problems to be solved in parallel. Each of the 12 scheduling problems, in deciding which of the 5 productions to schedule on the 2 machines at the 7 events, obtains the optimal batch processing time and cost as a function of the batch size targets for each of the 70 possible batch assignments. The optimal scheduling and control finally feed back into the planning. 152

5.4 Increasing levels of solution accuracy evaluation. (8a)-(8d) refer to evaluation metrics (5.8b)-(5.8d). The upper or planning-only evaluation computes the planning objective and constraints in a simulation. The bi evaluation also solves 12 scheduling problems in parallel in each evaluation to account for the scheduling cost in the objective. The approximate tri evaluation additionally embeds control surrogates into the scheduling to account for approximate control cost in the objective. The tri evaluation solves scheduling and control levels in sequence to emulate hierarchical decision-making and adds the 'real' hierarchical scheduling and control costs to the planning objective. 158

5.5 The three proposed methodologies: References to formulations as (X) refer to (5.X). The first methodology uses DFO to optimize any of the solution metrics in (5.8a)-(5.8d) directly by considering the planning simulation and scheduling-control optimization as a black-box. The second methodology uses hyperparameter optimization to find a surrogate architecture that, trained on one of the solution metrics in (5.8b)-(5.8d), leads to the best solution accuracy-time trade-off after embedding into the planning. The third methodology combines both workflows in using DFO to optimize the solutions obtained from the surrogate approach. 159

5.6 Solution accuracy and time for three methods: solution initialization using planning-only metric (blue), DFO on the bi metric (orange), and DFO on the approx tri metric (green). The solution accuracy bar plots denote the evaluation (y-axis) that the three methods (blue, orange, and green) achieve on the four solution quality metrics (x-axis) corresponding to (5.8a)-(5.8d). For example, using DFO with the bi solution metric as objective function (orange) outperforms DFO approx (green) on the bi solution metric (second bin). However, DFO approx (green) outperforms DFO bi (orange) on the approx solution metric (third bin). All three considered approaches are infeasible in the tri solution metric (fourth bin), and consequently heavily penalized, leading to a bad evaluation. 167

5.7 Solution accuracy and time of the best optimality surrogate solutions: the first three surrogate architectures are trained on the scheduling-only (bi: blue, orange, green), and the other three (approx: red, purple, brown) on the approximate scheduling-control subproblems. The two numbers in the label refer to the number of nodes in the two hidden layers. For example, all 6 surrogate approaches lead to a comparable solution quality at -4 in the planning-only solution metric (first bin). All three surrogates trained on the approx solution quality metric (red, purple, brown) achieve a similar performance around -2 on the approx solution quality metric (third bin), whereas the surrogate with 42 and 20 nodes (orange) is the only surrogate out of the three trained on the bi solution metric that achieves a good evaluation on the approx solution metric around -2 rather than 7. The surrogate with 45 and 2 nodes respectively (red) achieves the best performance on the tri solution quality (fourth bin). 169

5.8 Box-and-whisker plot of solution accuracy variability for all queried surrogates in the hyperparameter tuning: Surrogates trained on scheduling-only (bi: blue), approximate scheduling-control (approx: orange), and accurate scheduling-control (tri: green) information are compared on upper, bi, approx tri, and tri solution metrics from left to right. The box represents the quartiles, while the whiskers demonstrate variability outside the box plots. Outliers are shown as isolated data points outside of the whiskers. 170

5.9 Increase in solution quality and time when using DFO with the tri metric (orange) on the best solution (blue) of other methods. The solution approaches, included as initial guess, from left to right, are: planning-only initialization, DFO on the bi and approx tri metrics, and the best surrogates trained on the bi and approx tri metrics. For example, fine-tuning the solution obtained from the planning-only solution (first bin) increases the tri solution metric from about 9.5 to about 9.0. Fine-tuning the solution obtained from the surrogate trained on the bi solution metric improves the evaluation obtained on the tri solution metric (fourth bin) from about -0.5 to below -2. 175

6.1 Two-dimensional solution space convergence plots of CUATRO variations on the Himmelblau function. Black crosses denote starting points. Early samples are blue, later samples red. 182

6.2 Exploration (y-axis: Average distance to nearest neighbours) versus exploitation (x-axis: Optimum) on the Himmelblau function: The higher the distance to the nearest neighbors, the more explorative. Blue samples represent CUATRO variations, and orange samples represent reference solvers 182

6.3 Two-dimensional solution space convergence plots of CUATRO variations on the constrained, deterministic real-time optimization case study from Section 3.4.3 . . 184

6.4 Average constraint violation versus average best optimum found for each algorithm on the deterministic real-time optimization case study from Section 3.4.3. Blue samples represent CUATRO variations, and orange samples reference solvers . . . 184

6.5 Regret versus evaluation budget convergence plot for the synthetic high-dimensional Rosenbrock case study at constant effective dimensionality $n_e = 2$, and ambient dimensionality $n_h = 100$ (top) and $1,000$ (bottom) for model-based (left) and direct search (center) DFO methods. The corresponding solution times are displayed on the right. Reprinted from van de Berg et al. [388] with permission from Elsevier. . 195

6.6 Regret versus evaluation budget convergence plot on the 1000-dimensional Rosenbrock with an effective dimensionality of ten ($n_e = 10$), when CUATRO_PLS is used with different embedding dimensionalities ($n_{PLS} = 2, 6, 10, 20, 40$ in the labels) and other hyperparameters are fixed to their default. Reprinted from van de Berg et al. [388] with permission from Elsevier. 196

6.7 Best function evaluation versus evaluation budget convergence plots on high-dimensional chemical engineering case studies. Left: CUATRO with different hyperparameters (embedding dimensionality, number of trust region samples, and trust region initial radius). Right: Other competitive solvers. Reprinted from van de Berg et al. [388] with permission from Elsevier. 197

C.1 Predicted and actual optimal batch processing time and energy requirements corresponding to 10 uniformly sampled training and test batch targets. 279

C.2 Optimal scheduling cost versus planning target samples for optimality surrogate training. 281

C.3 Pair-wise plots for the feasible samples used for the training of the scheduling-only optimality surrogates. 282

C.4 Pair-wise plots for the feasible samples used for the training of the integrated scheduling-control optimality surrogates. 283

C.5 Pair-wise plots for the feasible samples used for the training of the hierarchical scheduling-control optimality surrogates. 284

C.6 Pairwise key performance indicators as we increase the size of the surrogates in solving an integrated planning-scheduling problem 285

C.7 Comparing the best solutions from the three methodologies on perturbed instances 286

C.8 Solution dashboard and planning profile of the planning-only problem 287

List of Tables

3.1 Case studies and their mathematical properties 63

3.2 MBDoE case study: definition of design space. 67

3.3 Average convergence-safe exploration trade-off on deterministic, constrained case studies 92

4.1 Performance comparison between ADMM and data-driven coordination based on mathematical problem and desired solution characteristics 132

5.1 Summary of experiments performed 165

5.2 Online optimization, and offline hyperparameter tuning, training, and sampling times associated with the construction of surrogates in the integration of scheduling-only, approximate scheduling-control, and accurate scheduling-control lower-level information 172

A.1 Self-optimizing case study: parameters of 3.19. 263

C.1 Online optimization, control surrogate sampling and surrogate training times involved in the solution of the scheduling-only and scheduling with embedded control surrogates problems 280

1 Overview

Contents

1.1 Setting the scene: A short history of PSE 1
1.1.1 Process systems as complex networks 1
1.1.2 Process systems engineering before Industry 4.0 2
1.1.3 The digital revolution and Machine Learning (ML) 3
1.2 Motivation 5
1.3 Contributions and structure 8
1.4 Notation 11

This thesis investigates how data-driven methods - especially black-box optimization - can be used to integrate manufacturing and supply chain operations under organizational considerations. This chapter starts off with a brief history of the Process Systems Engineering (PSE) field that sets the backdrop for this thesis. This introduction is followed by the description of a planning case study under different enterprise constraints. Finally, the structure of the remaining thesis is presented: in each chapter, we investigate how a different business scenario informs the development of distinct data-driven methodologies.

1.1 Setting the scene: A short history of PSE

1.1.1 Process systems as complex networks

The process industries are made up of a complex web of interconnected material, energy, and information flows [416]. Simon [349] characterize systems to be complex when they consist of

"numerous parts that interact in nonsimple ways". This definition is later extended to include non-linearities and asymmetries [27] . Simply put, the emerging system behaviour is difficult to predict from the isolated behaviour of the constituent parts. Unsurprisingly, manufacturing and supply chain systems are often approached from the angle of complex adaptive networks, i.e. dynamic networks where many agents act simultaneously and continuously react to the actions of other agents [117, 54]. While there is no consensus on the definition of an agent, agents are essentially considered to be: situated within an environment, autonomous in their actions, perceptive of their environment (able to sense), communicating with other agents or people, reactive to changes in the environment, and capable of influencing the actions of others [173]. Optionally, agents can be adaptive, pro-active, and rational. They can learn from their past interactions with their environment, coordinate, negotiate, collaborate, compete to reach a set goal, and plan their actions to maximize their objective. In the framework of Multi-Agent System (MAS), these networks are modelled using a 'bottom-up' approach. By leveraging object-oriented programming and distributed artificial intelligence, the emergent system behaviour is observed corresponding to a given MAS architecture (e.g. interaction protocols) or set of agent policies. Under this framework, we could consider agents to be placed within the chemical industry anywhere from software to humans - from PI controllers to strategic planners.

1.1.2 Process systems engineering before Industry 4.0

Sargent [327] lay the ground for a paradigm shift in the "development of systematic techniques for process modelling, design, operations and control". They advocate for a more "sophisticated analysis of larger complexes" and interdisplinary collaboration with control engineering, operational research, numerical analysis, and computer science. Developments in PSE have been closely linked to developments in computing [326]. The increase in computing power and the developments in numerical methods allowed for the creation of powerful models whose optimization leads to better decisions than their heuristic- or expert-informed counterparts. Paired with the inclusion of operations problems such as scheduling and planning into the chemical engineering domain, this has led to an interest in mathematical optimization techniques within the PSE and the related field of Computer Aided Chemical Engineering. Consequently, the chemical engineering community developed many state-of-the-art optimization solvers, which now also find wide-spread use outside of PSE [154, 394, 368].

Chemical companies rely on mathematical optimization for their process operations to remain

competitive in an environment of increasingly stringent safety, environmental, and economic requirements. This gives rise to the field of Enterprise-Wide Optimization (EWO) [152, 153, 150]. EWO aims to optimize *all* decision-making across supply chains. This encompasses the integration and coordination of decision-making across the various functions in a company (sourcing, manufacturing, distribution), across geographically distributed organizations (plants, markets), and across decision-making levels (strategic planning to tactical scheduling and operational control). However, the centralized coordination and optimization of interconnected manufacturing and supply chain systems becomes intractable since rigorous accounting of uncertainty and decision-making timescales leads to an explosion in the decision space. Enterprise-wide optimization will be discussed in more detail in Section 1.2.

1.1.3 The digital revolution and Machine Learning (ML)

Digitalization and the related concepts of smart manufacturing, Industry 4.0 and cyber-physical systems [176] have become increasingly relevant to the chemical and process industries and have been declared key strategies for the next industrial revolution [299]. The emergence of these phenomena is enabled by various digital solutions: smart measuring devices, process analytical technology and sensor technologies that give rise to comprehensive and abundant process data. This data is intended to be shared with all subsystems using internet and intranet connectivity such as the cloud and the Industrial Internet of Things (IIoT) [53]. This is where algorithms must rise to the challenge, and advances in optimization, Artificial Intelligence (AI) and Machine Learning (ML) become key for automated control and decision-making [44].

At the heart of process and supply chain operations lies optimization. The traditional optimization pipeline in chemical engineering requires an existing model. This model is built using an iterative process of leveraging *a priori* physical, chemical, or biological knowledge to propose plausible parameterized models. The adequacy of the model depends on the statistical significance of the estimated parameters and can be improved upon using model-based design of experiments, which includes model discrimination when presented with multiple plausible models [18]. This model can then be exploited within traditional mathematical programming optimization frameworks. However, the construction of these models is often restricted by a lack of mechanistic knowledge about the system at hand, or by a prohibitively expensive model-building process. 'Data-driven' or ML models present a useful way to circumvent these restrictions.

Early on, computers in chemical engineering were primarily used as tools for performing quanti-

tative calculations, essentially serving as advanced calculators. Stephanopoulos [360], introducing the expert systems era of AI in chemical engineering, stated in the early 1990s that using qualitative or semi-qualitative knowledge-based expert systems outperformed purely quantitative models of human decision-making. These expert systems focus on how to represent knowledge from different domains to be used as starting assumptions in imitating human decision-making through a combination of conjectures (logical statements such as IF..THEN) and quantitative models [359]. In parallel, a different branch of AI in the form of ML investigated algorithms that learn explicitly from data. Breakthroughs in deep learning and Reinforcement Learning (RL) such as the software of AlphaGo mastering the game of Go [346] have renewed interest in ML within the chemical engineering community. In fact, the computational power and volumes of data were not available for the successful adoption of ML algorithms when they were first investigated for chemical engineering applications [392]. However, the phenomenon of digitalization, and the emergence of abundant, comprehensive data ('big data') cast doubt upon this commonly accepted belief.

Lee et al. [230] were among the first to review ML applications in PSE. ML is generally categorized into supervised learning, unsupervised learning, and RL. Supervised learning uses labelled input-output data with the aim to train a model to predict the output given the input features. This framework essentially boils down to 'modelling', and has been investigated in many disciplines, from establishing empirical correlations to predict thermodynamic properties or coefficients in transport phenomena [232], to system identification [331] in the control literature, to surrogate-, meta- or black-box modelling [398, 38, 183] in chemical engineering. Unsupervised learning aims to 'find structure in unlabelled data', to learn about their underlying distribution. Chemical engineering has historically addressed the same problem by using dimensionality reduction techniques e.g. in process analytics and chemoinformatics [243]. Kramer [218] proposed autoencoder-like structures, combining both paradigms by first 'compressing' the input data into a lower-dimensional space before predicting an output of interest. This reduced order modeling also plays a key role in fault identification [230]. Finally, in RL, an agent interacts repeatedly with a dynamic environment to learn an optimal policy to maximize a desired reward signal [344]. In essence, the task of RL boils down to an optimal control problem. For example, the Q-Learning algorithm in RL shares similarities with Approximate Dynamic Programming (ADP). All of these methods can be loosely categorized to be 'data-driven' as opposed to their mechanistic counterparts.

Since the initial efforts to apply ML to chemical engineering, the field has seen significant progress. Advances in Generative Artificial Intelligence (GenAI) [30, 106], such as transformers and their use in Large Language Models (LLMs) [58], have shifted public perception and bolstered

the narrative that AI will eventually revolutionize many sectors of society. Meanwhile, 'traditional' ML techniques, such as graph and recurrent neural networks for property prediction and dynamic systems modeling, have become commonplace in the PSE literature [165]. Entire sessions are now devoted to Machine Learning at conferences like the *American Institute of Chemical Engineers Annual Meeting* and the *European Symposium on Computer Aided Process Engineering* (ESCAPE) [248].

Despite the widespread acclaim for AI's transformative potential, it is important to also appreciate its drawbacks. A purely data-driven approach to decision-making often overlooks valuable mechanistic and engineering knowledge, which can undermine trustworthiness, especially in safety-critical applications [344, 230]. Moreover, ML techniquess typically require large volumes of high-quality data. This can be difficult for chemical engineering applications, where passive data collection from operational processes is often expensive or uninformative [371]. The question remains if current ML techniques have the maturity - or can be improved - to complement or even replace current decision-making tools in manufacturing and supply chain systems.

We reopen the question of how far we are from achieving the holy grail of EWO, i.e. the model-based integration of all decision-making within a chemical company. There are many aspects that render the integrated problem numerically intractable. In this thesis, rather than leverage data-driven techniques purely to speed up optimization solvers, we explore how these methods can help with some of the organizational and game-theoretical constraints decision-makers face.

1.2 Motivation

Special challenges arise in EWO when the organizational setting and decision-making architecture are considered. We start with a simple multi-site, multi-product value chain as depicted in Figure 1.1. In site S1, raw material RM undergoes a series of intermediate processing steps to I2. I2 is then transported to sites S2 and S3, where further processing happens to products P1-P6. The products are then sold to meet corresponding demand in markets M2 and M3. Optimal inventory planning could in principle be determined by solving a linear programming formulation. Next, we discuss the more interesting challenges of coordinating decisions when the decision-makers are subject to operational constraints such as privacy or game-theoretical considerations [250].

1

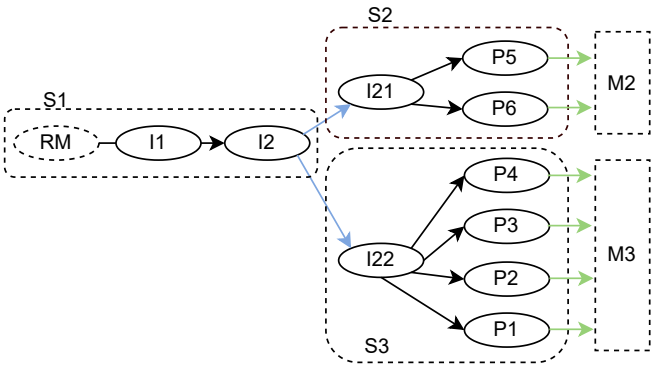


Figure 1.1: Motivating multi-site, multi-product value chain. S1-S3 denote three sites, where raw material RM is converted to intermediates I1, I2, I21, I22, and products P1-P6, before being sold to markets M2-M3.

Value chain coordination

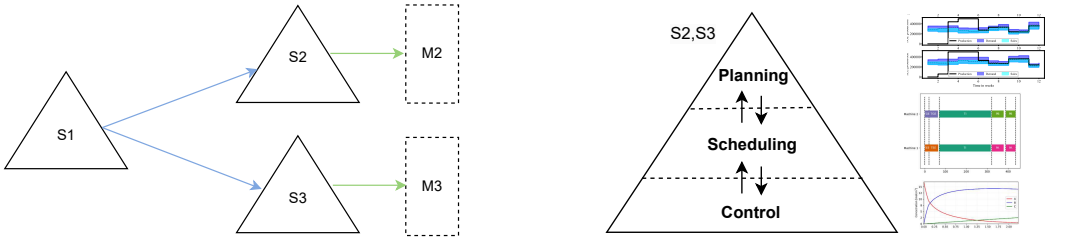
Even within the same company, different plants and sites might be operated by separate business entities. Decision architectures are often shaped by information-sharing constraints imposed by the organizational and legal rules that dictate communication between these entities [255]. Consequently, decision-making is often relegated to local, competing 'agents' instead of a centralized coordinator.

Each agent's decisions influence the behaviour of all others. While all players have autonomy over some decisions that only affect their operations, they need to agree on shared variables that concern multiple agents. As an example, when decision-makers agree on the material flow quantity and quality, it is up to each agent to decide *how* they make it happen (which production happens on what machine under what conditions). While it is common to use advanced tools for the 'local' decisions within the same entity, the shared variables are usually agreed upon following heuristic negotiation protocols [316].

However, not all agents have equal negotiating power due to information assymetry. Some players might for example exert their power of the first move to clear inventories before the end of the financial quarter and maximize their team's financial performance instead of the company's total environmental or economic objectives.

Figure 1.2a illustrates the problem of value chain coordination, where we need to agree on the 'shared' variables (material transport) while respecting the decision-making autonomy and privacy of the participating plants. It is crucial for solution methods to respect these organizational 'rules of

the game' to get the buy-in from participating agents in moving away from greedy 'Nash equilibria' towards Pareto-optimal solutions.



(a) Value chain coordination (b) Hierarchical planning-scheduling-control

Figure 1.2: Examples of supply chain decision-making under organizational constraints

Hierarchical decision-making

While value chain coordination could be considered integration between geographically distributed agents, we can also consider the integration of all decision layers within a specific site. According to the ISA-95 [358], production systems are split into different hierarchical levels. Figure 1.2b considers the hierarchical planning-scheduling-control problem within site S3, highlighting the intricate interactions among various decision types beyond planning.

Traditionally, the decisions of higher levels feed into the lower levels. While upper levels usually face simpler formulations under considerable uncertainty, lower levels typically face more complicated, detailed formulations under short time horizons. This sequential approach uses one-way information flow [87]. For example, at the highest level, supply chain management determines the demand orders to be fulfilled by a company's assets (monthly to yearly). For each asset, a daily or weekly production plan needs to be found that satisfies demand. Scheduling then determines the resource allocation or production sequence necessary to fulfil the plan on an hourly basis. Depending on the complexity of the control systems and their degrees of freedom, real-time optimization can be used to find control setpoints that maximize an economic objective, i.e. to operate batches at maximal efficiency or minimal cost. Finally, actuators take operating setpoints from real-time optimization or default recipes, and reject disturbances at a (milli)second scale for quality assurance and safety.

However, since in the sequential approach upper levels do not explicitly account for lower-level problem considerations, set-points fed to the lower levels might not even be feasible, let alone optimal. Given the complexity of a fully integrated approach, it is common for decision-makers to

potentially sacrifice solution accuracy for tractability, and use heuristics in the modelling of agent interactions.

In this thesis, we focus only on the planning, scheduling, and control levels. Although the sequential method is commonly used in practice, a game-theoretical approach would capture the closed-loop behavior of the integrated planning-scheduling-control problem most accurately as a multi-level optimization problem [87]. Only accounting for lower-level constraints would solely guarantee lower-level feasibility, but not optimality in the ulterior planning objective. For instance, even though there might be a feasible scheduling operation that meets planning setpoints and minimizes planning cost, the scheduling level might decide on a different assignment that meets planning requirements and minimizes overall makespan instead. At the same time, the batch scheduling assignments feed into the control layer, whose optimal operation affects the scheduling and planning costs, leading to a tri-level formulation of a 'nested leader-follower' game.

1.3 Contributions and structure

The current chapter presents some PSE background and a recurring case study that motivates our contributions. In the remainder of this chapter, we discuss the structure of this thesis and highlight the contributions of each chapter.

Chapter 2 starts off with a discussion of alternative decision-making frameworks for solving planning, scheduling, and control problems from adjacent fields. We follow up with an explanation of why mathematical programming has crystallized as the predominant framework for solving optimization problems in chemical companies. We continue by giving more background on mathematical programming theory and solvers. We discuss general ML paradigms before moving onto trends in the broader field of ML for optimization. Finally, we give a short perspective on the potentials and pitfalls of applying GenAI advances in operational decision-making that are not considered in the rest of the thesis. We also link the discussed paradigms to the contributions of this thesis. The purpose is to provide the reader with a holistic overview and novel insights from neighbouring fields. Detailed literature reviews are confined to the relevant section in each subsequent chapter.

Chapters 3 to 6 present the core contributions of this thesis spanning work conducted between October 2020 and March 2024. Each chapter is adapted from accepted publications with the

approval of the supervisory team and other co-authors. We give a short description of each work here.

Chapter 3 - A convex quadratic trust-region optimizer for safety-critical derivative-free optimization

Many systems in chemical engineering could be regarded as black-boxes. In optimizing said problems, it might be favourable to use gradient-free optimization instead of relying on the potentially labourious intermediate model-building step. However, a lot of chemical engineering applications pose the additional complexity of unknown safety-critical constraints and noisy evaluations. We developed CUATRO (Convex qUAdratic Trust-Region Optimizer) with these considerations in mind. We benchmark CUATRO against first-order methods based on finite difference approximation and state-of-the-art Derivative-Free Optimization (DFO) solvers on representative chemical engineering case studies: self-optimizing reactors, model-based design of experiments, controller design, and flowsheet optimization.

This research item is published in [385]: **D. van de Berg**, T. Savage, P. Petsagkourakis, D. Zhang, N. Shah, and E. A. del Rio-Chanona, "Data-driven optimization for process systems engineering applications," Chemical Engineering Science, vol. 248, p. 117-135, 2022

Chapter 4 - Data-driven coordination of subproblems under organizational considerations

Inspired by our recurring value chain coordination case study from Section 1.2, we investigate if we can use black-box optimization to coordinate subproblems in EWO under organizational considerations. We use DFO solvers to iteratively find the coordinated variables in collaborative model training, facility location, and multi-objective blending, that - when fixed - optimize a shared objective while preserving the autonomy and privacy of the agents that make up the subproblems. We demonstrate how, with the right exploitation-exploration trade-off, DFO solvers consistently recover the same optimum as the centralized formulation, and benchmark our methodology against the ADMM distributed optimization method .

Early results are published in conference proceedings [384], and the full research item in [387]: **D. van de Berg**, P. Petsagkourakis, N. Shah, and E. A. del Rio-Chanona, "Data-driven coordination of subproblems in enterprise-wide optimization under organizational considerations," AIChE

Journal, vol. 69, no. 4, e17977, 2023

Chapter 5 - Data-driven hierarchical planning-scheduling-control

Drawing from our hierarchical planning-scheduling-control example in Section 1.2, we argue how data-driven techniques become essential to tractably solving integrated problems under game-theoretical considerations. We demonstrate a step-by-step workflow to using optimality surrogates or DFO in finding a tractable solution to a tri-level formulation of our recurring case study. We discuss the tractability and accuracy trade-offs of both methods, along with scaling and safeguarding considerations. We suggest how we can combine both approaches to find solutions to previously intractable multi-level formulations that outperform the heuristic sequential solution approach.

Early results are published in conference proceedings [386], and the full research item in [389]: **D. van de Berg**, N. Shah, and E. A. del Rio-Chanona, "Hierarchical planning-scheduling-control — optimality surrogates and derivative-free optimization," Computers and Chemical Engineering, vol. 188, 108726, 2024

Side note: We could also consider some sort of optimality surrogate approach in the value chain coordination case study (1.2): the surrogates would predict optimal coordinated material flows as a function of planning-level agent observations instead of lower-level optimal responses as a function of higher-level setpoints.

The following co-authored research item explores the use of Multi-Agent Reinforcement Learning in distributed inventory control problems under different game-theoretical considerations, but is not included in later chapters [269]: M. Mousa, **D. van de Berg**, N. Kotecha, E. A. del Rio Chanona, and M. Mowbray, "An analysis of multi-agent reinforcement learning for decentralized inventory control systems," Computers and Chemical Engineering, vol. 188, 108783, 2024

Chapter 6 - Improving high-dimensional derivative-free optimization to enable coordination of hierarchically and geographically distributed systems

In this chapter, we come full circle and revisit the use of CUATRO in high-dimensional black-box optimization applications from the previous two chapters. We discuss improvements that have been made to CUATRO as a solver since its first publication. We specifically highlight an extension CUATRO_PLS that uses Partial Least Squares (PLS) at each iteration to limit computational

complexity by finding latent linear embeddings within the trust region over which to construct the convex quadratic surrogates. We demonstrate that - at low effective dimensionality - this approach generally matches the convergence of state-of-the-art high-dimensional Bayesian optimization and model-based DFO solvers at significant computational speed-up.

This work is an extension of the following conference proceedings [388]: **D. van de Berg**, N. Shah, and A. del Rio-Chanona, "High-dimensional derivative-free optimization via trust region surrogates in linear subspaces," in 34th European Symposium on Computer Aided Process Engineering / 15th International Symposium on Process Systems Engineering, ser. Computer Aided Chemical Engineering, vol. 53, 2024, p. 3193–3198

In the last chapter, we finally conclude with directions for future work.

1.4 Notation

We strive to maintain consistent notation throughout this thesis and carefully define new quantities as they are introduced. Scalar quantities are denoted using standard lowercase letters, vectors with lowercase boldface letters, and matrices with uppercase standard letters — for example, r, $\mathbf{q}$, and P respectively in the definition of multi-dimensional quadratic surrogates ($m(\mathbf{x}) = \mathbf{x}^\intercal P\mathbf{x} + \mathbf{q}^\intercal\mathbf{x} + r$). Lowercase boldface vectors, such as the decision variables $\mathbf{x} \in \mathbb{R}^{n_X}$, represent column vectors unless specified otherwise. Blackboard boldface uppercase letters are special sets, such as the sets of all real numbers $\mathbb{R}$ or natural numbers $\mathbb{N}$. Non-matrix uppercase quantities are denoted with calligraphic letters, including sets ($\mathcal{Z}$), decision-making processes ($\mathcal{D}(\cdot)$, $\mathcal{SC}(\cdot)$), simulations ($\mathcal{BB}(\cdot)$), models ($\mathcal{NN}(\cdot)$), or other transformations ($\mathcal{E}(\cdot)$). This approach helps distinguish, for example, the feasible set $\mathcal{X}$ from the input or feature matrix X containing n_D samples $\mathbf{x}$, or the vector of black-box evaluations $\mathbf{y}$ from its n_D scalar constituent elements y_j. Similarly, the function $\mu_f(\cdot)$ represents the posterior mean prediction of the function $f(\cdot)$ in Gaussian process regression, while the vector μ represents the dual variables in constructing the Lagrangian of an optimization formulation.

We follow specific indexing conventions: subscripts a are used to index agents and subproblems, i for variable dimensions, j for dataset samples, k as a superscript for iterations and as a subscript for discrete time instances, and l for constraints. In iterating over indices, we use $i = 1..n$ as shorthand for $i = 1, 2, \ldots, n$. We use apostrophes when we have to simultaneously index two similar types of sets ($\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_{i'}}$). The superscript $*$ denotes optimal quantities, and the subscript p indicates

private agent-level variables. The dimensionality or number of instances of B is represented as n_B.

While some notation may be overloaded across chapters or sections, we re-define variables within each context to ensure clarity. For instance, $\mathbf{x}$ denotes decision variables in our methodology but may also represent state variables in the case studies to align with conventions in control literature. Similarly, we use $\mathbf{z} \in \mathcal{Z}$ to denote shared or complicating variables in decomposition or coordination contexts, or lower-dimensional latent embeddings in subspace optimization methodologies.

Additional standard notation includes $||\mathbf{x}||_p$ for the L_p norm of $\mathbf{x}$, $\text{proj}_{\mathcal{X}}(\mathbf{x})$ for the projection of $\mathbf{x}$ onto set $\mathcal{X}$, and D_f for the domain of function $f(\cdot)$. $\mathbb{1}$ refers to the identity vector. $f : \mathbb{R}^n \to \mathbb{R}$ indicates that function $f(\cdot)$ maps from $\mathbb{R}^n$ into $\mathbb{R}$. The superscript $\intercal$ denotes the transpose, and $\times$ the cross product. While this list is not exhaustive, new notation is introduced and defined as needed.

We provide a list of acronyms that appear across thesis chapters, unless they are used to an extent where their hyperlinking hinders legibility (DFO). We omit section-specific acronyms, most algorithm names, and case study abbreviations from this list.

2

Background

Contents

2.1 Decision-making paradigms 14
2.1.1 The Beer distribution game: an example from supply chain management 14
2.1.2 The Knapsack problem: an example from operations research 15
2.1.3 The linear-quadratic-Gaussian control problem 17
2.1.4 Prevalence of mathematical programming in PSE 19
2.2 Mathematical programming 21
2.2.1 Duality 21
2.2.2 Decomposition 24
2.2.3 Mathematical programming solvers 27
2.3 Machine Learning (ML) 35
2.3.1 Paradigms 35
2.3.2 Recent wave of ML advances 36
2.3.3 ML for optimization 36
2.3.4 How Machine Learning and optimization are used in this thesis 43

Decision-making in the process industries spans fields as diverse as Supply Chain Management (Supply Chain Management (SCM)), Operations Research (OR), and control. First, we discuss various decision-making paradigms from adjacent fields with the help of concrete examples. Then, we reason why mathematical programming has emerged as the dominant framework in light of the idiosyncrasies of the industry. We continue with a brief overview of mathematical optimization theory and how mathematical programming solvers work. Finally, we finish off the chapter by introducing various Machine Learning (ML) paradigms and the field of ML for optimization, with some perspectives on future trends.

2.1 Decision-making paradigms

2

The act of decision-making can be reduced to a mapping of observable context information $\mathbf{s}$ to actionable decision variables $\mathbf{x}$:

$$\mathbf{x} \quad = \mathcal{D}(\mathbf{s}), \tag{2.1}$$

where $\mathcal{D}(\cdot)$ may be referred to as a 'decision rule' or 'policy'. Decision-makers may introduce additional parameters θ into the policy $\mathcal{D}_\theta$ via model parameters for example. In Supply Chain Management (SCM), decision rules $\mathcal{D}$ could involve the solution of mathematical programs, or simply represent a re-ordering policy $\mathcal{D}_\theta$ for some amount of material $\mathbf{x}$ if the inventory levels $\mathbf{s}$ fall below a certain safe inventory threshold θ. Let us examine different explicit and implicit forms that $\mathcal{D}(\cdot)$ can take based on examples from classical literature in SCM, OR, and control.

2.1.1 The Beer distribution game: an example from supply chain management

The Beer distribution game [201, 254] is an educational simulation-based exercise that provides insights into the dynamics of supply chain management in the face of imperfect information about customer demand, inventory levels, and lead times. The game, as depicted in Figure 2.1, typically involves four players representing different stages of a supply chain: retailer, wholesaler, distributor, and factory. Each player makes ordering decisions based on customer demand and inventory levels with the goal to fulfill customer orders at minimal inventory costs. The Beer Game simulates the bullwhip effect [372], where small changes in customer demand at the retail level can lead to amplified fluctuations in inventory levels and ordering decisions as they propagate upstream through the supply chain. The degree of information sharing and coordination determines which policies can be used. Decision rules $\mathcal{D}$ could take the form of re-ordering policies, mathematical programming, game-theoretical reasoning, and simulations.

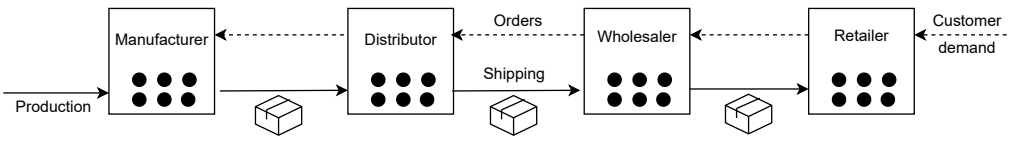


Figure 2.1: Beer distribution game schematic

Reordering policies are parameterized decision rules $\mathcal{D}_\theta(\cdot)$, where a fixed amount of material is ordered at regular intervals (periodic or continuous review policies) or when inventory falls below a threshold (base-stock policies). Parameters θ such as amount of material, interval length, or safe inventory can be determined using simulation optimization tools like genetic algorithms [361], or more recently even Q-Learning (RL) [286]. More broadly speaking, simulation-based methods [413] like Monte-Carlo and discrete-event simulation predict the behavior of the supply chain under different stochastic realizations of inventory policies, demand scenarios, and lead time. Consequently, they allow decision-makers to evaluate the performance of inventory strategies, identify bottlenecks, and assess the impact of uncertainties on supply chain operations.

Game-theoretical approaches analyze strategic interactions between supply chain participants to optimize decision-making strategies and achieve equilibrium outcomes under cooperative and non-cooperative scenarios [347].

Mathematical programming can be used to efficiently solve the inventory control problem and eliminate the bullwhip effect if information is centrally available. If there is perfect coordination among supply chain stages in the Beer distribution game, it essentially converges to a centralized inventory planning problem [189], which can be solved using linear programming. Alternatively, the decisions of each player in the game could be modelled as an inventory control problem under uncertainty, and solved, for example, using stochastic programming [114].

2.1.2 The Knapsack problem: an example from operations research

We consider the Knapsack Problem (KP) as a canonical Combinatorial Optimization (CO) problem from the OR literature [252]. Given a set of n_x items, where each item i has value v_i and weight w_i, the goal is to pick a subset of items that maximize the total value under the condition that the total weight remains below capacity c:

$$\begin{aligned} \max_{\mathbf{x}\in\{0,1\}^N} \quad & \sum_{i=1}^{n_x} x_i v_i \\ s.t. \quad & \sum_{i=1}^{n_x} x_i w_i \leq c. \end{aligned} \tag{2.2}$$

Decision rules $\mathcal{D}(\cdot)$ would accept $\mathbf{v}$, $\mathbf{w}$, and c as context $\mathbf{s}$ and return the binary decision variables $\mathbf{x}$. Asset allocation and ad placement could be formulated as KPs [205]. Exhaustive search

quickly becomes intractable given the combinatorial explosion in the decision variables. Algorithmic frameworks for CO (i.e. decision rules $\mathcal{D}(\cdot)$) can be broadly categorized into exact and heuristic methods. With the exception of constraint programming, exact methods like Dynamic Programming (DP) rely on divide-and-conquer techniques to ensure proven optimality, given sufficient time. They systematically explore the solution space by breaking down complex problems into simpler subproblems.

Dynamic programming exemplifies this approach by decomposing the problem, solving each subproblem only once, and storing the solutions in a table (memoization) [253]. When applied to the KP, each cell in the table represents the maximum value that can be achieved with a given capacity of the knapsack and a subset of items. By considering each item and each possible knapsack capacity, the DP algorithm fills in the table iteratively, considering whether to include each item in the knapsack based on its weight and value.

Mathematical programming techniques typically rely on branch-and-bound routines to systematically explore the solution space by dividing it into smaller subspaces (branches) that are easier to solve via Linear Programming (LP) relaxations. Alternatively, branch-and-cut methods bound the search by using 'cuts' (cutting planes) to eliminate suboptimal solutions. In the KP, the assignment variables would constitute the branching variables, and the objective would be used to prune suboptimal branches [252, 251]. The efficiency of mathematical programming approaches largely depends on the branching strategy, tightness of bounding technique used, and how quickly a good, feasible solution is found.

Constraint programming focuses on defining variables, domains, and constraints, and relies on constraint propagation and search techniques to find feasible solutions [95]. The propagation step is applied to remove the variable assignments from the domain that violate a certain constraint, and is repeated for each domain change and constraint. The efficiency of constraint programming largely depends on the quality of the constraint propagators.

The above techniques are especially powerful when used in a hybrid fashion. For instance, constraint programming techniques often perform much of the heavy lifting in commercial integer linear programming solvers as part of the 'pre-solve' routine [3]. The goal of the pre-solve is to identify implied or redundant constraints, fix variables, or otherwise reduce the variable domain.

While the aforementioned techniques present flexible frameworks with the promise of optimality guarantees, they display exponential time complexity in the worst case. Consequently, exact methods become intractable for large problem sizes ($\gg 1,000$ items and large capacities). Heuristics and metaheuristics become promising alternatives when good solutions need to be found in reasonable time without necessarily focusing on optimality guarantee.

Heuristics rely on domain-specific knowledge or intuition to guide the search process towards promising regions of the solution space. A greedy approximation algorithm for the KP for example involves sorting all items by their density (value/weight ratio), and constructing the solution by sequentially picking the most promising items until running out of space [66]. Approximation algorithms can still allow provable guarantees on the distance of the returned solution to the optimum [390]. The greedy heuristic solution for the KP for example is identical to the LP relaxation without the last fractional item.

Metaheuristics are higher-level problem-solving frameworks that can rely on heuristics as building blocks. Genetic algorithms, simulated annealing, or tabu search for example [320] provide a systematic way to escape local optima and explore diverse regions of the solution space.

Local search is an approximate solution approach that iteratively constructs a neighbourhood via permutations of the current solution [238]. This neighbourhood is often constructed by using problem-specific heuristics (k-opt in the traveling salesman problem for example [193]). A candidate selection routine (potentially based on metaheuristics) is then used to choose the next solution among the neighbourhood candidates.

Hybrid approaches can again be particularly powerful in solving real-world CO problems. For instance, in large neighbourhood search [6], a subset of the total domain of variables is picked at each iteration. Mathematical programming, or local search, can then be used to optimize this 'large neighbourhood'. While randomly picking the neighbourhood is theoretically sound, domain expertise and heuristics can be used to improve this choice.

2.1.3 The linear-quadratic-Gaussian control problem

The Linear-Quadratic-Gaussian (LQG) problem is a canonical optimal control formulation that considers the minimization of a quadratic cost function in the state $\mathbf{x}(t)$ and control $\mathbf{u}(t)$ variables.

These variables are subject to linear, time-varying dynamics with Gaussian additive $\mathbf{v}(t)$ and measurement $\mathbf{w}(t)$ noise [239]:

$$\begin{aligned} \min. \quad & \mathbb{E}[\mathbf{x}^\intercal(T)F\mathbf{x}(T) + \int_0^T \mathbf{x}^\intercal(t)Q\mathbf{x}(t) + \mathbf{u}^\intercal(t)R\mathbf{u}(t)dt] \\ \text{s.t.} \quad & \dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t) + B(t)u(t) + \mathbf{v}(t), \\ & \mathbf{y}(t) = C(t)\mathbf{x}(t) + \mathbf{w}(t), \\ & F \succcurlyeq 0, \quad Q \succcurlyeq 0, \quad R \succcurlyeq 0. \end{aligned} \tag{2.3}$$

Decision rules $\mathcal{D}$ can again involve the solution of mathematical programming models, or exploit the mathematical structure of the problem to derive explicit control laws. In this case, matrices $A(t), B(t), C(t), F, Q$, and R would correspond to the context information $\mathbf{s}$, and the control variables $\mathbf{u}(t)$ are the decision variables returned by the decision rule $\mathcal{D}(\cdot)$ at any time t. We use $\mathbf{x}(t)$ for the state variables to align with control literature notation standards. We could distinguish again between exact and approximate schemes, as well as explicit control laws and implicit decision rules. We begin with Model Predictive Control (MPC), often considered the gold standard for nonlinear industrial process control.

Model predictive control is a receding-horizon control strategy. At each time step, MPC solves a finite-horizon optimal control problem to determine the optimal control inputs over a prediction horizon. The first control input from the optimal trajectory is applied to the system, and the process is repeated in the next time step [209]. Generally, numerical methods for optimal control problems fall into the category of direct or indirect methods [311].

Direct methods approximate the control (and sometimes also the state) trajectories over time intervals, transforming the optimal control problem into a finite-dimensional optimization problem. This formulation is then readily handled by Nonlinear Programming (NLP) solvers such as Sequential Quadratic Programming (SQP) [43] or IPOPT [394]. Indirect methods formulate the optimal control problem as a two-point boundary value problem by deriving necessary conditions for optimality from Pontryagin's Minimum Principle [200] or the Euler-Lagrange equations. These methods typically involve solving a system of differential equations, known as the Hamiltonian system, rather than nonlinear programming.

Dynamic programming can again be used by applying Bellman's equation on a discretization of the LQG problem, representing the principle of optimality on the optimal 'cost-to-go' $J(\mathbf{x}_k)$

function at time step k. The objective in (2.3) could be rewritten recursively as:

$$J(\mathbf{x}_m) = \min_{\mathbf{u}_k,\ k=m..n_T-1} L_f(\mathbf{x}_{n_T}) + \sum_{k=m}^{n_T-1} L(\mathbf{x}_k, \mathbf{u}_k) = \min_{\mathbf{u}_m}. \ L(\mathbf{x}_m, \mathbf{u}_m) + J(\mathbf{x}_{m+1}).$$

The problem would typically be solved backward in time, starting from the final condition. DP can become computationally intractable for problems with a large number of state variables or a long time horizon. In the case of the linear-quadratic regulator problem (LQG but assuming perfect state information), we can derive a quadratic optimal cost [341]: $J(\mathbf{x}_m) = \frac{1}{2}\mathbf{x}_m^\intercal P_m \mathbf{x}_m$, where P_m can be obtained via Riccati recursion. This leaves us with a discrete-time feedback law with linear gain matrix K_m: $\mathbf{u}_m = -K_m \mathbf{x}_m$.

Explicit control laws can also be derived for continuous-time dynamics by combining linear-quadratic state estimators (Kalman filters) [199] and linear-quadratic regulators. This enables an efficient computation of the optimal linear state-feedback gain matrix $K(t)$ in $\mathbf{u}(t) = -K(t)\mathbf{x}(t)$ by solving two associated associated Riccati differential equations [19], while remaining robust to noise and disturbances.

Approximate dynamic programming attempts to break the curse of dimensionality of DP by approximating the state-value function $\hat{J}(\cdot)$, which can then be trained via expert or policy iteration [292]. RL techniques like Q-Learning would present an alternative framework for value function approximation [56]. Other methods like policy-based RL would train a model with the aim of returning optimal controls as a function of the states by using gradient-based methods to optimize rewards directly [160]. ADP is not necessarily advisable for the LQG problem given the existence of exact control laws.

2.1.4 Prevalence of mathematical programming in PSE

Mathematical programming stands out as the most popular decision-making paradigm in the PSE literature [153], with contributions ranging from specialized solvers [394, 368], to Mixed-Integer Programming (MIP) frameworks like generalized disjunctive programming [155], and specific model formulations such as the state-task-network [339]. Planning, scheduling, and control have unique requirements that call for mathematical programming.

Supply chains are often vertically and horizontally integrated. In the absence of information sharing constraints, strategic planning is simply reduced to inventory control. Supply chain dynamics can be accurately modelled using LP or Integer Programming (IP) formulations, and large instances solved efficiently by commercial solvers [130, 157]. As a result, the primary challenges in supply chain management are not due to complex system descriptions or optimization formulations but rather stem from uncertainties in parameters and forecasts.

Scheduling problems typically involve continuous- or discrete-time state-task- or resource-task-network formulations [75]. Mathematical programming effectively handles both continuous and discrete decisions, whereas constraint programming and local search primarily address pure CO problems. MIP solvers manage a wide range of constraints in large-scale real-world problems with mathematical rigor.

Industrial process control [310] differs from other optimal control literature due to the slow dynamics of chemical processes, which hinder the use of linearization techniques. The complexity here lies in solving detailed, nonlinear control problems in short time horizons.

Problem-solving in OR and computer science follows similar workflows [403, 290, 97]: problems are formulated to closely match canonical mathematical problems, and algorithms are selected based on complexity and problem properties. For instance, the maximum-flow problem can determine if plants with production constraints can meet vendor demands via routes with given capacities [7]. The integral flow theorem ensures linear programming finds the optimal solution, i.e. when demand and capacities are integral, the optimal flows are as well. Alternatively, any number of algorithms can be used: based on the relative number of edges and nodes, one would either prefer the Edmonds-Karp [122] or Dinic's [115] algorithm, scaling with $\mathcal{O}(VE^2)$ or $\mathcal{O}(V^2E)$ respectively. The choice of data structure matters too: Dinic's algorithm with dynamic trees reduces complexity to $\mathcal{O}(VE \log V)$ [351].

Real-world chemical engineering problems often do not fit neatly into standard problem classes, particularly when integrating planning, scheduling, and control. Advances in mathematical programming frameworks and solvers offer flexibility and efficiency. However, model heterogeneity and increasing complexity at smaller time horizons limit the scale of problems that yield provably optimal solutions.

2.2 Mathematical programming

In this section, we present background information on optimization theory and solvers. We set the scene for methodologies in later chapters, where we employ decomposition and distributed optimization algorithms, and are confronted with high-dimensional optimization problems. Before starting with the exposition, the reader is reminded of the concept of convexity. A set $\mathcal{C}$ is said to be convex if for any two points $\mathbf{x}$ and $\mathbf{x}'$ belonging to $\mathcal{C}$, any point on the line segment connecting them also belongs to $\mathcal{C}$:

$$\forall \mathbf{x}, \mathbf{x}' \in \mathcal{C}, \forall t \in [0,1] : t\mathbf{x} + (1-t)\mathbf{x}' \in \mathcal{C}.$$

A function $f : \mathbb{R}^{n_x} \to \mathbb{R}$ is said to be convex if for any two points $\mathbf{x}$ and $\mathbf{x}'$ in its domain D_f, the line segment connecting $(\mathbf{x}, f(\mathbf{x}))$ and $(\mathbf{x}', f(\mathbf{x}'))$ on the graph of the function lies above the graph itself:

$$\forall \mathbf{x}, \mathbf{x}' \in D_f, \forall t \in [0,1] : f(t\mathbf{x} + (1-t)\mathbf{x}') \leq tf(\mathbf{x}) + (1-t)f(\mathbf{x}').$$

2.2.1 Duality

We start off with introducing duality, which is instrumental to understanding decomposition and distributed optimization methods. While many of the subsequent concepts can be generalized, we first restrict our discussion to a generic convex optimization problem for the purpose of illustration:

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^{n_x}} . \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_l(\mathbf{x}) \leq 0, \quad l = 1..n_g \\ & h_{l'}(\mathbf{x}) = 0, \quad l' = 1..n_h, \end{aligned} \tag{2.4}$$

where $\mathbf{x} \in \mathbb{R}^{n_x}$ is a vector of continuous variables, each inequality constraint $g_l(\mathbf{x})$ is convex, and each equality constraint $h_{l'}(\mathbf{x})$ is affine. The feasible domain $\mathcal{X}$ is defined as follows: $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^{n_x} \quad | \quad \mathbf{g}(\mathbf{x}) \leq \mathbf{0}, \quad \mathbf{h}(\mathbf{x}) = \mathbf{0}\}$ where $\mathbf{g}(\cdot)$ and $\mathbf{h}(\cdot)$ denote all n_g and n_h inequalities and equalities respectively. A solution $\mathbf{x}^*$ is said to be optimal if $\forall \mathbf{x} \in \mathcal{X} : f(\mathbf{x}^*) \leq f(\mathbf{x})$. For a strongly convex problem, there is only one such optimum. The basic idea in Lagrange duality is to relax the original problem (2.4) by constructing the Lagrangian (2.5):

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

Lagrangian relaxation involves removing the constraints and adding them to the objective with the help of Lagrange multipliers $\lambda \in \mathbb{R}^{n_h}$ and $\mu \in \mathbb{R}^{n_g}_+$ [289]. These dual variables are used throughout dual decomposition and distributed optimization techniques. Lagrange duality theory links the original minimization problem (2.4), termed the primal problem, with a dual maximization problem [289]. The Lagrange dual function is defined as:

$$q(\mu, \lambda) = \inf_{\mathbf{x}} \quad \mathcal{L}(\mathbf{x}, \mu, \lambda). \tag{2.6}$$

$q(\mu, \lambda)$ is concave because it is affine in μ and λ. Since the infimum in (2.6) is taken over all $\mathbf{x}$, feasible and infeasible, the following property holds between the primal and dual objectives: $f(\mathbf{x}) \geq q(\mu, \lambda), \quad \forall \mathbf{x} \in \mathcal{X}$. In other words, the solution to the maximization of (2.6) provides a valid lower bound to the optimal solution $f^* = f(\mathbf{x}^*)$ of the original problem in (2.5). The difference between the optimal primal objective and the optimal dual objective is called the duality gap, which is always nonnegative $f^* - q^* \geq 0$ (weak duality). When the problem is convex, under some mild technical conditions (Slater's conditions), the duality gap reduces to zero at the optimum, i.e. strong duality holds. In these instances, the primal problem (2.4) can be equivalently solved by solving:

$$q^* = \max_{\mu, \lambda}. \quad q(\mu, \lambda) \quad s.t. \quad \mu_l \geq 0, \quad l = 1..n_g. \tag{2.7}$$

As a side note, since this problem is convex, the optimal $\mathbf{x}^*, \lambda^*,$ and μ^* could also be computed by solving the system of Karush-Kuhn Tucker (KKT) conditions, satisfying primal and dual optimality:

$$\begin{aligned}
&\nabla_{\mathbf{x}^*} \mathcal{L}(\mathbf{x}^*, \lambda^*, \mu^*) = 0 && \text{Gradient of } \mathcal{L} \text{ with respect to } \mathbf{x}^*, \\
&h_{l'}(\mathbf{x}^*) = 0, \quad l' = 1..n_h && \text{Primal feasibility or } \nabla_{\lambda^*_{l'}} \mathcal{L}(\cdot), \\
&g_l(\mathbf{x}^*) \leq 0, \quad l = 1..n_g && \text{Primal feasibility or } \nabla_{\mu^*_l} \mathcal{L}(\cdot), \\
&\mu_l \geq 0, \quad l = 1..n_g && \text{Dual feasibility}, \\
&\mu_l g_l(\mathbf{x}^*) = 0, \quad l = 1..n_g && \text{Complementarity slackness}.
\end{aligned} \tag{2.8}$$

Subgradient method

The generic convex optimization problem (2.4) can be rewritten as:

$$\min_{\mathbf{x} \in \mathcal{X}}. \quad f(\mathbf{x}), \tag{2.9}$$

where $\mathcal{X}$ denotes the convex feasible set. Depending on whether f is differentiable, one can use either the gradient or subgradient with respect to $\mathbf{x}$, both denoted $\mathbf{s}(\mathbf{x})$ for this purpose, to approach the optimum $\mathbf{x}^*$, by iteratively updating $\mathbf{x}$ using:

$$\mathbf{x}^{k+1} = \text{proj}_{\mathcal{X}}(\mathbf{x}^k - \alpha^k \mathbf{s}(\mathbf{x}^k)), \tag{2.10}$$

where $\text{proj}_{\mathcal{X}}(\cdot)$ denotes the projection onto $\mathcal{X}$. (2.10) essentially corresponds to a (sub)gradient descent step projected onto the feasible domain. In theory, as k approaches infinity, $\mathbf{x}^k$ converges towards the optimum $\mathbf{x}^*$ under certain restrictions on (sub)gradients and step size choices α^k.

The (sub)gradient method becomes a useful tool in the framework of dual ascent. For this, the primal and dual problems are iteratively optimized while keeping the dual and primal variables fixed respectively. The primal update can be solved efficiently as part of a convex programming routine, and the subgradient method is used to update the dual variables μ with $\mathbf{s}(\mathbf{x}^{k+1}) = \sum_l^{n_g} \mathbf{s}_l(\mathbf{x}^{k+1})$:

$$\begin{aligned} &1) \quad \mathbf{x}^{k+1} = \arg\min_{\mathbf{x}\in\mathcal{X}} .\mathcal{L}(\mathbf{x}, \mu^k) \\ &2) \quad \mu^{k+1} = \text{proj}_+(\mu^k + \sum_l^{n_g} s_l(\mathbf{x}^{k+1})). \end{aligned} \tag{2.11}$$

Gauss-Seidel and Jacobi updates

We then consider the same optimization formulation, but where $\mathbf{x}$ is split into multiple vectors:

$$\min_{\mathbf{x}} \quad f(\mathbf{x}_1, \ldots, \mathbf{x}_a, \ldots, \mathbf{x}_{n_a}) \quad s.t. \quad \mathbf{x}_a \in \mathcal{X}_a, \quad a = 1..n_a, \tag{2.12}$$

where $\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_a, \ldots, \mathbf{x}_{n_a}]$. Rather than update all of $\mathbf{x}$ at once, we can iteratively optimize one set of variables $\mathbf{x}_a$ while keeping the other vectors constant at their previous iterate or their most recent update:

$$\begin{aligned} \mathbf{x}_a^{k+1} &= \arg\min_{\mathbf{x}_a\in\mathcal{X}_a} .f(\mathbf{x}_1^k, \ldots, \mathbf{x}_{a-1}^k, \mathbf{x}_a, \mathbf{x}_{a+1}^k, \ldots, \mathbf{x}_{n_a}^k) \qquad \text{Jacobi method,} \\ \mathbf{x}_a^{k+1} &= \arg\min_{\mathbf{x}_a\in\mathcal{X}_a} .f(\mathbf{x}_1^{k+1}, \ldots, \mathbf{x}_{a-1}^{k+1}, \mathbf{x}_a, \mathbf{x}_{a+1}^k, \ldots, \mathbf{x}_{n_a}^k) \qquad \text{Gauss-Seidel method.} \end{aligned} \tag{2.13}$$

These iterations converge to the optimum for convex problems. Practically, it might be more efficient to only take smaller but faster gradient descent steps in the general direction of the exact

updates from (2.13) at each iteration k.

2

2.2.2 Decomposition

Decomposition techniques break up a centralized formulation into subproblems, and approach the centralized solution by iterating over the subproblem instances [51]. At each iteration, the subproblems can be solved sequentially, or in parallel. In high dimensions, the sequential solution of the subproblems still leads to computational savings if the problem complexity scales worse than linearly with the number of problem dimensions. Outside of numerical considerations, privacy concerns might also favour the use of decomposition: Decentralized solution methods can often be interpreted as protocols that allow a network of subsystems to coordinate their actions to achieve optimality, with methods differing in the amount of information sharing between subsystems.

Problems for which decomposition works in one step are called (block) separable, or trivially parallelizable. The solution to this convex problem:

$$\min_{\mathbf{x}} \quad \sum_{a=1}^{n_a} f_a(\mathbf{x}_a) \quad s.t. \quad \mathbf{x}_a \in \mathcal{X}_a, \quad a = 1..n_a, \tag{2.14}$$

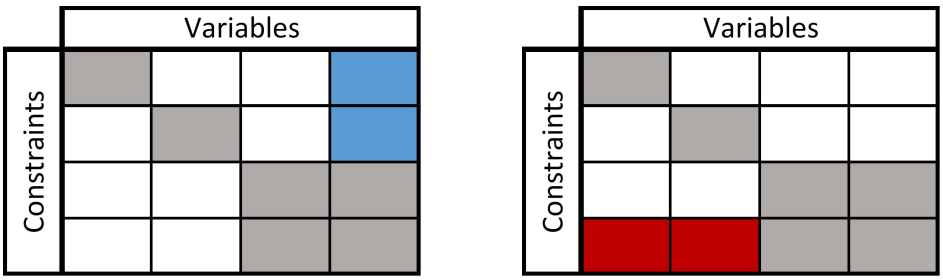
can be recovered by solving the following problems either in parallel or in sequence:

$$\forall a = 1..n_a : \quad \mathbf{x}_a^* = \arg\min_{\mathbf{x}_a} . \quad f_a(\mathbf{x}_a) \quad s.t. \quad \mathbf{x}_a \in \mathcal{X}_a. \tag{2.15}$$

However, problems are rarely trivially separable. More interesting cases arise when otherwise separable subproblems are coupled by few shared, or 'complicating', variables or constraints as illustrated in the variable-constraint matrices in Figure 2.2. These call for slightly different solution approaches, but decomposition always involves the splitting of the original large problem into subproblems that are solved after distribution. The subproblem solutions are then coordinated by a higher-level problem through communication protocols [289].

$$\begin{aligned} \min_{\mathbf{x},z} \quad & \sum_{a=1}^{n_a} f_a(\mathbf{x}_a, z) & \min_{\mathbf{x}} \quad & \sum_{a=1}^{n_a} f(\mathbf{x}_a) \\ \text{s.t.} \quad & \mathbf{g}_a(\mathbf{x}_a, z) \leq \mathbf{0}, \quad a = 1..n_a & \text{s.t.} \quad & \sum_{a=1}^{n_a} \mathbf{g}_a(\mathbf{x}_a) \leq \mathbf{0} \end{aligned} \tag{2.16}$$

Complicating variables are usually associated with primal decomposition; and complicating constraints are usually associated with dual decomposition. The former is based on decomposing



(a) Complicating variables (b) Complicating constraints

Figure 2.2: Constraint-Variable matrix for subproblem interactions

the original primal problem; the latter is based on decomposing the Lagrangian dual problem. Using the economic analogy of resource allocation, primal methods allocate resources directly through the shared variables, while dual methods allocate resources via pricing [289]. Many different variations of these distributed algorithms can be obtained based on a combination of decomposition (primal, dual, hybrid, hierarchical) and update schemes (gradient/subgradient, Gauss–Seidel, and Jacobi) [289].

Primal decomposition

A primal decomposition is appropriate when the problem has shared variables (z in Equation 2.16). After fixing these, the original primal problem becomes trivially separable, and can be decomposed into a higher-level problem and lower-level subproblems. The subproblems are easily solved in parallel or sequentially using any suitable optimization solver of choice. Before addressing the primary problem, we define $f_a^*(\mathbf{z}) = \min_{\mathbf{x}_a \in \mathcal{X}_a} f_a(\mathbf{x}_a, \mathbf{z})$. The original problem from (2.16) becomes:

$$\min_{\mathbf{z} \in \mathcal{Z}} \sum_{a=1}^{n_a} f_a^*(\mathbf{z}). \tag{2.17}$$

Then, the sub- and primary problems are solved iteratively in a 'dual-descent type' algorithm. This could involve the following steps:

$$\begin{aligned}
&1) \quad \forall a = 1..n_a: \quad \mathbf{x}_a^{k+1} = \arg\min_{\mathbf{x}_a \in \mathcal{X}_a} .\mathcal{L}(\mathbf{x}_1^k, \ldots, \mathbf{x}_a, \ldots, \mathbf{x}_{n_a}^k, \mathbf{y}^k), \\
&2) \quad \text{Get subgradients } \mathbf{s}_a^k \text{ from 1) for all subproblems,} \\
&3) \quad \text{Solve (2.17) via subgradient method : } \mathbf{z}^{k+1} = \text{proj}_{\mathcal{Z}}(\mathbf{z}^k - \alpha^k \sum_{a=1}^{n_a} s_a(\mathbf{x}^{k+1})).
\end{aligned} \tag{2.18}$$

However, as is often the case with subgradient methods, convergence is generally slow. On top of

this, apart from specific convex cases, it is non-trivial to get cheap evaluations of the subgradients from the solution of the subproblems. For instance, when the original convex problem only has affine inequalities ($A_a\mathbf{x}_a \leq \mathbf{z}$), which upon solution of the primal have optimal Lagrange multipliers $\mu_a^*(\mathbf{z})$, then the overall subgradient becomes: $\mathbf{s}(\mathbf{z}) = \sum_{a=1}^{n_a} \mathbf{s}_a(\mathbf{z}) = \sum_{a=1}^{n_a} \mu_a^*(\mathbf{z})$. While $\mathbf{x}_a$ can be interpreted as local or private variables, $\mathbf{z}$ are considered shared or global variables.

Dual decomposition

Dual decomposition is appropriate when the problem has complicating constraints ($\sum g_a(\mathbf{x}_a) \leq 0$ in Equation 2.16). After applying Lagrangian relaxation, the original problem can be decomposed into a higher-level (primary) and separable lower-level subproblems. The subproblems are again easily solved using any optimization solver of choice. Before addressing the primary problem, the Lagrangian is constructed: $\mathcal{L}(\mathbf{x}, \mu) = \sum_{a=1}^{n_a} f_a(\mathbf{x}_a) + \mu^\intercal \sum_{a=1}^{n_a} \mathbf{g}_a(\mathbf{x}_a)$. This allows for the formation of the Lagrangian dual as :

$$q(\mu) = \inf_{\mathbf{x}\in\mathcal{X}} \mathcal{L}(\mathbf{x}, \mu) = \sum_{a=1}^{n_a} \inf_{\mathbf{x}_a\in\mathcal{X}_a} (f_a(\mathbf{x}_a) + \mu^\intercal \mathbf{g}_a(\mathbf{x}_a)).$$

The following subproblems are then solved iteratively in a 'dual-ascent type' algorithm:

$$\begin{aligned}
&1) \quad \forall a = 1..n_a : \quad \mathbf{x}_a^{k+1} = \arg\min_{\mathbf{x}_a\in\mathcal{X}_a} .\mathcal{L}_a(\mathbf{x}, \mu^k) = \arg\min_{\mathbf{x}_a\in\mathcal{X}_a} .f_a(\mathbf{x}_a) + \mu^{k\intercal} \mathbf{g}_a(\mathbf{x}_a), \\
&2) \quad \text{Get subgradients } \mathbf{s}_a^{k+1} = \mathbf{g}_a(\mathbf{x}_\mathbf{a}^{\mathbf{k+1}}) \text{ from 1) for all subproblems ,} \\
&3) \quad \text{Subgradient method: } \mu^{k+1} = \underset{\mu\geq\mathbf{0}}{\text{argmax.}} \sum_{a=1}^{n_a} q_a(\mu) = \text{proj}_+(\mu^k + \alpha^k \sum_{a=1}^{n_a} \mathbf{s}_a^{k+1}).
\end{aligned} \tag{2.19}$$

In the case where the problem contains equalities $\mathbf{h}(\cdot)$ rather than inequalities $\mathbf{g}(\cdot)$, λ is used instead of μ where λ is not restricted to the positive domain. The drawbacks of the dual decomposition are similar to those of the primal one, in that they converge slowly and are only applicable to convex problems.

$\mathbf{x}_a$ are considered local or private variables, while λ can be interpreted as global prices. Although coupling variables are more readily handled by primal decomposition, they can be easily transformed into shared constraints. First, local copies are added as additional decision variables for each subproblem, then constraints are added requiring the local copies to be equal. These constraints are then relaxed by penalizing any deviation between these variables in the Lagrangian. This is called 'Lagrangean decomposition', or 'variable splitting' [156].

While all these concepts are introduced for a convex case study, these concepts and techniques can still be useful for nonconvex problems. However, they lose any guarantee of convergence or global optimality [365]. In the above exposition, we mostly focus on theory and often reference the fact that we can efficiently solve convex programming formulations. In what follows, we explain the workings of mathematical programming solvers.

2.2.3 Mathematical programming solvers

Continous optimization

In (2.10), we introduced the (sub)gradient method as a descent step combined with a potential projection onto the feasible domain. Gradient descent presents the simplest instance of gradient-based optimization. At each iteration, (automatic) differentiation is used to extract first- or second-order gradient information with respect to the primal and/or dual variables from the differentiable model expressions. Update steps are then taken in a search direction $\mathbf{d}^k$, obtained from this gradient information, with a step size α^k. This is repeated for a fixed number of iterations or until convergence criteria to a (local) optimizer are met.

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha^k \mathbf{d}^k \tag{2.20}$$

Solvers mostly differ in the way that search directions and the update step sizes α^k are computed, and how feasible regions are treated [283]. Newton's method for example conditions the first-order gradient update based on second-order derivative information. α^k is usually computed by backtracking, meaning that α^k is reduced from an initial guess until the resulting update step leads to a reduction in the objective of a certain quality. Gradient-based update steps feature in subroutines across a wide range of optimization problems, from (non)linear programming to mixed-integer and global optimization.

Nonlinear programming typically relies on state-of-the-art interior point solvers such as sequential quadratic programming (SQP), IPOPT [394], APOPT [169], and CONOPT [17]. Given a feasible starting point, IPOPT performs Newton-like update steps, and penalizes inequalities in the objective using self-concordant barrier methods. Additionally, if the problem is of a special form, namely conic or semidefinite, the barrier function can be tailored to the specific problem type, leading to a range of specialized solvers such as MOSEK [15]. These solvers can then be leveraged

within the framework of convex or more general optimization modeling languages like CVXPY [111] or Pyomo [164] respectively. Other techniques to address convex problems [50] include the bundle [175], ellipsoid [42], and cutting-plane methods [206].

Linear Programming (LP) inequality constraints present feasible regions that are defined by convex polytopes. If a problem has an optimum in the feasible region, it must have this value on at least one of the extreme points (vertices) of the polytope [274]. Consequently, for lower- to moderate-dimensional LP formulations, the simplex algorithm can be used to find the optimum by moving along polytope edges from a starting vertex. However, in large problems, commercial LP solvers usually rely on a combination of the simplex method and gradient-based interior-point solvers to cut through the feasible region and avoid the combinatorial enumeration of polytope vertices.

Nonconvex programming usually comes at the expense of either speed or global optimality. A locally but not necessarily globally optimal solution to a nonconvex problem can be found by repeatedly solving local convex approximations of the original problem at the current iterate - this is called sequential convex programming. The most popular of these methods is SQP [43], where a problem of the form (2.4) is solved by finding the descent direction $\mathbf{d}$ that, at each iteration k, minimizes the second-order Taylor approximation of the objective given first-order approximations of the constraints:

$$\begin{aligned} \min_{\mathbf{d}\in\mathbb{R}^{n_x}} \quad & f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^\intercal \mathbf{d} + \mathbf{d}^\intercal \nabla^2 f(\mathbf{x}^k)\mathbf{d} \\ \text{s.t.} \quad & g_l(\mathbf{x}^k) + \nabla g_l(\mathbf{x}^k)^\intercal \mathbf{d} \leq 0, \quad l = 1..n_g, \\ & h_{l'}(\mathbf{x}^k) + \nabla h_{l'}(\mathbf{x}^k)^\intercal \mathbf{d} = 0, \quad l' = 1,\dots,n_h. \end{aligned} \tag{2.21}$$

SQP could be viewed as an extension of Newton's method capable of handling constraints. Whereas Newton's method uses the Hessian of the objective function, SQP uses approximations of the Hessian of the Lagrangian. A multi-start approach is often employed in an effort to escape local minima. Variations on the SQP method include using linear objectives in sequential linear programming. Linear or quadratic approximation updates can also be employed within trust region frameworks [283], which can be an effective technique in high-dimensional problems.

Trust regions

In the case of sequential linear or quadratic programming, line search and trust region methods use the objective and constraint approximations in different ways to find the update step. Line search methods, such as backtracking, solve (2.21) to find a search direction $\mathbf{d}^k$, and then focus their efforts on finding a suitable step length α^k in this direction that leads to adequate improvement in the objective and/or constraint satisfaction. Trust-region methods on the other hand define a region around the current iterate within which the approximations are assumed to be accurate. Then, the update step is chosen to be the minimizer of (2.21) after a convex trust region constraint is included. The trust region center and radius are updated based on the actual and expected improvements at each step.

The choice between both methods is mostly determined by the behaviour of the function to be optimized. Trust region methods are preferred over line search methods in numerical optimization when dealing with non-smooth problems, imperfect gradient information, and constrained optimization problems.

Divide-and-conquer

MIP is nonconvex in the complicating discrete variables $\mathbf{z} \in \mathbb{N}^{n_z}$ [237]. A brute force method would enumerate all possible combinations of complicating variables and pick the best solution from the resulting continuous problems. As an alternative, MIP solvers iterate between higher- and lower-level problems. Integrality conditions are relaxed in the main problems with their solution giving a lower bound (LB). The complicating variables are then fixed and fed to the subproblem that determines feasibility of the proposed set of complicating variables, and has the potential to update the upper bound (UB), the best-to-date solution. The program terminates if the relaxed solution is feasible ($LB = UB$), when the lower and upper bounds are within an acceptable distance ϵ of each other ($UB - LB < \epsilon$), or if a time or evaluation budget is reached.

The most popular MIP method is branch-and-bound [223], where at each node, some discrete (complicating) variables are fixed while the remainder are relaxed. The parent node represents a completely continuous problem. Heuristics determine which variables to fix, and which nodes to explore next, but the termination criteria are rigorous and allow for the pruning of leaves whose relaxation is shown to be inferior to a feasible solution.

Alternatively, Benders decomposition is based on a sequence of projections, cuts, and relax-

ations [145, 146]. In the primary problem, the original complicating constraints involving any continuous variables are relaxed. This problem is then optimized with respect to the complicating discrete variables. The complicating variables are then fixed in the subproblems, whose solution in turn determines optimality or feasibility cuts to be added to the primary problem. Cuts are linear inequalities, grounded in duality theory, that restrict the decision space of the primary problem without compromising any feasible, optimal solutions. This process is repeated until termination.

Global optimization operates in a similar primary and subproblem framework. It is combinatorial in nature, like its MIP counterpart. The most popular global optimization algorithm is called spatial branch-and-bound [236]. Rather than relaxing discrete variables to be continuous, nonconvex terms are relaxed using their convex envelope. Branching happens on the continuous and/or discrete variables leading to space partitioning over which the convex envelopes are constructed. Over each region in the space partitioning, one can solve the original formulation to get a potential upper bound and the relaxation to get a lower bound. The solution of Mixed-Integer Nonlinear Programming (MINLP) is significantly more complex than its Mixed-Integer Linear Programming (MILP) counterpart, since a convex MIP must be solved for each relaxation. State-of-the-art global MINLP solvers include BONMIN [45] and BARON [368].

Solving 'huge' optimization problems

Many optimization formulations present mathematical structure that lends itself well to decomposition techniques and distributed optimization as introduced in Section 2.2.2. Here, we highlight some solution techniques that can help in huge-scale settings, where full-dimensional vector operations are computationally expensive, and mathematical structure cannot be exploited.

Sketching techniques[1] reduce dimensionality by projecting the original space randomly onto a lower-dimensional subspace [406, 246, 242]. They are theoretically underpinned by the Johnson-Lindenstrauss lemma [195], which states that points can be embedded into a lower-dimensional space such that distances between the points are approximately preserved. Sketching has been successfully used to improve the efficiency of nonlinear least squares solvers [73].

These techniques are not restricted to model-based optimization and can be extended to DFO

[1]Stochastic gradient descent - the simplest form of first-order optimization crucial for training in deep learning - effectively performs sketching on data instances rather than on the optimization variables. It accelerates the optimization process by considering only a subset of data instances (mini-batches) at each iteration, rather than the entire dataset.

[70], and Bayesian Optimization (BO) [231]. Instead of random linear embeddings, one can also use active learning [144] or manifold learning [267] to find more informative non-linear embeddings for dimensionality reduction.

Block-coordinate-descent could be considered a sketching technique [277]. It extends coordinate search methods by updating a subset of the variables at each iteration, typically along coordinate directions. Block-coordinate-descent methods have been extended to nonlinear least-squares problems, leading to so-called subspace Gauss-Newton methods [64].

(Multi-)Parametric programming

In (multi-)parametric programming [291], we consider the objective and/or constraints to be parameterized by θ:

$$J^*(\theta) = \min_{\mathbf{x}} f(\mathbf{x}, \theta) \quad \text{s.t.} \quad \mathbf{g}(\mathbf{x}, \theta) \leq \mathbf{0}. \tag{2.22}$$

Here $J^*(\theta)$ denotes the optimal objective value as a function of θ. For each fixed θ, the feasible region for $\mathbf{x}$ is $X(\theta) = \{\mathbf{x} \mid \mathbf{g}(\mathbf{x}, \theta) \leq \mathbf{0}\}$. Consequently, both the feasible set and the optimal solution typically vary as the parameters change. Parametric programming seeks to explicitly characterize the optimal solution as a function of the parameters. In other words, the aim is to turn the implicit decision rule $\mathcal{D}(\theta)$ from (2.1) relying on mathematical programming into an explicit form.

Parametric optimization methods fall into several categories. For mpLPs and convex mpQPs, explicit analytical solutions can be derived by enumerating active constraint sets, yielding a piecewise-affine mapping $\mathbf{x}^*(\theta)$ [291]. For nonlinear problems, local linearization or quadratic approximation techniques transform the problem into an mpLP or mpQP, while parametric sensitivity analysis provides a local solution trajectory. Numerical algorithms—such as active-set enumeration, geometric partitioning, and hybrid methods—are commonly used to explore high-dimensional parameter spaces. While powerful for the solution of robust [23] and real-time optimization [151] formulations, parametric programming suffers from combinatorial complexity, high sensitivity and degeneracy in the parameter space.

In mpLPs and mpQPs, the dependence on θ is usually piecewise-affine. In more general nonlinear parametric programming (mpNLPs), the optimal solution map may be non-polyhedral and non-smooth, often exhibiting multiple local optima. Approximation methods such as local linearizations or partial offline computations are commonly used to obtain a tractable, albeit approx-

imate, solution. When discrete decision variables are involved (e.g., mpMILP or mpMINLP), the solution can change discontinuously at certain parameter thresholds. The combinatorial explosion in the number of candidate regions makes explicit enumeration impractical for all but the smallest problems, often requiring tailored heuristic or decomposition-based algorithms. In mp-MPC, the states (and sometimes disturbances) are treated as parameters in an optimization problem — usually an mpQP — to compute an explicit control law. This approach is also known as 'MPC-on-a-chip' [302], where the online controller simply evaluates a pre-computed piecewise-affine function, ensuring rapid and predictable response times.

Bi-level and multi-level optimization

Bi-level optimization involves two optimization problems in a hierarchical (leader-follower) structure, where the solution of the higher level is subject to an optimal soluton of one or multiple lower-level optimization problems [92, 350]. The bi-level problem can be formalized as:

$$\min_{\mathbf{x},\mathbf{y}} \quad F(\mathbf{x},\mathbf{y}) \tag{2.23}$$

$$\text{s.t.} \quad \mathbf{G}(\mathbf{x},\mathbf{y}) \leq \mathbf{0}, \tag{2.24}$$

$$\mathbf{y} \in \arg\min_{\mathbf{z}} \{f(\mathbf{x},\mathbf{z}) \mid \mathbf{g}(\mathbf{x},\mathbf{z}) \leq \mathbf{0}\}, \tag{2.25}$$

where $\mathbf{x}$ and $\mathbf{y}$ denote the upper-level (leader) and lower-level (follower) variables; $F(\mathbf{x},\mathbf{y})$ the leader's and $f(\mathbf{x},\mathbf{y})$ the follower's objectives; $\mathbf{G}(\cdot)$ and $\mathbf{g}(\cdot)$ constraints at the upper and lower levels, respectively. (2.25) enforces the follower's decision $\mathbf{y}$ to be an optimal solution of the lower-level problem for the given $\mathbf{x}$. A major modelling challenge arises when the follower's optimal solution is not unique. In such cases, one must specify how the follower breaks ties among multiple optima, which leads to two alternative bi-level problem definitions. The optimistic (or cooperative) formulation assumes the follower chooses the optimal response that is most favorable for the leader. Mathematically, letting $S(\mathbf{x})$ denote the set of optimal solutions of the lower-level problem for a given $\mathbf{x}$, the optimistic bi-level problem can be expressed as:

$$\min_{\mathbf{x}\in X,\ \mathbf{y}\in S(\mathbf{x})} F(\mathbf{x},\mathbf{y}), \tag{2.26}$$

meaning the leader optimizes $F(\mathbf{x},\mathbf{y})$ under the assumption that the follower selects $\mathbf{y} \in S(\mathbf{x})$ that minimizes F (among the follower's optima). In contrast, the pessimistic (or adversarial)

formulation assumes the follower might behave in a way that is worst for the leader, formalized as:

$$\min_{\mathbf{x}\in X} \max_{\mathbf{y}\in S(x)} F(\mathbf{x},\mathbf{y}). \tag{2.27}$$

The optimistic case is more common in the literature as it assumes some cooperation or coordination, whereas the pessimistic case is more conservative but might require additional constraints, making it computationally more cumbersome. Bi-level problems are inherently hard to solve. Even the simplest case – a bi-level problem where both levels are linear programs – is NP-hard to solve to global optimality. Even if each level alone is convex, the feasible set defined by the follower's optimal reaction $S(x)$ can be disconnected. In general, solution methods either reformulate the problem as a single-level optimization, or apply decomposition, relaxation, or heuristic techniques.

Single-level reformulation is applicable when the lower-level problem is convex and satisfies regularity conditions. The follower's optimality is enforced by embedding the lower-level KKT equations as constraints into the upper-level problem. This results in a highly nonconvex single-level mathematical programming formulation with equilibrium constraints (MPEC), which can in principle be solved with standard nonlinear or mixed-integer solvers, but usually remains intractable.

Single-level reformulation could be considered parametric programming, where the decision variable $\mathbf{y}$ is obtained from an explicit mapping $\hat{S}(\mathbf{x})$, parameterized by the complicating upper-level variables $\mathbf{x}$. Avraamidou and Pistikopoulos [21] describe the parametric programming approach to the solution of more general multi-level, mixed-integer optimization problems.

Decomposition techniques rely on the coordination between iterative solutions of the upper- and lower-level problems. For instance, similar to Benders decomposition, when the upper and lower levels are linear, but are coupled via integer decisions, the master problem handles the integer decisions, and the solution of the corresponding subproblems at fixed $\mathbf{x}$ is used to generate cuts (constraints) for the master problem, excluding leader decisions that lead to suboptimal or infeasible follower responses.

When the bilevel problem is non-convex (for example, the follower's problem or the leader's objective is non-convex), the difficulty escalates. The leader's effective objective $F(x, y(x))$ can be highly multi-modal. Global optimization techniques may be required, such as global branch-and-bound or partitioning methods that systematically search the space.

(Meta)heuristic and Evolutionary Approaches sacrifice theoretical guarantees to find satisfactory solutions when exact methods are impractical. These methods are particularly popular for nonconvex formulations. A popular strategy is to use a heuristic optimization routine to propose candidate leader decision, and solve the corresponding lower-level problems using mathematical programming. This ensures feasibility (lower-level optimality) of evaluated solutions, but can still remain intractable to find *good* solutions given the expense of solving lower-level formulations.

Absence of second-order information

Optimization solvers typically rely on automatic differentiation and second-order gradient information. There are however many instances where this information is not available because gradient evaluations would be too expensive or because access is hindered by privacy or proprietary constraints.

First-order optimization has been instrumental to the developments in the field of deep learning. Training neural networks relies on 'backpropagation', essentially a composite chain rule to extract gradient information from a training loss function with respect to the model parameters. State-of-the-art techniques rely on using past evaluations and gradients for adaptive moment estimation (Adam) [212], or Hessian estimation in quasi-Newton methods (BFGS) [240].

Zeroth-order optimization methods purely rely on function evaluations [228]. Similar problem statements are known across different fields under various names: Black-Box Optimization (BBO), DFO, or simulation optimization. DFO solvers operate similarly to local search: candidate solutions (neighbourhoods) are iteratively proposed, evaluated, and updated in an attempt to approach the optimum. Methods differ mostly in how the candidates are suggested. DFO constitutes a key pillar of what we consider data-driven optimization in this thesis, and will be introduced at length in Chapter 3.

DFO solvers could purely consider input-output evaluations like the aforementioned coordinate descent, or the grid- and mesh-based methods DIRECT [197] and NOMAD [229]. Similarly, stochastic methods, like random or evolutionary search, and simulated annealing could fall under this remit. Alternatively, model-based DFO methods rely on the intermediate construction and optimization of surrogates for their update steps, from linear-quadratic trust region methods to BO.

After this whirlwind tour of optimization theory and solvers - the scope of which is admittedly biased towards the needs of this thesis - we present a high level overview of various ML paradigms and how they can be used in conjunction with mathematical programming.

2.3 Machine Learning (ML)

First, we present the major paradigms in ML. Then, we investigate various ways that ML can be used to aid optimization, and give some perspectives on the impact of the new wave of GenAI advances. For an in-depth review of the application of Machine Learning to PSE and adjacent fields, the interested reviewer is referred to [103, 335, 271, 344, 106].

2.3.1 Paradigms

Supervised learning involves training a model on labeled data with the goal to learn a mapping from input features to output labels. Common model families include decision trees, random forests, support vector machines, neural networks, and GP [166, 273].

Unsupervised learning has the objective of training an algorithm to uncover patterns or structures in the data without access to 'ground truth' labels. Unsupervised learning is useful for clustering, dimensionality reduction, and density estimation. Common algorithms include k-means clustering, t-sne, hierarchical clustering, principal component analysis, and autoencoders [167].

Reinforcement learning trains agents directly from reward signals obtained by interaction with a (potentially simulated) environment. The agent learns to take actions that maximize cumulative rewards over time through trial and error. Reinforcement learning algorithms could be generally classified into value function (Q-Learning), policy optimization (Vanilla Policy Gradient), or actor-critic methods. State-of-the-art algorithms like PPO, A2C, and SAC usually involve a mix of policy optimization and Q-Learning [364, 2].

Active learning is a framework in which the learning algorithm can interactively query an oracle (user, expert, teacher) to label selected new data points [336]. The process of Active Learning (AL) is also referred to as optimal experiment design [79]. The goal is to learn as much information about an underlying function or distribution in as few samples as possible. Roughly speaking, AL

reduces to BO [312, 143] when the goal is to find information about the function optimum rather than the whole distribution [110]. Consequently, BO and RL could be viewed as special cases of AL.

2.3.2 Recent wave of ML advances

Generative models are a class of models that learn the underlying probability distribution of the data and can generate new samples from said distribution. Examples of generative models include variational autoencoders (VAEs), generative adversarial networks (GANs), and autoregressive models like the transformer architecture [30, 203].

Foundation models are large-scale, pre-trained language models that serve as the basis for a wide range of downstream tasks [287]. These models, such as Generative Pre-trained Transformer (GPT) [58], are trained on massive text corpora and can be fine-tuned for specific tasks with relatively little additional training data. Language models learn the statistical structure of natural language text.

Foundation models often represent core components of multi-modal models, which are designed to process and integrate information from multiple modalities or types of data, such as text, images, audio, and video. Some would argue that multi-modal models are crucial towards advancing Artificial General Intelligence (AGI) [268, 59], the holy grail of AI research, where a machine can understand, learn, and apply knowledge across a wide variety of tasks at a level comparable to or even surpassing human intelligence.

2.3.3 ML for optimization

In this section, we describe some of the subfields of ML for optimization. For an in-depth review, the interested reader is referred to [31], and to [68, 331] for reviews focused on CO and system identification respectively.

A key observation that enables ML for optimization is that much of operational decision-making involves the regular solution of subsequent optimization instances with similar problem structures, formulations, and data realizations. We essentially consider the optimization problems

to be parameterized by context information $\mathbf{s}$:

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^{n_x}} . \quad & f(\mathbf{x},\mathbf{s}) \\ \text{s.t.} \quad & g_l(\mathbf{x},\mathbf{s}) \leq 0, \quad l = 1,\ldots,n_g, \\ & h_{l'}(\mathbf{x},\mathbf{s}) = 0, \quad l' = 1,\ldots,n_h. \end{aligned} \tag{2.28}$$

For example, in planning, the context $\mathbf{s}$ might involve prices, demand forecasts, and initial inventory levels. Note that (2.28) could in principle be cast in the form of a decision rule as introduced at the very beginning of the chapter in (2.1): $\mathbf{x}^* = \mathcal{D}(\mathbf{s})$. We now describe various ways that we can improve solution tractability or accuracy with ML.

ML for model reduction

A straightforward way of using ML for optimization entails using easier-to-handle surrogates to replace high-fidelity, fine-grained expressions to reduce model complexity and solution time [40, 38, 355]. For instance, in planning formulations, one could replace tiered, nonlinear pricing equations ($h_{l'}(\mathbf{x},\mathbf{s}) = 0$) to obtain cost $\mathbf{x}_c \subset \mathbf{x}$ as a function of demand $\mathbf{s}_d \subset \mathbf{s}$ with a simpler surrogate $\mathcal{M}$: $\mathbf{x}_c = \mathcal{M}(\mathbf{s}_d)$ that reduces online optimization time. Surrogates abstract detailed, lower-level constraints into a simple input-output mapping that is easier to handle by the optimization solver: $\mathcal{M}(\mathbf{s}) \leq 0$. This particular example might only require linear regression, and more involved models might even worsen tractability after integration into the optimization formulation. However, surrogates become especially important when integrating lower-level decision-making considerations into higher levels [376] as discussed in the hierarchical decision-making example from Section 1.2.

Optimization over manifolds

In Section 2.2.3, we briefly touch on subspace optimization [242] as an alternative way of reducing model complexity. Instead of learning surrogates to replace difficult-to-handle model expressions in the original space, we can learn an encoder-decoder structure $\hat{\mathbf{x}} \approx \mathcal{R}(\mathbf{z}) = \mathcal{R}(\mathcal{E}(\mathbf{x}))$, that allows us to perform optimization in the lower-dimensional, active embedding space $\mathcal{Z}$ [267, 144]. Optimizing in this subspace holds the promise of breaking the curse of dimensionality by limiting the scaling to the number of active dimensions n_z.

In principle, any dimensionality reduction technique can be used as long as it prevents exces-

sive information loss when reconstructing the solution from the subspace to the original space. While non-linear projections can discover more informative embeddings, they often create highly nonconvex manifolds that are difficult to optimize over with gradient-based solvers. The presence of bounds and constraints in the original problem might also severely restrict the choice of suitable subspace techniques.

End-to-end or 'amortized' learning of optimization models

Related to the use of surrogates for model reduction are 'optimality' surrogates. We can use surrogates to map optimal quantities rather than feasible spaces from context information $\mathbf{s}$, which would essentially correspond to building a surrogate over the decision rule directly: $\hat{\mathbf{x}}^* = \mathcal{M}(\mathbf{s}) \approx \mathcal{D}(\mathbf{s})$. This concept is related to amortized learning [13], optimal explicit control laws [29] and ADP [308]. In optimal control, we could for example learn to predict optimal cost or optimal control variables directly as a function of production requirements, instead of extracting these quantities from the expensive solution of MPC problems.

Imitation Learning and Reinforcement Learning

There are two major ML frameworks that can be used to train policies to map optimal quantities as a function of optimization context or observed states $\mathbf{s}$ [31]: RL and Imitation Learning (IL). In the latter, we have a high-fidelity optimization solver, and want to improve online solution time at the potential expense of accuracy. We can first query the 'expert' to return the optimal quantities corresponding to a set of optimization instances sampled from distributions over data realizations and problem topologies. Then, we can train a surrogate to mimic the expert as closely as possible. However, we generally cannot outperform the expert via IL. In RL, we train a policy to maximize cumulative rewards by interacting with the system directly. While there are some success stories for applying RL to physical systems, such as the industrial cooling of datacenters [244], deploying RL on safety-critical systems is still limited by trustworthiness [140, 120].

Black-box optimization

The previous fields revolve around conventional mathematical programming by using surrogates to replace model formulations, or to predict quantities of interest directly. The field of black-box optimization in contrast abstracts system performance into an input-output mapping. The

inputs correspond to the the degrees of freedom $\mathbf{x}_d \subseteq \mathbf{x}$, such that all other quantities in $\mathbf{x}$ can be derived from the equality constraints $\mathbf{h}(\cdot)$. BBO is a zeroth-order optimization method (Section 2.2.3) with the aim to identify the optimal inputs $\mathbf{x}_d$ that optimize the function objective without explicit access to gradient information.

BBO either relies on the intermediate construction and optimization of surrogates, or purely on function evaluations for update steps [228]. While BBO, closely related to DFO, is not usually thought of as ML, using uncertainty-aware Gaussian Processes for BBO gives rise to BO [144, 312] - a sub-field of AL [336] crucial for Automated Machine Learning (AutoML) [373]. BBO has successfully been applied to optimize chemical engineering systems [47], and can even be used in conjunction with mathematical programming in the solution of multi-level optimization problems [37].

Learning to optimize: from the dual perspective

In Section 2.3.3, we touch on how we can use RL to optimize policies. Crucial to RL is the concept of Markov Decision Process (MDP), which describes sequential decision-making (control) problems as a sequence of state-action-reward tuples. Instead, we can also view optimization solvers as MDPs [309]. In MIP solvers for example, problem and solver information could represent states, upper and/or lower bounds on the function evaluations could represent rewards, and the set of potential nodes to branch could represent actions. These methods could be characterized as dual [68], as they still preserve guarantees for proving optimality. The intersection of ML and MIP has given rise to numerous research avenue: learning to decompose, cut, branch, or relax [375, 367, 109, 25, 168]. Learning to decompose, for instance, involves discrete action spaces that represent various decomposition strategy choices. Similarly, our action spaces can determine which cuts to implement, the search routine to follow, or whether to relax specific expressions.

Learning to optimize: from the primal perspective

Cappart et al. [68] would characterize the direct inference of optimal decision variables or objectives using optimality surrogates or amortized learning as primal. Primal methods are preferred when a *good* solution should be obtained *quickly* without requiring optimality guarantees. For instance, we could try to predict initial guesses to CO instances from problem information $\mathbf{s}$. Similarly, neural CO can iteratively construct the solution to Knapsack and Traveling Salesman problems from

scratch using policies trained via IL or RL [28, 142]. Even unsupervised or contrastive learning techniques can be developed for specific problem classes at hand [374, 119, 119].

Meta-learning [180], often referred to as 'learning to learn', becomes key to learning solution approaches that improve generalization across problem classes and sizes. This usually involves learning from previous experiences, and leveraging context or 'metadata' about the problem instance.

Learning optimization algorithms

Chen et al. [82, 83] extensively review the field of learning to optimize. They distinguish between model-free and model-based methods. Model-free approaches learn parameterized update rules $\mathbf{x}^{k+1} = \mathcal{U}_\theta(\mathbf{x}^k, \mathbf{s})$ with the updates not bound to analytic expressions. They leverage different orders of gradient and other problem information. Model-based methods on the other hand integrate deep learning architectures into the analytic expressions of conventional optimization algorithms. This includes 'meta-learning' of hyperparameters such as the step size in BFGS [123] and decay rates in Adam [397]. Recent trends involve algorithm unrolling and plug-and-play approaches. For instance, Aharon et al. [5] replace one of the proximal operators in the distributed optimization algorithm ADMM with a denoiser.

Similar techniques exist for local search and BBO. [202] review the use of ML for local search, where learned operations range from operator selection, to parameter control, variable and/or value selection for candidate generation, or mutation operators. We can even perform meta-black-box-optimization by learning operators that mimic optimization strategies such as genetic or evolutionary algorithms [225, 224].

What is not included

The above discussion is by no means exhaustive. When faced for example with a context-aware planning problem in the form of (2.28), once the context $\mathbf{s}$ is known, the resulting LP can be solved relatively efficiently. However, when determining the context $\mathbf{s}$ presents to be challenging, ML is often key in forecasting $\mathbf{s}$ in the first place [69, 419]. When no nominal estimates of $\mathbf{s}$ are available, ML has also enabled the solution of optimization problems under uncertainty [281] with techniques such as distributionally robust optimization. Community detection is another interesting avenue, which combines ML and network theory to find structure in mathematical optimization problems

and suggest optimal decompositions [261, 107] for distributed optimization. Fault or anomaly detection plays a crucial role in process monitoring [88], and last but not least, interest in ML for drug or molecular discovery transcends the PSE community [104, 258].

Perspectives on future trends

In the above discussion, while the scope is centered around the needs of this thesis, the goal is to provide an overview of key trends in ML for optimization. In what follows, I diverge to offer a personal perspective on promising avenues not covered in this thesis. Please note that some of these statements may become outdated as ML continues to advance.

Computational irreducibility is a concept introduced by Stephen Wolfram [404], which states that certain systems cannot be predicted without performing every step of their evolution. In other words, there are problems for which no computational shortcut exists. Many scientific and decision-making problems involve identifying 'pockets of computational reducibility' to manage the complexity and tractability of solving certain systems. AI has emerged as a powerful paradigm for uncovering new pockets of tractability. For instance, AlphaFold [1] has revolutionized protein folding predictions, turning what was once a computationally intractable problem into a more manageable one.

Machine Learning for combinatorial optimization has been largely confined to specific families of problem classes. To date, there have been limited successes in applying ML to 'industrial' problems beyond controlled academic problem sets. If a ML tool could improve the tractability of solving entire classes of combinatorial problems, it would have profound implications across diverse fields, from operations research to bioinformatics. The presence of computational irreducibility in complex systems implies that there must be infinite pockets of reducibility. In other words, there are always properties that can be inferred with limited computational expense. The question remains if these are accessible to the current or emerging family of models [405].

Currently, pure learning-based approaches struggle to consistently outperform traditional OR tools across problem instances since the latter are purpose-built to exploit problem information and mathematical structure for efficiency. However, OR workflows are limited by the need to translate natural inputs into an abstract form that can be manipulated by the algorithm. For example, while it might be convenient to model delivery routing as a Travelling Salesman Problem (TSP),

simplifying the complexities of the real-world into edge distances between nodes misses out on the details of traffic. Neural algorithmic reasoning [391, 147] promises to hybridize these approaches by first learning to imitate (steps of) OR algorithms and then jointly learning (linear) transformations from natural to algorithmic inputs and algorithmic to natural outputs.

Generative models such as generative flow networks have shown great potential in molecular and drug discovery [188]. These models have also successfully been used as heuristic combinatorial optimization solvers for multi-modal or multi-objective applications [418, 210]. Their broader applicability for black-box optimization should be further investigated.

Transformers have demonstrated strong results in abstracting the learning and training process, predicting test labels directly from training data and test features. This approach has outperformed the best AutoML frameworks on mid-sized tabular data [177]. Ongoing research explores their potential for AL in BBO [272]. Transformers have also enabled the huge-scale training and deployment of LLMs. Lange et al. [226] find preliminary success in using LLMs as evolution strategies for 'in-context' learning of small synthetic BBO tasks.

Large language models have shown to be powerful tools for streamlining scientific and coding workflows. A competitive effort is underway to develop scientific assistants that can help with or even automate tasks ranging from hypothesis generation to literature review, code generation, data analysis, and scientific writing [420, 216]. However, key challenges in trustworthiness — such as hallucinations — must be addressed before safe adoption in organizational decision-making [41]. In chemical companies, operational decision-making is often constrained by business considerations, which leads to a heavy reliance on humans-in-the-loop. In causal decision-making for example, LLMs are used more reliably for workflow orchestration, augmented information retrieval, and hypothesis generation, rather than pure inference [417, 191, 192, 421]. Alternatively, LLMs can be integrated into decision-making pipelines to complement human workflows [329, 162]. Finally, research into AI ethics and value alignment is crucial to ensure that AI tools are safe and compatible with human values, especially if they become increasingly powerful to the point of achieving AGI [138].

2.3.4 How Machine Learning and optimization are used in this thesis

DFO serves as the unifying thread for the methodologies in Chapters 3 to 6, and has a rich tradition in OR and ML. Known also as BBO [228], DFO is applied here to coordinate the integration of previously separated optimization models. BBO has a strong presence in ML such as in AutoML. Optimizing systems without explicit access to models or gradient expressions itself is a special case of AL. In Chapters 3 and 6, Bayesian optimization presents the state-of-the-art in sample-efficient solvers that we benchmark against.

While BO is firmly grounded in statistical learning, most solvers used in Chapters 3 to 5 are more rooted in the optimization and operations research communities than in ML. Our algorithmic contributions to DFO in the form of CUATRO in Chapters 3 and 6 especially rely heavily on convex quadratic programming frameworks and solvers.

We wrap up the chapter by linking our contributions to the decision-making paradigms from PSE, OR, ML, and control that were introduced throughout.

Machine Learning

In Chapter 6, PLS, used as dimensionality reduction technique in our high-dimensional DFO solver, has a long history in chemoinformatics [243], and is considered part of the core canon of ML. Subspace optimization is also common in high-dimensional BO.

In Chapter 4, the problem we address has attracted interest from both the optimization and ML communities. Decomposition techniques and distributed optimization [289] optimize decisions between subproblems without needing explicit access to subproblem model expressions or data. In ML, collaborative model training under privacy is studied in Federated Learning [319, 198].

In Chapter 5, we address the hierarchical planning-scheduling-control problem from Section 1.2 using optimality surrogates. This concept appears across many fields. In the control literature, similar surrogates take the form of optimal explicit control laws [29] and ADP [308]. In our case, we use ‘IL to learn expert solutions demonstrated by mathematical programming solvers. By using cheaper surrogates, we amortize the online cost of optimization by solving and learning from similar problems offline.

Policies in RL can also serve as optimality surrogates, but are trained directly using rewards. In value chain coordination as described in Section 1.2 and Chapter 4, we could also train surro-

gates under organizational considerations to return optimal complicating material flows between coordinating sites. While not included in this thesis, we compare shared and agent-specific policies trained in different multi-agent reinforcement learning settings against distributed optimization for stochastic decentralized inventory control [269].

Mathematical programming

In the development of CUATRO throughout Chapter 3, we formalize surrogate update steps based on well-established trust-region frameworks from the numerical optimization and DFO literature. Roughly speaking, we could consider CUATRO to be a data-driven approximation to a trust region-based sequential quadratically constrained quadratic programming framework. In Chapter 6, we combine sketching and hashing techniques, useful in solving huge-dimensional mathematical programming problems, with CUATRO to solve high-dimensional DFO problems.

In Chapter 3, even though real-time optimization, model-based design of experiments, and self-optimizing reactors can be solved using mathematical programming, we reformulate them as black-box input-output evaluations.

In Chapter 4, we benchmark a data-driven framework for the coordination of decentralized subproblems against the flagship distributed optimization method ADMM. In both paradigms, we can formalize the coordination problem using a bi-level formulation, and solve the upper-level coordination step without having to rely on mathematical programming. However, we still employ nonlinear or mixed-integer programming to solve the subproblems in the collaborative model training, facility location, and multi-objective blending case studies.

In Chapter 5, we tackle a hierarchical planning-scheduling-control problem by considering a combination of optimality surrogates and DFO to solve a multi-level formulation. The optimality surrogate approach embeds lower-level information as a surrogate into the upper-level planning formulation, resulting in a single mixed-integer linear programming formulation. Additionally, mixed-integer linear, and nonlinear programming is still used for the solution of the scheduling and control problems in simulating the objective obtained in the tri-level planning formulation in the DFO approach, and in sampling training data for the optimality surrogates.

Overall, the contributions of this thesis illustrate how classical optimization and state-of-the-art Machine Learning techniques can be unified to address a diverse array of coordination problems under organizational considerations.

3

Derivative-free optimization for process systems engineering applications

Contents

3.1 Introduction . . . 46
3.1.1 Categorization of derivative-free optimization solvers . . . 47
3.1.2 Previous work in the chemical engineering community . . . 48
3.1.3 Our contribution . . . 49
3.2 Methodology . . . 50
3.2.1 Problem statement . . . 50
3.2.2 Steepest descent via finite difference approximations . . . 50
3.2.3 Py-BOBYQA . . . 52
3.2.4 Nesterov random Search . . . 53
3.2.5 SQSnobfit . . . 54
3.2.6 DIRECT-L randomized . . . 55
3.2.7 Simplex method . . . 55
3.2.8 Bayesian Optimization (BO) . . . 56
3.2.9 Constraint handling . . . 58
3.3 CUATRO . . . 58
3.3.1 Surrogate fitting . . . 59
3.3.2 Explicit constraint handling . . . 60
3.3.3 Surrogate minimization . . . 61
3.4 Case Studies . . . 63
3.4.1 Problem considerations and selection of case studies . . . 63
3.4.2 Constrained Rosenbrock . . . 64
3.4.3 Real-time optimization . . . 64
3.4.4 Model-based design of experiments . . . 66
3.4.5 Self-optimizing reactions . . . 67

3.4.6 Controller synthesis 68
3.4.7 Flowsheet optimization 71
3.5 Results and Discussion 73
3.5.1 Algorithms 73
3.5.2 Performance of DFO algorithms on the case studies 78
3.5.3 Performance of DFO algorithms at increasing levels of stochasticity . . . 87
3.5.4 Constraint Satisfaction 90
3.6 Conclusion 94
3.6.1 Looking Back 95

In this chapter, we benchmark both model-based and direct state-of-the-art derivative-free optimization (DFO) algorithms against finite difference-based methods. This comparison is conducted on one mathematical optimization problem and five chemical engineering applications: model-based design of experiments, flowsheet optimization, real-time optimization, self-optimizing reactions, and controller tuning. Additionally, we introduce CUATRO, a convex quadratic trust region optimizer with explicit constraint-handling capabilities, whose development is tailored to expensive, high-dimensional, safety-critical, and noisy process systems engineering 'black-boxes'. This chapter, along with the associated Appendix A, is adapted from work published in [385]. Figures in this chapter are reused with permission under the author's rights in accordance with Elsevier's policies.

3.1 Introduction

At the heart of process and supply chain operation lies optimization. The traditional decision-making approach in chemical and process engineering leverages algebraic expressions whose derivative information is used for gradient-based optimization (Section 2.2.3). When trying to leverage collected process data for decision-making, this data usually serves to develop models to be used within this framework. However, in other cases, the organically collected data already represents expensive black-box evaluations of the quantities of interest.

In fact, in many chemical engineering optimization problems, the objective and constraint functions are only available as the output of black-box evaluation functions or simulation oracle queries. These function calls tend to be prohibitively expensive in terms of cost or time, meaning that the number of function evaluations is severely limited. Examples of such instances include problems ranging from stochastic simulation and multi-level optimization involving rigorous (partial) differential algebraic equations, to the design of physical experiments and even the process of model creation itself. Conventional optimization frameworks requiring explicit gradient information or analytic function expressions for automatic differentiation cannot be used when the

solution requires queries to physical experiments or proprietary software. The lack of derivative information then calls for derivative-free, also called zeroth-order or 'data-driven' optimization techniques (Section 2.2.3), or the finite difference approximation of derivatives [259]. In the case where optimization speed is limited by the number of function evaluations, approximating first- and second-order derivative information through finite differences scales poorly with the number of dimensions, and can lead to poor accuracy.

3.1.1 Categorization of derivative-free optimization solvers

At present, derivative-free optimization (DFO) techniques can be categorized into 1) direct and model-based, 2) local and global exploration, 3) deterministic, randomized, and hybrid methods [228]. On the one hand, direct-search methods, such as Nelder-Mead (simplex) [275], directional direct search, generalized pattern or mesh adaptive search algorithms, directly compare function values. On the other hand, model-based methods rely on the construction and optimization of objective function surrogate models for candidate updates. Model-based methods essentially correspond to surrogate optimization techniques [38, 34, 211, 370, 334]. Derivative-free optimization does not rule out the use of conventional derivative-based solvers for the optimization of surrogate models, and only refers to the lack of gradient information of the original expressions.

Before we dive into our classification of DFO solvers into methods exhibiting local or global exploitation-exploration patterns, we want to define what local convergence, global convergence, and global convergence to local minima means. Quasi-Newton methods like BFGS are locally convergent to a local minimum under smoothness conditions *if* the initial guess is sufficiently close. Some pattern search algorithms are globally convergent to local minima, i.e. they can reach a local minimum from any starting point. Other methods display global convergence, i.e. convergence to a global optimum.

However, we consider DFO methods to be 'local' or 'global' when they tend to exploit only their vicinity or the whole solution space instead. We maintain this classification to be consistent with our labels of the proposed CUATRO approach when displaying exploitative or explorative behaviour. We are explicit when referring to convergence *guarantees*. For instance, we would consider particle swarm optimization to be 'global', as the method uses heuristics with the aim to find a global optimum, but lacks theoretical guarantees. Bayesian optimization using radial basis function [99] or Kriging (Gaussian process) surrogates [65, 149] display both global search patterns *and* asymptotic global convergence. Partition-based methods like DIRECT [196, 137] also

show asymptotic convergence to the global optimum for deterministic problem formulations under assumptions on the Lipschitz continuity of the objective. In model-based DFO, local methods may construct and optimize surrogates within trust regions rather than the entire search space. The conditions under which the additional function evaluations of global search space exploration methods are worth the potential increase in solution quality often remain problem-specific.

A final distinction is drawn between deterministic and randomized methods, depending on if there is an inherent stochastic element to the method, for example in polling for the search directions. These distinctions are often purely constructed for the purpose of mathematical analysis, as hybrid methods are often powerful in practice.

While DFO considers the optimization of black boxes with deterministic outputs, Simulation optimization (SO) deals with the optimization of stochastic simulation outputs. Amaran et al. [12] review SO techniques for problems with continuous and discrete variables. While many algorithms for problems with continuous solution spaces have analogs in DFO, the authors argue that theoretical convergence scales linearly with the inverse of the number of samples. Thus, most practical noisy applications of DFO algorithms terminate after a fixed function evaluation budget rather than convergence to rigorous theoretical criteria.

3.1.2 Previous work in the chemical engineering community

DFO, also known as zeroth-order or data-driven optimization, has a long history in chemical engineering. Model-based methods have attracted a lot of attention in process engineering, given the discipline's expertise in surrogate modelling and more general model reduction [40, 47, 402, 355] as discussed in Section 2.3.3. Local trust-region methods and global Kriging interpolation models have been found to be particularly promising in process design and optimization [121, 48, 171, 39]. Other applications range from multi-scale and multi-level optimization to chemical synthesis, product design and design of dynamic experiments [422, 170, 89, 285, 363].

There is an existing literature gap between the optimization and process systems engineering fields. While the mathematical programming literature usually benchmarks algorithms and software implementations on a range of abstract mathematical test problems [317], the chemical engineering literature often studies a narrow range of DFO methods for specific complex case studies. Previous work thoroughly investigates uncertainty in simulation-based optimization in the form of uncertain model parameters, and uncertainty in the surrogate model types and fitted surrogate model parameters [183]. However, there does not seem to be a systematic study that

includes stochasticity and constraint satisfaction considerations of the leading DFO algorithms on expensive chemical engineering applications.

3.1.3 Our contribution

In this chapter, we propose CUATRO as a convex quadratic trust region DFO solver, and benchmark in-house, and established DFO implementations of the model-based, direct search and finite difference philosophies. We include methods well grounded in theory, alongside more heuristic methods. The case studies are chosen to present the behaviour of typical chemical/process engineering problems while remaining tractable for detailed analysis, and include a constrained test function and five chemical engineering applications.

In the rest of this chapter, we introduce the benchmarked algorithms in Section 3.2: Nelder-Mead (direct, local), Nesterov random Search (random, direct, local), Py-BOBYQA (model-based, local), SQSnobfit (model-based, branching strategy), DIRECT-L (direct, local and global) and Bayesian optimization (model-based, global) as well as the alternative methods based on finite difference approximations, namely approximate Newton, Adam and BFGS. In Section 3.3, we introduce CUATRO and discuss key distinctions to other trust region model-based DFO solvers. In Section 3.4, the synthetic and process engineering case studies are introduced: model-based design of experiments, self-optimization of reaction synthesis, flowsheet optimization, real-time optimization, and controller tuning (PI and explicit control law). In Section 3.5, the algorithms are compared on said case studies based on the assumption that the black boxes that are queried are expensive. This means that the optimum should be found in the lowest number of function evaluations.

We focus our discussion on constraint satisfaction, on convergence patterns, and on how different algorithms can handle increasing levels of stochasticity in the objective and constraint evaluations. We also describe practical algorithm considerations, ranging from their sensitivity to hyperparameters, to their scaling with problem dimension and their off-the-shelf applicability to unseen problem instances. When dealing with expensive and computationally limiting function evaluations, the efficiency of DFO algorithms is not determined by the complexity and evaluation time of the black-box functions but by other problem characteristics. The aim is to provide practitioners with the necessary insight to select an algorithm based on the dimensionality of the problem, conditioning of the solution space, and the presence of stochasticity.

3.2 Methodology

3.2.1 Problem statement

Consider the constrained, box-bound optimization problem of the following form:

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^{n_x}} \; & f(\mathbf{x},\omega) \\ \text{s.t.} \quad & g_l(\mathbf{x},\omega) \leq 0, \quad l = 1..n_g, \\ & \mathbf{x} \in [\mathbf{x}^L, \mathbf{x}^U], \end{aligned} \tag{3.1}$$

where n_g denotes the number of inequality constraints and n_x denotes the dimensionality of $\mathbf{x}$. $\omega \in \mathbb{R}^{n_\omega}$ represents uncertainty. We assume no prior knowledge about ω. It should be noted that, while the treatment of equality constraints can be non-trivial, we do not explicitly account for the handling of equality constraints in this work. Some options include relaxation into a set of inequality constraints, or implicit handling by reducing the dimensions of the input variables, and solving for the missing degrees of freedom as part of the black-box formulation. For mixed-integer surrogate optimization, the reader is directed to [12, 35, 49, 381].

3.2.2 Steepest descent via finite difference approximations

In this section, we present algorithms that directly approximate first- and/or second-order derivatives via finite differences. If gradient information is not readily available, it seems intuitive to simply approximate the derivatives and use them within the framework of gradient-based solvers. Finite difference derivative approximations estimate the first- and second order derivatives at a point $\mathbf{x} \in \mathbb{R}^{n_x}$ using functions evaluations of surrounding, perturbed points. Thus, the forward difference approximation of the partial derivative of f with respect to the ith dimension of $\mathbf{x}$ using step size h is defined as follows:

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = \frac{f(x_1, ..., x_i + h, ..., x_{n_x}) - f(x_1, ..., x_i, ..., x_{n_x})}{h}. \tag{3.2}$$

Similarly, the second derivative of f with respect to x_i and x_j can be approximated using:

$$\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_{i'}} = \frac{f(\mathbf{x} + h_i + h_{i'}) - f(\mathbf{x} + h_i) - f(\mathbf{x} + h_{i'}) + 2f(\mathbf{x}) + f(\mathbf{x} - h_i - h_{i'}) - f(\mathbf{x} - h_i) - f(\mathbf{x} - h_{i'})}{2h_i h_{i'}}, \tag{3.3}$$

where h_i denotes the step length in the ith dimension. The interested reader is referred to A.1.1 for more details on finite difference approximation schemes. Three gradient-based optimization algorithms are considered here: Newton's method, BFGS, and Adam. All of these methods constitute steepest descent-type algorithms of the form (2.20) with distinctions in how first- and/or second-order derivative information is obtained. A generalisation of all steepest descent methods is presented in Algorithm 1. Given the absence of known functions for the constraints (only samples are available), these are handled via backtracking [283].

Newton's method

Newton's is a second-order method that requires the approximation of the gradient and the Hessian at each iteration. The second-order information helps in capturing the curvature of the problem when computing the search direction. This should lead to convergence in fewer iterations. The interested reader is referred to A.1.2 for an in-depth discussion of how the derivative information is obtained.

BFGS

BFGS, named after Broyden, Fletcher, Goldfarb, and Shanno, is the most popular quasi-Newton algorithm [283]. It seeks the same search direction as Newton's method, but avoids the repeated computation and inversion of the Hessian. After an initial finite-difference approximation of the Hessian, subsequent iterations of the inverse of the Hessian are obtained based on current (finite-difference) gradient information. BFGS is known to be efficient even in the case of non-differentiable and mildly noisy functions [46]. The interested reader is referred to A.1.3 for an in-depth discussion of how the derivative information is obtained.

Adam

Adam is an efficient stochastic optimization algorithm requiring only first-order gradient information. The name is derived from adaptive moment estimation [212]. As the name suggests, Adam uses bias-corrected estimates of the first- and second-order momentum, making the algorithm efficient for non-stationary objectives, and probably the most popular choice in deep learning. The interested reader is referred to A.1.4 for an in-depth discussion of how first-and second-order momentum is obtained from gradient information, and how momentum is used to compute the search

step.

Steepest descent

3

All three previous methods are steepest descent-type algorithms. Steepest descent methods involve the computation of a decreasing search direction $\mathbf{s} \in \mathbb{R}^{n_x}$ at each iteration based on first-order gradient information, potentially corrected by momentum and second-order information. An update step size is then taken in the direction of $\mathbf{s}$, the magnitude of which is determined by a step size parameter $\alpha \in \mathbb{R}$. α can be fixed, but in our case is determined by backtracking. Finite difference approximations are expected to break down in the presence of stochasticity. Especially second-order derivative approximations are sample-inefficient and thus inappropriate for expensive evaluations A.1.1. The momentum correction of Adam and its dependence only on first-order derivatives should improve Adam's performance in the presence of stochasticity and high-dimensional formulations.

Algorithm 1: General steepest descent

Input: Black-box objective f and constraints $g_l, \quad l = 1..n_g$, Initial guess $\mathbf{x}^0$, Step size parameter α, Maximum number of function evaluations $N_{f,max}$

Initialisation: Function evaluation counter $n_f := 1$, Iteration counter $k := 0$

while *($N_{f,max} \geq n_f$)* **do**

  Compute search direction $\mathbf{s}$ as discusssed in A.1 ;

  $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k + \alpha\mathbf{s}$ (Use backtracking to find α)

  Record all n_s function evaluations ; $n_f \leftarrow n_f + n_s$; $k \leftarrow k + 1$;

end

3.2.3 Py-BOBYQA

Py-BOBYQA is a Python implementation of Powell's BOBYQA [307]. At each iteration k, it constructs a quadratic regression-interpolation model for the objective, and determines the best search step $\mathbf{s} \in \mathbb{R}^{n_x}$ by minimizing said model within a trust-region framework:

$$m^k(\mathbf{s}) = r^k + \mathbf{q}^{k\intercal}\mathbf{s} + \frac{1}{2}\mathbf{s}^\intercal P^k \mathbf{s}, \tag{3.4}$$

where $r^k \in \mathbb{R}$, $\mathbf{q}^k \in \mathbb{R}^{n_x}$, and $P^k \in \mathbb{R}^{n_x \times n_x}$. This can be interpreted as using regression or interpolation for the construction of the 2nd order Taylor approximation, similar to (A.4). Determining those coefficients however takes at least $\frac{(n_x+1)(n_x+2)}{2}$ function evaluations $(X^k, \mathbf{y}^k)$, which can become restrictive at higher dimensions. To this end, the user can choose any number of evaluations n_D between $n_x + 1$ and $\frac{(n_x+1)(n_x+2)}{2}$ for smooth and noisy problems respectively and instead solve

the following problem to determine the parameters:

$$\begin{aligned} \min_{r^k, \mathbf{q}^k, P^k} \quad & ||P^k - P^{k-1}||_F^2 \\ \text{s.t.} \quad & m^k(\mathbf{x}_j - \mathbf{x}^k) = y_j, \quad \forall j = 1..n_D \text{ where } (\mathbf{x}_j, y_j) \subset (X^k, \mathbf{y}^k), \end{aligned} \tag{3.5}$$

where $\mathbf{x}^k$ is the best sample found at iteration k, $(\mathbf{x}_j, y_j)$ denote the jth of n_D input and output samples in $(X^k, \mathbf{y}^k)$, and $||\cdot||_F$ represents the Frobenius or L_2 norm. Py-BOBYQA also offers the heuristic option of multiple restarts to try and escape local optima. The method is expected to be sample-efficient and deal well with stochasticity. For an in-depth technical description of the algorithm and trust-region framework see Cartis et al. [71]. Since Py-BOBYQA does not explicitly handle constraints, these are handled using the penalty method (Equation 3.11).

3.2.4 Nesterov random Search

Nesterov random search is a derivative-free optimization algorithm analysed extensively in Nesterov and Spokoiny [278], where it gained attention for being possibly the first algorithm with an associated worst-case complexity in the case where f may be both non-smooth and non-convex. The method approximates the derivative via an approximation of the expectation of the smoothed gradient. This approach enables analysis of the algorithm in the case of stochastic, non-convex functions, as the Gaussian smoothing technique used to generate the gradient approximation maintains many key properties of the function.

First, Gaussian smoothing of a function f is hypothesized. This smoothing is parameterized by a smoothing parameter η:

$$f_\eta(\mathbf{x}) = \sqrt{\frac{\det(C)}{(2\pi)^{n_x}}} \int_{\mathbb{R}^{n_x}} f(\mathbf{x} + \eta\mathbf{s}) \exp\left(-\frac{1}{2}\mathbf{s}^\intercal C\mathbf{s}\right) \mathrm{d}\mathbf{s}, \tag{3.6}$$

where $\eta \in \mathbb{R}, \mathbf{s} \in \mathbb{R}^{n_x}$, and $C \in \mathbb{R}^{n_x \times n_x}$ is a valid covariance matrix. To derive a descent step of the algorithm, this smoothed function is differentiated to generate the following [228]:

$$\nabla f_\eta(\mathbf{x}) = \frac{1}{\eta}\sqrt{\frac{\det(C)}{(2\pi)^{n_x}}} \int_{\mathbb{R}^{n_x}} (f(\mathbf{x} + \eta\mathbf{s}) - f(\mathbf{x})) \exp\left(-\frac{1}{2}\mathbf{s}^\intercal C\mathbf{s}\right) C\mathbf{s}\mathrm{d}\mathbf{s}, \tag{3.7}$$

which is subsequently approximated as

$$\nabla f_\eta(\mathbf{x}) \approx (f(\mathbf{x} + \eta\mathbf{s}) - f(\mathbf{x}))C\mathbf{s}, \tag{3.8}$$

where $\mathbf{s}$ is generated from a Gaussian distribution of the form $\mathcal{N}(\mathbf{0}, C^{-1})$. Nesterov random search is presented in Algorithm 2. It is expected to deal well with stochasticity, but theoretical results suggest that finite difference methods are more efficient than smoothing methods in providing good gradient approximations [32].

Algorithm 2: Nesterov random search

Input: Black-box objective f and constraints $g_l, l = 1..n_g$, Initial guess $\mathbf{x}^0$, Step size α, smoothing parameter η, covariance matrix C, Maximum number of function evaluations $N_{f,max}$

Initialisation: Initial data sets $X^0 := [\mathbf{x}^0]$ and $Y^0 := [\{(f(\mathbf{x}^0), g_l(\mathbf{x}^0)), l = 1..n_g\}]$, Function evaluation counter $n_f := 1$, Iteration counter $k := 0$

while *($N_{f,max} \geq n_f$)* **do**

  $\mathbf{x}^k \leftarrow \arg\min_{\mathbf{x}_j \in X^k} f(\mathbf{x}_j) + \gamma \sum_{l=1}^{n_g} \max(0, g_l(\mathbf{x}_j))^2$;

  Generate $\mathbf{s}^k \in \mathcal{N}(0, C^{-1})$;

  $d(\mathbf{x}^k, \mathbf{s}^k) \leftarrow (f(\mathbf{x}^k + \eta \mathbf{s}^k) - f(\mathbf{x}^k))C\mathbf{s}^k$;

  $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \alpha C^{-1} d(\mathbf{x}^k, \mathbf{s}^k)$;

  Update datasets with n_s new evaluations X_{n_s}:

  $X^{k+1} \leftarrow [X^k, X_{n_s}], Y^{k+1} \leftarrow [Y^k, \{(f(X_{n_s}), g_l(X_{n_s})), l = 1 \dots n_g\}]$;

  $n_f \leftarrow n_f + n_s$; $k \leftarrow k + 1$;

end

* γ is a tunable penalty parameter.

3.2.5 SQSnobfit

SQSnobfit is a Python implementation of the Matlab package SNOBFIT (Stable Noisy Optimization by Branch and Fit) by Huyer and Neumaier [184]. The method is intended for the solution of noisy, bound-constrained optimization problems with continuous variables. It combines local and global search by using a branching strategy with local fitting of a linear or quadratic surrogate model to function evaluations. The authors claim that the algorithm is safeguarded against common issues arising within derivative-free optimization methods such as expensive and noisy function evaluations, the evaluation of functions not just at exact but also nearby points, and the presence of hidden and soft constraints. Especially, the method's flexibility to let the user change the bound-constraints in-between function evaluations, and to choose between local and global sampling methods for the next evaluations, lends itself well to expensive simulations and experimental design.

3.2.6 DIRECT-L randomized

We use the randomized DIRECT-L implementation of the NLopt nonlinear optimization package library [194]. This implementation is based on the DIviding RECTangles algorithm for global optimization, originally written in FORTRAN [196]. DIRECT is a Lipschitzian, deterministic search algorithm, based on systematic partitioning of the search space into smaller hyperrectangles. Gablonsky and Kelley [137] then made the algorithm biased towards local search for problems that only have a few local minima. The current implementation uses a randomized version of the locally-biased DIRECT, which randomly decides on the dimension to partition along next when function evaluations are close. The method is not originally intended to be used with stochastic objective functions.

3.2.7 Simplex method

Not to be confused with Dantzig's simplex method used in linear programming [274], this simplex method, also known as the Nelder-Mead method was first presented in its most basic form by Spendley et al. [357]. As is the case with many early derivative-free optimization algorithms, the simplex method is model-free, relying solely on function evaluations. This original implementation was further refined by Nelder and Mead [276] into the algorithm used in this study. Nelder and Mead's method first specifies a simplex defined by $n_x + 1$ linearly independent vertices over the search-space $\mathbb{R}^{n_x}$. By transforming the vertices of this simplex through a series of reflections, contractions, and expansions, the simplex moves through the solution space towards areas of decreasing objective value. The method is not originally intended to be used with stochastic objective functions. Yet, its simple geometric interpretation and extensive implementation in scientific and engineering computing software has ensured its sustained popularity [228]. The algorithm implemented here is as follows:

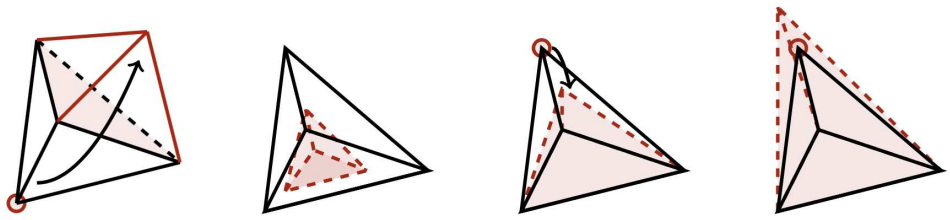


Figure 3.1: The four basic simplex operations: reflect, shrink, contract and expand.

3

Algorithm 3: Simplex method (Nelder-Mead)

Input: Black-box objective f and constraints $g_l, l = 1..n_g$, Initial guess $\mathbf{x}^0$, Maximum number of function evaluations $N_{f,max}$

Initialisation: Initial simplex $S_X^0 := [\mathbf{x}^0]$ and $S_Y^0 := [\{(f(\mathbf{x}^0), g_l(\mathbf{x}^0)), l = 1..n_g\}]$, Function evaluation counter $n_f := 1$, Iteration counter $k := 0$

Generate n_x feasible points X_{n_x} randomly around $\mathbf{x}^0$: $S_X^0 \leftarrow [S_X^0, X_{n_x}], S_Y^0 \leftarrow [S_Y^0, \{(f(X_{n_x}), g_l(X_{n_x})), l = 1..n_g\}]$;

while *($N_{f,max} \geq n_f$)* **do**

  $\mathbf{x}_{\text{worst}} \leftarrow \arg\max_{\mathbf{x}_j} f(\mathbf{x}_j)$ for $\mathbf{x}_j \in S_X^k$;

  $\mathbf{x}_{\text{second-worst}} \leftarrow \arg\max_{\mathbf{x}_j} f(\mathbf{x}_j)$ for $\mathbf{x}_j \in (S_X^k \setminus \mathbf{x}_{\text{worst}})$;

  Reflect $\mathbf{x}_{\text{worst}}$ across the plane $(S_x^k \setminus \mathbf{x}_{\text{worst}})$ to get $\mathbf{x}_{\text{reflected}}$;

  if $f(\mathbf{x}_{reflected}) < f(\mathbf{x}_{second\text{-}worst})$ & $f(\mathbf{x}_{reflected}) > f(\mathbf{x}_{best})$ & $\mathbf{x}_{reflected}$ *feasible* **then**

    $\mathbf{x}_{\text{worst}} \leftarrow \mathbf{x}_{\text{reflected}}$

  else if $f(\mathbf{x}_{reflected}) < f(\mathbf{x}_{best})$ & $\mathbf{x}_{reflected}$ *feasible* **then**

    Expand $\mathbf{x}_{\text{reflected}}$ further across the plane $(S_x^k \setminus \mathbf{x}_{\text{worst}})$ generating $\mathbf{x}_{\text{expanded}}$;

    if $f(\mathbf{x}_{expanded}) < f(\mathbf{x}_{reflected})$ & $\mathbf{x}_{reflected}$ *feasible* **then**

      $\mathbf{x}_{\text{worst}} \leftarrow \mathbf{x}_{\text{expanded}}$

    else

      $\mathbf{x}_{\text{worst}} \leftarrow \mathbf{x}_{\text{reflected}}$

  else

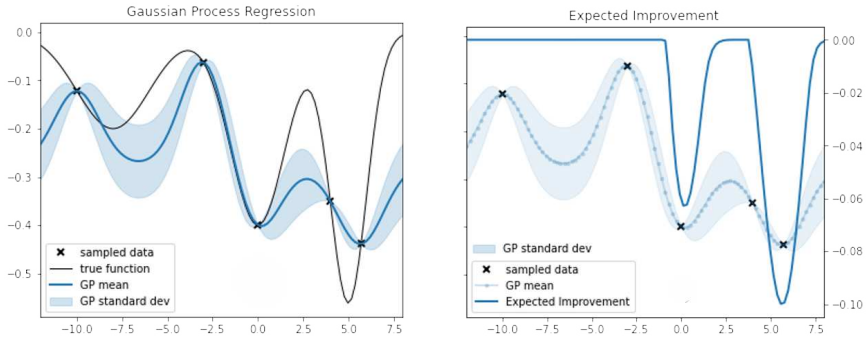
    Contract $\mathbf{x}_{\text{worst}}$ towards the plane of the remaining vertices.

end

3.2.8 Bayesian Optimization (BO)

For our in-house implementation of BO, we use Gaussian process surrogates with zero mean and squared-exponential covariance function. The reader unfamiliar with Gaussian processes (GP) is referred to A.2. GP offer the advantage of modelling both the mean and variance at unseen points with minimal additional cost. After the surrogates are constructed, they are employed within the acquisition function optimization to balance exploration and exploitation in selecting the next sample to query [340]. We use expected improvement (EI) as our acquisition function, motivated by favorable empirical results reported by [353, 340]. Although we only use the GP posterior mean in explicitly modelling constraint evaluations, the variance term can be incorporated for safe exploration [108]. Alternative acquisition functions include lower confidence bounds, probability of improvement [222], knowledge gradient [135], and entropy search [172], each offering unique strategies for navigating the trade-off between exploration and exploitation and improving sample efficiency.

The GP posterior mean and variance function for a sampled function and the corresponding expected improvement (EI) acquisition function are visualized in Figure 3.2. The EI function is



(a) Posterior mean and variance in the GP regression (A.7)

(b) Expected improvement acquisition function (3.9) of GP surrogate.

Figure 3.2: Illustration of GP regression and expected improvement acquisition function of a sampled function.

expressed as follows:

$$\mathcal{A}_{\text{EI}}[\mu_0, \sigma_0, f_{\text{L}}](\mathbf{x}) := -\left[f_{\text{L}} - \mu_0(\mathbf{x})\right] \Phi\left(\frac{f_{\text{L}} - \mu_0(\mathbf{x})}{\sigma_0(\mathbf{x})}\right) - \sigma_0(\mathbf{x})\ \phi\left(\frac{f_{\text{L}} - \mu_0(\mathbf{x})}{\sigma_0(\mathbf{x})}\right), \tag{3.9}$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ denote the standard normal probability density and cumulative distribution functions; $\mu_0(\cdot)$ and $\sigma_0(\cdot)$ denote the GP posterior mean and variance functions of the objective; and $f_{\text{L}} := \min(y_1, \ldots, y_{n_D})$ is the best observed value. This expression corresponds to $\mathbb{E}[\max(f_{\text{L}} - \mu_0(\cdot), 0)]$, where the improvement function is only positive at points where the predicted mean value is lower than f_{L}. In our implementation, we minimize (3.9) at each iteration under the condition that the suggested sample point be feasible in expectation according to all constraint surrogates:

$$\begin{aligned} \min_{\mathbf{x}\in\mathcal{B}} \quad & \mathcal{A}_{\text{EI}}[\mu_0, \sigma_0, f_{\text{L}}](\mathbf{x}) \\ \text{s.t.} \quad & \mu_l(\mathbf{x}) \leq 0, \quad l = 1..n_g, \end{aligned} \tag{3.10}$$

where μ_l denotes the GP posterior mean approximation for constraint $g_l(\cdot)$. If the problem is unconstrained, then only the objective is considered.

3

Algorithm 4: Bayesian optimization

Input: Black-box objective f and constraints $g_l, l = 1..n_g$, initial data sets $(X^0, Y^0 := \{(f(X^0), g_l(X^0)), l = 1..n_g\})$; initial guess $\mathbf{x}^0$; evaluation budget $N_{f,max}$

Initialisation: Train $\mathcal{GP}_l^0$, $l = 0..n_g$ with data sets (X^0, Y^0), iteration counter $k \leftarrow 0$, function evaluation counter n_f

while $(N_{f,max} \geq n_f)$ **do**

  $\mathbf{x}_{new} \leftarrow$ Solution of Problem (3.10) with $\mathcal{GP}_l^k$, $l = 0 \dots n_g$;

  Update datasets $X^{k+1} \leftarrow [X^k, \mathbf{x}_{new}], Y^{k+1} \leftarrow [Y^k, \{(f(\mathbf{x}_{new}), g_l(\mathbf{x}_{new})), l = 1 \dots n_g\}]$;

  $k \leftarrow k + 1$; $n_f \leftarrow n_f + 1$;

  Train $\mathcal{GP}_l^k$, $l = 0 \dots n_g$ with data sets (X^k, Y^k);

end

BO is generally regarded as the go-to method for stochastic, expensive black-box evaluations. Popular research directions explore multifidelity [296] approaches and deep Gaussian processes [101], but their success is case-dependent and will not be discussed further.

3.2.9 Constraint handling

All previously discussed methods aside from BO handle constraints using the penalty method of the L_p norm, usually L_2, which transforms problem (3.1) into:

$$\begin{aligned} \min_{\mathbf{x} \in \mathbb{R}^{n_x}} \quad & f(\mathbf{x}, \omega) + \gamma \sum_{l=1}^{n_g} \max\left(0, g_l(\mathbf{x}, \omega)\right)^p \\ \text{s.t.} \quad & \mathbf{x} \in [\mathbf{x}^L, \mathbf{x}^U]. \end{aligned} \tag{3.11}$$

The penalty parameters γ are tuned to each case study to optimize performance.

In the next section, we propose our in-house solver CUATRO, which handles constraints explicitly like BO, with the exception that CUATRO models them directly using convex quadratic surrogates within a trust region framework, rather than using GP posterior mean functions.

3.3 CUATRO

CUATRO, short for Convex qUAdratic Trust-Region Optimizer, is an in-house implementation of a convex quadratic trust region framework for the solution of constrained problems using semidefinite

programming [4, 111]. CUATRO takes inspiration from convex programming [50], derivative-free optimization with trust regions [94], and efficient global optimization [197].

The initial idea behind CUATRO was to explore whether there were any performance benefits to using *convex* quadratic surrogates within trust-region frameworks in finding the *global* optimum of the surrogates. However, early iterations in the convex programming framework CVXPY [111] mostly revealed other benefits. CVXPY allows for explicit handling of convex constraint surrogates with semi-definite programming solvers and explicit treatment of discrete variables in specific mixed-integer convex programming formulations. Additionally, there is increased flexibility over the trust region frameworks we want to embed the optimization-based update steps into. As an example, CUATRO can be configured by to display more 'global' exploration behaviour by following heuristic trust region update schemes, or to display more 'local', exploitative behaviour by following update schemes that conserve convergence guarantees to local minima. This configurability enables the exploration of interesting combinations of rigorous convex programming frameworks with heuristic routines explored in subsequent work in Chapter 6. Other popular trust region-based DFO implementations are built on [393] and mostly focused on finding the most numerically efficient way to train and optimize surrogates. The interested reader is referred to Neufang et al. [279] for an in-depth comparison and benchmarking with an extended focus on state-of-the-art trust region-based solvers.

3.3.1 Surrogate fitting

The aim is to construct the same approximation of the quadratic Taylor expansion as in Equation (3.4) for the objective function, but also the constraints. To this end, the following convex regression problem is solved at each iteration k to extract the parameters $r^k, \mathbf{q}^k$ and P^k:

$$\begin{aligned} \min_{r^k,\mathbf{q}^k,P^k} \quad & \sum_{j=1}^{n_D} ||y_j - m^k(\mathbf{x}_j)||_F^2 \\ \text{s.t.} \quad & m^k(\mathbf{x}_j) = r^k + \mathbf{q}^{k\intercal}\mathbf{x}_j + \frac{1}{2}\mathbf{x}_j^\intercal P^k \mathbf{x}_j, \quad \forall j = 1..n_D \text{ where } \mathbf{x}_j \subset X^k, \\ & P^k \succeq 0, \end{aligned} \tag{3.12}$$

where $\mathbf{x}_j$ denotes the jth of n_D sample inputs X^k in the trust region $\mathcal{T}(\mathbf{x}^k, \delta^k)$ of center $\mathbf{x}^k$ and radius δ^k. The last constraint defines P^k to be positive semi-definite, which makes the above problem as well as the resulting problem to be optimized in (3.14) convex. The user defines how many function evaluations are required within a given trust region. We find that a default minimum

number of six samples works well across all case studies. These samples are uniformly randomly selected from the intersection of the trust region $\mathcal{T}(\cdot)$ and the user-defined bounds $\mathcal{B} = [\mathbf{x}^L, \mathbf{x}^U]$. In Section 6.1.1 we discuss how we can improve this sampling step by maximizing the distance to the nearest samples and incorporating constraint information. The surrogate regression does not necessarily require $\frac{(n_x+1)(n_x+2)}{2}$ trust region samples.

The alert reader might have already noticed that (3.14) is mathematically underdetermined. In fact, the surrogate construction formulation (3.5) in Py-BOBYQA has unique properties: Instead of sampling up to $\mathcal{O}(n_x^2)$ to fully determine P^k, only a minimum of $\mathcal{O}(n_x)$ candidates are required within the trust region to at least fully determine $\mathbf{q}^k$ and r^k. The remaining degrees of freedom are handled by regularizing P^k to be as close to P^{k-1} as possible. This leads to adaptive, smoother updates in P^k.

While this improves sample efficiency in the trust region framework of Py-BOBYQA, these recursive updates might actually deteriorate performance in CUATRO, where the global configuration may encourage bigger update steps over larger trust regions, where we want P^k to be different from its previous iteration. We found additional empirical benefits to using a purely regression-based approach in that CUATRO becomes more robust towards noise by avoiding interpolation and relying too much on previous updates. We find that the solution to underdetermined formulations of (3.12) can lead to useful updates before committing to spend up to $\mathcal{O}(n_x)$ or $\mathcal{O}(n_x^2)$ samples upfront. However, we could consider other forms of regularization in (3.12) to find simpler surrogates when our problem is underdetermined. We could employ L_1 or L_2 regularization of the coefficients, weighted such that we first find the linear, then the diagonal quadratic, then the remaining quadratic coefficients.

3.3.2 Explicit constraint handling

CUATRO handles the constraints explicitly instead of augmenting the objective with penalty terms. The user has the option to choose between constraint handling using regression or discrimination. When using the regression method, a convex quadratic model is constructed for each of the n_g constraints, like in (3.12), by replacing y_j by $g_l(\mathbf{x}_j)$. In the discrimination case, each data sample $\mathbf{x}_j$ is part of X_U if all constraints are met, and part of X_V otherwise. Then, convex quadratic discrimination is performed to obtain a single constraint surrogate [50]:

$$
\begin{aligned}
\min_{\mathbf{u},\mathbf{v},r_1^k,\mathbf{q}_1^k,P_1^k} \quad & \sum_{j=1}^{n_u} u_j + \sum_{j'=1}^{n_v} v_{j'} \\
\text{s.t.} \quad & r_1^k + \mathbf{q}_1^{k\intercal}\mathbf{x}_j + \frac{1}{2}\mathbf{x}_j^\intercal P_1^k \mathbf{x}_j \leq -(1-u_j), \quad \forall j = 1..n_U \text{ where } (\mathbf{x}_j, u_j) \subset (X_U, \mathbf{u}), \\
& r_1^k + \mathbf{q}_1^{k\intercal}\mathbf{x}_{j'} + \frac{1}{2}\mathbf{x}_{j'}^\intercal P_1^k \mathbf{x}_{j'} \leq (1-v_{j'}), \quad \forall j' = 1..n_V \text{ where } (\mathbf{x}_{j'}, v_{j'}) \subset (X_V, \mathbf{v}), \\
& P_1^k \succ 0, \quad \mathbf{u}, \mathbf{v} \geq 0,
\end{aligned}
\tag{3.13}
$$

where u_l and $v_{l'}$ are the lth and l'^{th} dimension of the dummy variables $\mathbf{u} \in \mathbb{R}^{n_U}$ and $\mathbf{v} \in \mathbb{R}^{n_V}$ respectively.

3.3.3 Surrogate minimization

Finally, the fitted models are minimized within the user-defined bounds, fitted constraints, and trust region radius in a convex optimization problem to find a new candidate point:

$$
\begin{aligned}
\min_{\mathbf{x}\in\mathcal{B}\cap\mathcal{T}(x^k,\delta^k)} \quad & r_0^k + \mathbf{q}_0^{k\intercal}\mathbf{x} + \frac{1}{2}\mathbf{x}^\intercal P_0^k \mathbf{x} \\
\text{s.t.} \quad & r_l^k + \mathbf{q}_l^{k\intercal}\mathbf{x} + \frac{1}{2}\mathbf{x}^\intercal P_l^k \mathbf{x} \leq 0, \quad l = 1..n_g \text{ or } l = 1, \\
& \mathcal{B} = \{\mathbf{x} \in R^{n_x} \mid \mathbf{x} \in [\mathbf{x}_L, \mathbf{x}_U]\}, \\
& \mathcal{T}(\mathbf{x}^k, \delta^k) = \{\mathbf{x} \in R^{n_x} \mid ||\mathbf{x} - \mathbf{x}^k||_F^2 \leq {\delta^k}^2\}.
\end{aligned}
\tag{3.14}
$$

We emphasize that since (3.12) and (3.13) are semidefinite programs, the resulting problem (3.14) is also convex. Therefore, $(2+n_g)$ convex problems are solved for the local and three for the global variation at each iteration. The iterative framework and the conditions for updating the trust region center and radius are presented in Algorithms 5 and 6, for the local and global implementations respectively. The updating scheme for the local implementation is based on Conn et al. [94].

In this chapter, we only describe CUATRO as it was initially implemented in [385]. In Chapter 6, we discuss various heuristic improvements that have been made over the course of this PhD journey to the default solver, and introduce CUATRO_PLS as an extension tailored to high-dimensional applications. The next section describes the case studies in more detail and discusses their mathematical properties.

Algorithm 5: CUATRO local

Input: Black-box objective f and constraints $g_l, l = 1..n_g$, Initial guess $\mathbf{x}^0$, Initial radius δ^0, β_{red}, β_{inc}, η_1, η_2, Minimum number of samples for fitting $N_{s,min}$, Maximum number of function evaluations $N_{f,max}$

Initialisation: Initial data sets $X^0 := [\mathbf{x}^0]$ and $Y^0 := [\{(f(\mathbf{x}^0), g_l(\mathbf{x}^0)), l = 1..n_g\}]$, Sampling region $\mathcal{T}_s(\mathbf{x}^0, \delta^0)$, Trust region $\mathcal{T}(\mathbf{x}^0, \delta^0)$, with $\mathcal{T} \subset \mathcal{T}_s$, Bound-constrained region $\mathcal{B}$, Function evaluation counter $n_f := 1$, Iteration counter $k := 0$

while *($N_{f,max} \geq n_f$)* **do**
- Sample points uniformly from $\mathcal{B} \cap \mathcal{T}_s(\mathbf{x}^k, \delta^k)$ until the region contains $N_{s,min}$ samples ;
- Compute $(P_l^k, \mathbf{q}_l^k, r_l^k)$ from (3.12) for $l = 0..n_g$ using all evaluations in $\mathcal{B} \cap \mathcal{T}_s(\mathbf{x}^k, \delta^k)$;
- Find $\mathbf{x}_{new}$ by minimizing (3.14) within $\mathcal{B} \cap \mathcal{T}(\mathbf{x}^k, \delta^k)$ using $(P_l^k, \mathbf{q}_l^k, r_l^k)$ for $l = 0..n_g$;
- Update datasets with n_s new evaluations X_{n_s}: $X^{k+1} \leftarrow [X^k, X_{n_s}], Y^{k+1} \leftarrow [Y^k, \{(f(X_{n_s}), g_l(X_{n_s})), l = 1..n_g\}]$;
- $n_f \leftarrow n_f + n_s$; $k \leftarrow k + 1$;
- **if** $\neg(g_l(\mathbf{x}_{new}) \leq 0$ *for* $l = 1..n_g) \vee (f(\mathbf{x}_{new}) \geq f(\mathbf{x}^{k-1}))$ **then**
  - Reject update step $\mathbf{x}^k \leftarrow \mathbf{x}^{k-1}$, and decrease radius: $\delta^k \leftarrow \beta_{red}\delta^{k-1}$
- **else**
  - **if** $\frac{\textit{Actual decrease}}{\textit{Predicted decrease}} \geq \eta_2$ *and candidate step is at the trust-region boundary* **then**
    - Accept update step $\mathbf{x}^k \leftarrow \mathbf{x}_{new}$, and increase radius: $\delta^k \leftarrow \beta_{inc}\delta^{k-1}$
  - **else if** $\frac{\textit{Actual decrease}}{\textit{Predicted decrease}} \leq \eta_1$ **then**
    - Reject update step $\mathbf{x}^k \leftarrow \mathbf{x}^{k-1}$, and decrease radius: $\delta^k \leftarrow \beta_{red}\delta^{k-1}$
  - **else**
    - Accept update step $\mathbf{x}^k \leftarrow \mathbf{x}_{new}$, and keep radius $\delta^k \leftarrow \delta^{k-1}$

end

Algorithm 6: CUATRO global

Input: Black-box objective f and constraints $g_l, l = 1 \dots n_g$, Initial guess $\mathbf{x}^0$, Initial radius δ^0, β_{red}, Minimum number of samples for fitting $N_{s,min}$, Maximum number of function evaluations $N_{f,max}$

Initialisation: Initial data sets $X^0 := [\mathbf{x}^0]$ and $Y^0 := [\{(f(\mathbf{x}^0), g_l(\mathbf{x}^0)), l = 1 \dots n_g\}]$, Sampling region $\mathcal{T}_s(\mathbf{x}^0, \delta^0)$, Trust region $\mathcal{T}(\mathbf{x}^0, \delta^0)$, with $T \subset T_s$, Bound-constrained region $\mathcal{B}$, Function evaluation counter $n_f := 1$, Iteration counter $k := 0$

Sample $N_{s,min}$ points using Latin-Hypercube sampling within $\mathcal{B}$;

while *($N_{f,max} \geq n_f$)* **do**
- Compute $(P_0^k, \mathbf{q}_0^k, r_0^k)$ from (3.12) using all evaluations within $\mathcal{B} \cap \mathcal{T}_s(\mathbf{x}^k, \delta^k)$;
- Compute $(P_1^k, \mathbf{q}_1^k, r_1^k)$ from (3.13) using all evaluations within $\mathcal{B} \cap \mathcal{T}_s(\mathbf{x}^k, \delta^k)$;
- Find $\mathbf{x}_{new}$ by minimizing (3.14) within $\mathcal{B} \cap \mathcal{T}(\mathbf{x}^k, \delta^k)$ using $(P_l^k, \mathbf{q}_l^k, r_l^k)$ for $l = 0, 1$;
- Update datasets with n_s new evaluations X_{n_s}: $X^{k+1} \leftarrow [X^k, X_{n_s}], Y^{k+1} \leftarrow [Y^k, \{(f(X_{n_s}), g_l(X_{n_s})), l = 1 \dots n_g\}]$;
- $n_f \leftarrow n_f + n_s$; $k \leftarrow k + 1$;
- **if** $(g_l(\mathbf{x}_{new}) \leq 0$ *for* $l = 1, \dots, n_g) \wedge (f(\mathbf{x}_{new}) < f(\mathbf{x}^{k-1}))$ **then**
  - Accept update step $\mathbf{x}^k \leftarrow \mathbf{x}_{new}$;
- **else**
  - Reject update step $\mathbf{x}^k \leftarrow \mathbf{x}^{k-1}$;
- Reduce radius: $\delta^k \leftarrow \beta_{red}\delta^{k-1}$;
- Sample points uniformly from $\mathcal{B} \cap \mathcal{T}_s(\mathbf{x}^k, \delta^k)$ until the region contains $N_{s,min}$ samples ;

end

3.4 Case Studies

Under the assumption of expensive, computationally limiting function evaluations, the efficiency of DFO algorithms does not depend on the algorithm's solution time (if the per-iteration overhead is lower than the evaluation time) but on the problem characteristics and solution space. In the next section, we justify our selection of case studies based on the problem characteristics that impact algorithm choice.

3.4.1 Problem considerations and selection of case studies

The case studies are defined by the following considerations as they relate to (3.1): a) problem dimensionality, the number of decision variables n_x that represent the degrees of freedom; b) number of constraints n_g and their type (hard or soft); c) stochasticity, the magnitude and dimensionality of ω; d) solution space topology, the 'geometry' of the function including convexity and ill-conditioning. Ill-conditioned functions are defined as having a high condition number, meaning that they can display large variations in the output for small perturbations in the input.

The benchmarking problems are chosen to present various combinations of these mathematical properties, as summarized in Table 3.1.The case studies present the behaviour of typical process engineering problems while remaining tractable and amenable to analysis.

Table 3.1: Case studies and their mathematical properties

Case study	Dimensionality (n_x)	Number of constraints (n_g)	Stochasticity	Topology
Constrained Rosenbruck	2	2	Deterministic + Stochastic	Non-convex
Real-Time Optimization	2	2	Deterministic + Stochastic	Convex
Model-based Design of Experiments	4	0	Deterministic	Unknown
Self-optimizing reactions	2	1	Deterministic + Stochastic	Convex
PID tuning	4	0	Deterministic + Stochastic	Ill-conditioned
Quadratic control law	10	0	Stochastic	Ill-conditioned
Flowsheet optimization	4	4	Deterministic	Nonconvex

The topology of the problem is often unknown, both *a priori* and *a posteriori*. In our case studies, ill-conditioning can arise from non-converging simulation outputs, unstable controls, and artificial constraint handling through barrier methods with high penalty coefficients. Next, we present the case studies in more detail.

3.4.2 Constrained Rosenbrock

The Rosenbrock function is a nonconvex test function with a flat, long, narrow 'valley'. A linear, and a cubic constraint are imposed on the unconstrained function. For the stochastic case, normally distributed noise of mean 0 and standard deviation 0.05 and 0.02 is added to the objective and constraint functions respectively:

$$\begin{aligned} \min_{x_1,x_2} \quad & (1-x_1)^2 + 100(x_2 - x_1^2)^2 \\ \text{s.t.} \quad & (x_1-1)^3 - x_2 + 1 \leq 0, \\ & x_1 + x_2 - 1.8 \leq 0, \\ & -1.5 \leq x_1, x_2 \leq 1.5. \end{aligned} \tag{3.15}$$

This problem is solved using a starting point of $\mathbf{x}_0 = [-1.5, 0.5]^\intercal$ with a function evaluation budget of 100. This two-dimensional solution space serves as a canvass to illustrate convergence patterns of all algorithms in Figure (3.5). This synthetic study is also used to determine default hyperparameters for the methods.

3.4.3 Real-time optimization

The advantages of real-time optimization (RTO) in industrial processes are thoroughly studied [105, 67]. However, currently, most implementations rely on heuristics and trial and error to reach optimality [67]. Real-time optimization is plagued by the difficulty to exactly model complex phenomena. In this context, we refer to the 'plant' as the 'true' system, and to 'plant-model mismatch' as the error of our model with respect to the plant. Unfortunately, the most widely used method in industry (model-adaptation) does not converge to the systems's optimal operating conditions [78]. This has motivated research into alternative RTO solution strategies [78, 377], including the use of derivative-free and BO methods [129, 80, 108].

The optimization of a given plant subject to operational or safety constraints under a stochastic

environment can be formulated as:

$$\min_{\mathbf{u}\in U} G_0^{\mathrm{p}}(\mathbf{u}) := \mathbb{E}\left(g_0\left(\mathbf{u}, \mathbf{y}^{\mathrm{p}}(\mathbf{u})\right)\right) \tag{3.16}$$
$$\text{s.t. } G_l^{\mathrm{p}}(\mathbf{u}) := \mathbb{E}(g_l\left(\mathbf{u}, \mathbf{y}^{\mathrm{p}}(\mathbf{u})\right)) \leq 0, \quad l = 1 \ldots n_g,$$

where $\mathbf{u} \in \mathbb{R}^{n_u}$ and $\mathbf{y}^{\mathrm{p}} \in \mathbb{R}^{n_y}$ are vectors of the plant input (design) and output (measured) variables, respectively. While in the methodology section we use $\mathbf{x}$ instead of $\mathbf{u}$ to denote our degrees of freedom, the notation in the case studies is selected to be consistent with control literature standards. Additionally, $g_l : \mathbb{R}^{n_u} \times \mathbb{R}^{n_y} \to \mathbb{R}$, $l = 1..., n_g$, denote the cost and inequality constraint functions; and $U \subseteq \mathbb{R}^{n_u}$ the control domain, e.g. lower and upper bounds on the input variables, $\mathbf{u}^{\mathrm{L}} \leq \mathbf{u} \leq \mathbf{u}^{\mathrm{U}}$. The superscript $(\cdot)^{\mathrm{p}}$ is used to indicate plant-related quantities. The challenge in RTO is that the exact mapping $\mathbf{y}^{\mathrm{p}}(\cdot)$ is unknown, and that the measurement $\mathbf{y}^{\mathrm{p}}(\mathbf{u})$ is subject to process disturbances.

Williams-Otto benchmark problem

We choose the classical Williams-Otto benchmark as our system of interest. A continuous stirred-tank reactor (CSTR) is fed with two streams of pure components A and B, with respective mass flowrates F_{A} and F_{B}. The reactor operates at steady state at temperature T_{r}. The chemical reactions between these reagents produce two main products P and E, through a series of chemical reactions that also give rise to an intermediate C and a byproduct G:

$$\mathsf{A} + \mathsf{B} \longrightarrow \mathsf{C}$$
$$\mathsf{B} + \mathsf{C} \longrightarrow \mathsf{P} + \mathsf{E}$$
$$\mathsf{C} + \mathsf{P} \longrightarrow \mathsf{G}$$

The complete set of mass balance equations and kinetic rate equations for the reaction system can be found in Mendoza et al. [257].

For the stochastic case, normally distributed noise with mean 0 and standard deviation 0.05 and 0.0005 are added to the objective and constraints respectively. RTO is performed from an initial guess $\mathbf{u}_0 = [6.9, 83]^{\intercal}$ within the bounds of $[[4., 7.], [70., 100.]]$. The first dimension of $\mathbf{u}$ indicates the equimolar flowrates F_A and F_B, whereas the second dimension indicates the reactor temperature T_r. The convexity of the two-dimensional solution space is visualized in Figure 3.3a.

3.4.4 Model-based design of experiments

Model-based design of experiments (MBDoE) has been widely used to improve parameter precision in nonlinear dynamic systems. The aim of MBDoE is to find the design of experiment conditions [133] that are most informative to the parameter fitting. Solving an optimal MBDoE problem can for instance involve the maximization of a measure of the expected Fisher information (or the minimization of a measure of the expected variance of model parameters).

$$
\begin{aligned}
\min_{\mathbf{u}} \quad & \psi\left(V_\theta\left(\mathbf{u}, \theta, t_\mathrm{f}\right)\right) \\
\text{s.t.} \quad & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \theta, \mathbf{u}(t)),\ \hat{\mathbf{y}}(t) = \mathbf{g}(\mathbf{x}(t)), \\
& \mathbf{x}(t) \in \mathcal{X},\ \mathbf{u}(t) \in \mathcal{U}, \quad \forall t \in [0, t_\mathrm{f}], \\
& \mathbf{x}(0) \in \mathcal{X}_0,\ \theta \in \Theta, \\
& V_\theta(\mathbf{u}, \theta, t_\mathrm{f}) = \left[\left[V_\theta^0\right]^{-1} + \sum_{t=t_1}^{t_{N_\mathrm{sp}}} Q^\mathsf{T}(t) \Sigma_\mathrm{Y}^{-1} Q(t) \right]^{-1}, \\
& Q(t) = \begin{pmatrix} \frac{\partial y_1}{\partial \theta_1} & \frac{\partial y_1}{\partial \theta_2} & \cdots & \frac{\partial y_1}{\partial \theta_n} \\ \frac{\partial y_2}{\partial \theta_1} & \frac{\partial y_2}{\partial \theta_2} & \cdots & \frac{\partial y_2}{\partial \theta_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial \theta_1} & \frac{\partial y_m}{\partial \theta_2} & \cdots & \frac{\partial y_m}{\partial \theta_n} \end{pmatrix}(t), \\
& \frac{\mathrm{d}\hat{\mathbf{y}}}{\mathrm{d}\theta} = \frac{\partial \mathbf{g}}{\partial \mathbf{x}} \cdot \frac{\partial \mathbf{x}}{\partial \theta}, \\
& \frac{\mathrm{d}}{\mathrm{d}t}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\theta} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \cdot \frac{\partial \mathbf{x}}{\partial \theta} + \frac{\partial \mathbf{f}}{\partial \theta}.
\end{aligned}
\tag{3.17}
$$

In Equation (3.17), $\mathbf{x} \in \mathbb{R}^{n_x}$ is the vector of differential states, $\hat{\mathbf{y}} \in \mathbb{R}^{n_y}$ is the vector of predictions for measured variables $\mathbf{y}(t))$, $\mathbf{u}(\cdot) \in \mathbb{R}^{n_u}$ is the vector of manipulated variables and $\theta \in \mathbb{R}^{n_\theta}$ is the vector of model parameters. We diverge again from methodology notation in that $\mathbf{u}$ and $\psi(\cdot)$ represent our black-box inputs and outputs, whereas $\mathbf{x}(t)$ and $\hat{\mathbf{y}}(t)$ denote our differential states and measured variable predictions. Σ_Y is the variance–covariance matrix of the experimental noise, Q is the local first-order sensitivity of the measured responses. The matrix V_θ is the approximated covariance matrix defined as the inverted expected Fisher information matrix (FIM). Additionally, $\psi(\cdot)$ is a metric of the covariance function, i.e. trace$(\cdot)$, det$(\cdot)$, etc.

Nucleophilic aromatic substitution reaction

We adapt our system of interest from Hone et al. [179], and consider a nucleophilic aromatic substitution (SNAr) of 2,4-difluoronitrobenzene {1} with pyrrolidine {2} in ethanol (EtOH) to

give a mixture of the desired ortho-substituted product {3}, with para- {4} and bis- {5} as side products. The three major products of this case study are relevant to the pharmaceuticals and fine chemicals industry [57]. The inlet flowrates of the reagents 2,4-difluoronitrobenzene {1} with pyrrolidine {2} (F_1 and F_2), solvent ethanol (F_3) and the temperature T are the decision (or design) variables: $\mathbf{u} = [T, F_1, F_2, F_3]^\intercal$ are bounded according to Table 3.2 and are subsequently normalized between 0 and 1.

Table 3.2: MBDoE case study: definition of design space.

Design Variable	Min	Max
F_1(mL/min)	0.3	3.0
F_2(mL/min)	0.3	3.0
F_3(mL/min)	0.3	3.0
$T(^oC)$	60.0	120.

The molar balance of the reactor can be written as:

$$\frac{dc_1}{d\tau} = -r_1 - r_2, \quad \frac{dc_2}{d\tau} = -r_1 - r_2 - r_3 - r_4, \\ \frac{dc_3}{d\tau} = r_1 + r_3, \quad \frac{dc_4}{d\tau} = r_2 - r_4, \quad \frac{dc_5}{d\tau} = r_3 + r_4, \tag{3.18}$$

with τ (min) being the residence time, and c_i (mol/L) the molar concentrations for compounds {1} to {5}. The total residence time is given as $\tau_{total} = \frac{2.7mL}{\sum_{i=1}^{5} F_i}$. The reaction rates follow a power law. The values for the kinetic parameters of the true model are taken from Hone et al. [179]. The objective of this case study is to minimize the maximum eigenvalue of the covariance matrix. MBDoE is performed from a starting point of the normalized design variables $\mathbf{u}_0 = [0.1, 0.1, 0.1, 0.1]^\intercal$ within bounds $[[0, 1], [0, 1], [0, 1], [0, 1]]$.

3.4.5 Self-optimizing reactions

Self-optimizing continuous flow platforms leverage automation, robotics, analytics, machine learning, and optimization to advance chemical synthesis and reaction optimization [343, 90, 332, 128]. Chemists often use model-driven approaches to reaction optimization, where models are used to map the effect of reaction parameters on the objectives. The parameters are then subsequently optimized using gradient expressions with respect to the reaction parameters. While the purpose of MBDoE (Section 3.4.4) is to design experiments that lead to the highest information gain in the process of model-building, self-optimizing reaction platforms skip the model-building step. The process parameters to objective input-output data are essentially obtained through iterative ex-

pensive black-box evaluations. Derivative-free and data-driven optimization is used to explore and exploit the search space and find the parameters that minimize a single or scalarization of multiple synthesis objectives (yield, selectivity, economics, safety, etc.).

Nucleophilic aromatic substitution reaction

We consider a similar Nucleophilic Aromatic Substitution (SnAr) reaction to the model-based design of experiments case study (Section 3.4.4). The mass and energy balances of the flow reactor are given by (3.19) followed by a detailed description of the parameters in Section A.4.

$$\begin{aligned} 0 &= D\frac{d^2c_i}{dV^2} - F\frac{dc_i}{dV} - \sum_{i'} \nu_{i',i} r_{i'} \\ 0 &= \kappa\frac{d^2T}{dV^2} - c_p\rho F\frac{dT}{dV} + UA(T_w - T), \end{aligned} \tag{3.19}$$

where c_i and T are the concentration of the ith reagent and the temperature of the reactor, respectively. T_w and F constitute the two manipulated decision variables and correspond to coolant temperature and inlet flow rate. The reaction rates $r_{i'}$ follow a power law with parameters given in Hone et al. [179]. The objective of this case study is to maximize the concentration of the Mono-product (c_3) at the end of the reactor while maintaining the concentration of the Bis-product (c_5) throughout the reactor below a threshold. The constraints are aggregated to accelerate the computation [305].

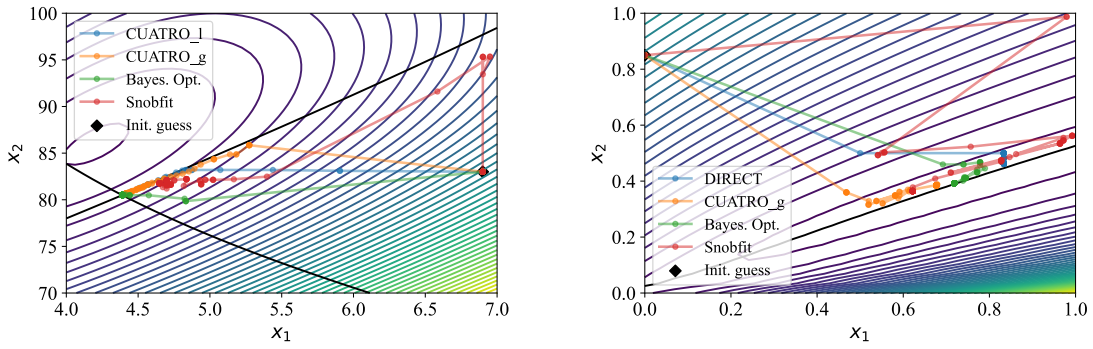
The manipulated variables $T_w(^oC)$ and $F(L/min)$ are bound within $[40, 150]$ and $[0.5, 10]$, respectively, and consequently normalized between $[[0, 1], [0, 1]]$ using a starting point of $[0.00, 0.85]^\intercal$. For the stochastic case, normally distributed noise with mean 0 and standard deviation 0.001 and 0.01 is added to the objective and constraints respectively. The convexity of the two-dimensional solution space is visualized in Figure (3.3b).

3.4.6 Controller synthesis

PI controller tuning

PI controllers are feedback control laws widely used in the control of industrial process systems. For a given process variable $x_i(t)$ and its setpoint $x_{i,ref}(t)$, PI control defines its error as: $e_i(t) = x_i(t) - x_{i,ref}(t)$, to get the control variable $u_i(t)$ as:

$$u_i(t) = K_{P,i}e_i(t) + K_{I,i}\int_0^t e_i(\tau)\,d\tau, \tag{3.20}$$



(a) (b)

Figure 3.3: Two-dimensional search space convergence plot for the best methods (3.3b) from the starting point towards the optimum for the a) real-time optimization case study, and the b) self-optimizing reactions case study.

where K_P and K_I are the proportional and integral controller gains respectively. Tuning of these parameters is crucial to ensuring stability of the state variables. This is usually achieved manually by a series of experiments, like subjecting the system to step changes, and heuristics, such as the Ziegler-Nichols, Cohen-Coon or Åström–Hägglund [426, 91, 427] methods. A more efficient way to tune the controller gains is to construct a process model and choose the gains based on the process parameters. More recently, computer-aided or computer-automated control system design have been used to aid or automate the process of controller tuning [233]. For instance, derivative-free methods, such as evolutionary search, have been used for this purpose [423, 215].

The performance of various DFO algorithms is first compared on the following two-state, two-control linear system:

$$\begin{aligned} \frac{dx_1}{dt} &= 2x_1 + x_2 + 2u_1 - u_2 \\ \frac{dx_2}{dt} &= x_1 + 2x_2 - 2u_1 + 2u_2. \end{aligned} \tag{3.21}$$

The integral squared error is discretized into the sum of n_T steps, and then used as the objective function of the black-box optimization problem:

$$\begin{aligned} \min_{K_{P,1},K_{I,1},K_{P,2},K_{I,2}} \quad & \sum_{k=0}^{n_T} (e_1(t_k)^2 + e_2(t_k)^2) \\ \text{s.t.} \quad & (3.21), \quad x_1(t_0) = x_2(t_0) = 15, \\ & (3.20), \quad i = 1, 2, \quad t = t_0, ..., t_{n_T}, \\ & e_1(t) = x_1(t) - x_{1,ref}, \quad e_2(t) = x_2(t) - x_{2,ref}, \\ & 0 \leq K_{P,1}, K_{I,1}, K_{P,2}, K_{I,2} \leq 8, \end{aligned} \tag{3.22}$$

where the four controller gains $K_{P,i}$ and $K_{I,i}$ denote the decision variables, and where $\mathbf{x}(t)$ and $\mathbf{u}(t)$ denote the state and control variables as per control literature conventions.

Synthesis of quadratic feedback regulator control laws

As opposed to the PI control case study, we examine a control problem without integral terms in the feedback law, or a fixed one-to-one mapping of state-to-control variables. The system under consideration is adopted from Kazantzis and Kravaris [204].

The nonlinear state feedback regulator synthesis method is applied to a continuous stirred tank reactor (CSTR), where the exothermic irreversible reaction between sodium thiosulfate and hydrogen peroxide is taking place in non-isothermal operation. This reaction exhibits steady-state multiplicity. Specifically, the system has three steady states. The upper and lower steady states are stable, while the middle one is unstable. Additionally, the linearized system around the steady states is controllable.

The control objective is to regulate the reactor at the middle, unstable steady-state by manipulating the dilution rate as a function of a bicubic polynomial in the outlet concentration and temperature error terms. Kazantzis and Kravaris (2000) [204] derive the polynomial coefficients, i.e. the controller gains, using a third-order Taylor series truncation. They show that this control law manages to overcome adverse conditions, where the system is perturbed to a temperature of 368.5 K and a concentration of component A of 0.116 $mols^{-1}$, and brought back to the unstable steady-state of 308.5 K and 0.666 $mols^{-1}$ rather than spontaneously driven to the closer equilibrium around 380.0 K. It is not possible to bring the system back to the equilibrium when including only terms up to the second order in the control law.

However, their derivation relies on the existence of a process model. Subsequently, the equivalent reactor is considered, where on top of the dilution rate, the inlet temperature can also be controlled on top of the dilution rate as an explicit control law with quadratic, bilinear, and linear terms in the outlet temperature and concentration error variables. The optimization problem takes

the following form:

$$
\begin{aligned}
&\min_{\substack{K_{C,i},K_{T,i}\\ i=1,\dots,5}} \quad \sum_{k=0}^{n_T}((\frac{e_1(t_k)}{c_{A,ss}})^2+((\frac{e_2(t_k)}{T_{in,ss}})^2)\\
&\text{s.t.} \quad \begin{cases} \frac{dC_A}{dt} = (\frac{F}{V}+u_1)(C_{A,in}-C_A)-2k(T)C_A^2, \\ \frac{dT}{dt} = (\frac{F}{V}+u_1)(T_{in}+u_2-T)+2\frac{(-\Delta H)_R}{\rho c_P}k(T)C_A^2-\frac{UA}{V\rho c_P}(T-T_j), \\ k(T)=k_0e^{-\frac{E}{RT}}\ , C_A(t_0)=0.116\frac{mol}{L}\ , T(t_0)=368.5K, \end{cases} \qquad (3.23)\\
&\qquad e_1(t)=C_A(t)-C_{A,ss}\ ;\ e_2(t)=T(t)-T_{in,ss} \quad t=t_0,\dots,t_{n_T},\\
&\qquad u_1(t)=K_{C,1}e_1(t)+K_{C,2}e_2(t)+K_{C,3}e_1(t)e_2(t)+K_{C,4}e_1(t)^2+K_{C,5}e_2(t)^2,\\
&\qquad u_2(t)=K_{T,1}e_1(t)+K_{T,2}e_2(t)+K_{T,3}e_1(t)e_2(t)+K_{T,4}e_1(t)^2+K_{T,5}e_2(t)^2,\\
&\qquad \mathbf{K}_C\in[\mathbf{K}_C^L,\mathbf{K}_\mathbf{C}^\mathbf{U}], \quad \mathbf{K}_T\in[\mathbf{K}_T^L,\mathbf{K}_T^U].
\end{aligned}
$$

Since the dynamics are only accessible through a simulation, this corresponds to a DFO problem with the ten controller coefficients as the decision variables (five dimensions in $\mathbf{K}_C$ and $\mathbf{K}_T$), and the discretized normalized squared error intregral in the two state variables as the output.

The initial guess for the first five quadratic coefficients ($\mathbf{K}_C$) is taken from the original derivation in Kazantzis and Kravaris [204]. The initial guess for $\mathbf{K}_T$ is taken as the null-vector of the same size. $\mathbf{K}_C$ has a lower bound corresponding to half, and an upper bound corresponding to double its initial guess. The bounds for $\mathbf{K}_T$ are constructed within a factor of ten around the initial guess for $\mathbf{K}_C$. Stochasticity is introduced in this case study by adding normally distributed noise with mean 0 and standard devation 0.001 and 1 to the molar fraction and temperature at each time step.

3.4.7 Flowsheet optimization

Flowsheet optimization problems often need to be approached as black-box optimization instances, since gradient expressions may be inaccessible due to proprietary restrictions or simply intractable to compute. In order to compare different DFO solvers in this context, we present an optimization problem relating to the CryoMan cascade liquefied natural gas refrigeration cycle [330, 10]. The objective of the problem is to minimize the specific shaft-work for compression whilst maintaining important process constraints such as minimum approach temperatures, vapour fractions on entry to compressors, and compression ratios. Due to the presence of multiple recycle loops, the process behaviour is extremely nonlinear, with many solutions violating process constraints.

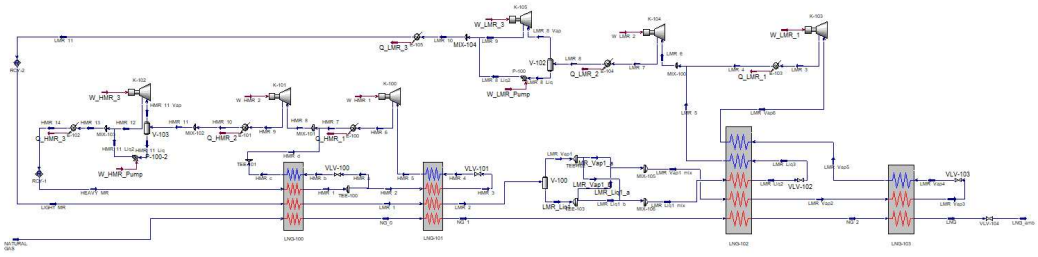


Figure 3.4: The CryoMan cascade refrigeration cycle as implemented in Aspen HYSYS [10].

The four design variables ϕ are chosen to be the heavy mixed refrigerant (HMR) flowrate, HMR 2nd stage compression ratio, light mixed refrigerant (LMR) flowrate, and LMR 2nd stage compression ratio. The mathematical description of the optimization is as follows:

$$\min_{\boldsymbol{\phi}} \quad \left(\sum_{i=1}^{N} W_i(\boldsymbol{\phi})\right)/m_{\text{LNG}} \tag{3.24}$$

$$\text{s.t.} \quad \Delta T^i_{\text{min}}(\boldsymbol{\phi}) \geq 2^o C \ , \ \Delta T^i_{\text{min}}(\boldsymbol{\phi}) \in \text{MSHE}_{\Delta T_{\text{min}}}, \tag{3.25}$$

$$\sum_{i=1}^{m} x_i(\boldsymbol{\phi}) = 1 \ , \ x_i(\boldsymbol{\phi}) \in [0,1], \quad i = 1, \ldots, m, \tag{3.26}$$

$$P_{\text{rat}}(\boldsymbol{\phi}) \leq 3.5, \tag{3.27}$$

$$\text{VF}^{\text{ref}}(\boldsymbol{\phi}) = 1.00, \tag{3.28}$$

$$\phi^L \leq \boldsymbol{\phi} \leq \phi^U. \tag{3.29}$$

The objective function consists of the specific shaft work, defined as the sum of the individual process compressor shaft works $W_i(\boldsymbol{\phi})$, divided by the mass flow rate of LNG m_{LNG}. The four decision variables are represented by $\boldsymbol{\phi}$ with the set of inputs having corresponding upper and lower bounds ϕ^L and ϕ^U (3.29). Constraints include: the minimum approach temperatures for each of the MSHEs $\Delta T^i_{\text{min}}(\boldsymbol{\phi})$ respectively where $\text{MSHE}_{\Delta T_{\text{min}}}$ is the set of all approach temperatures within the process ensuring feasible heat transfer between refrigerant and natural gas streams (3.25), maximum compression ratios $P_{\text{rat}}(\boldsymbol{\phi})$ to discourage mechanical damage to compressors (3.27), valid molar compositions represented by $x_i(\boldsymbol{\phi})$ (3.26), and no wetness within compressors to reduce mechanical damage to turbine blades, which is enforced by constraining vapour fractions at the inlet of compressors $\text{VF}^{\text{ref}}(\boldsymbol{\phi})$ to one (3.28). The flowsheet is implemented in Aspen HYSYS 8.8 and linked to Python using the COM interface. Deterministic flowsheet optimization is performed from a starting point of $\phi_0 = [0.5, 0.5, 0.36, 0.2]^\intercal$ within bounds $[[0,1],[0,1],[0,1],[0,1]]$, found during an initial random feasibility search.

In the following sections, empirical algorithm convergence and implementation considerations are discussed (Section 3.5.1). Then, the convergence of all methods is compared on the deterministic and noisy test function and chemical engineering-specific case studies before their sensitivity to increasing levels of noise is investigated (Sections 3.5.2 and 3.5.3). This is followed by a discussion on safe exploration and robust sample average approximation in Section 3.5.4.

3.5 Results and Discussion

3.5.1 Algorithms

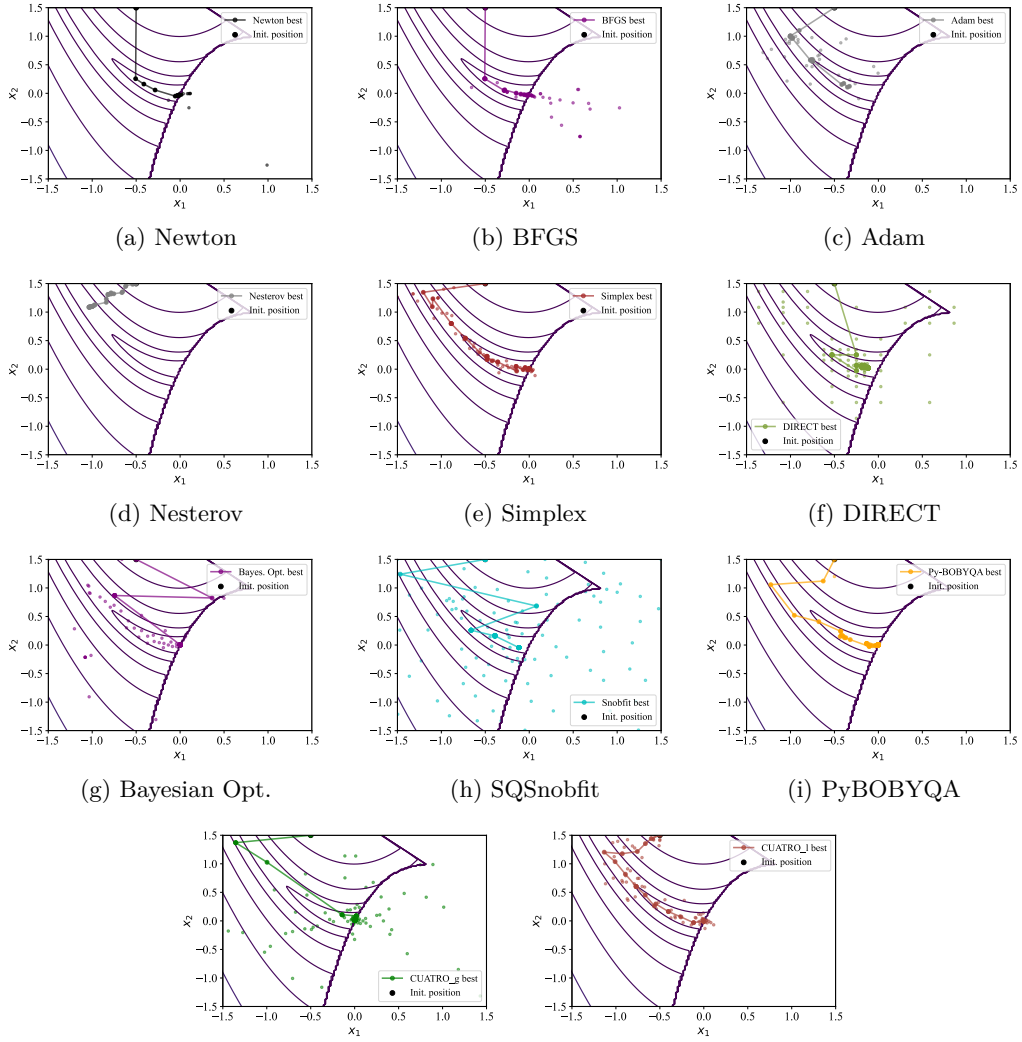
Sampling and Convergence

Figure 3.5 illustrates how each algorithm samples function evaluations and shows the convergence of the best function evaluations toward the optimum in the two-dimensional case of the constrained Rosenbrock function, discussed in more detail in Section 3.5.2. The figure illustrates how the finite difference algorithms, Nesterov, simplex, Py-BOBYQA and CUATRO local can be regarded as local methods, or 'exploitative', with a tendency to follow the gradient. Other methods like SQSnobfit, and CUATRO global involve more 'exploration', which in some cases leads to better optima in fewer function evaluations. Figure 3.5g illustrates the exploration-exploitation trade-off for Bayesian Optimization. Figure 3.5f shows a peculiar pattern for DIRECT, which starts off in the middle, with its exploitation following clear geometric patterns. The interested reader is referred to A.3 for an in-depth description of algorithm two-dimensional convergence patterns. Section 3.5.2 discusses the convergence of these algorithms in terms of number of function evaluations on specific case studies, and relates the methods' performance to the mathematical properties of the case studies.

Practical and numerical considerations

Before choosing an algorithm for a given application, one should keep in mind certain practical considerations which may hinder algorithm performance. These considerations are based on observed results rather than common knowledge. First, one should identify the hyperparameters that each algorithm has, and how sensitive performance is with respect to these.

- **Newton and BFGS**: General choices include the type of finite difference approximation (e.g.



(a) Newton (b) BFGS (c) Adam

(d) Nesterov (e) Simplex (f) DIRECT

(g) Bayesian Opt. (h) SQSnobfit (i) PyBOBYQA

(j) CUATROg (k) CUATROl

Figure 3.5: Two-dimensional solution space convergence on the deterministic constrained Rosenbrock function from starting point $[-0.5, 1.5]^\intercal$ for all algorithms.

central, forward), its derivative step sizes , the choice between fixed, adaptive, scheduled step size and/or backtracking. Standard finite difference approximations work quite well, but the algorithm is sensitive towards the decision to use backtracking or fixed step sizes, as well as the latter's magnitude. In this work, backtracking is used and the hyperparameters are tested and optimized for the smaller test functions, and slightly fine-tuned for each of the more complex case studies.

- **Adam**: Hyperparameters include the first- and second-order parameters β_1 and β_2, the step size parameter δ, and the backtracking routine. The default hyperparameters stated in the original technical report [212], used for machine learning applications such as training logistic regression and neural networks, are not transferable to DFO when used with the backtracking routine. Instead, we choose hyperparameters that involve less momentum, making the method behave more like gradient descent.
- **Simplex**: Works well with standard settings from the literature [276].
- **Nesterov**: Is the method that is the most sensitive to its hyperparameters (μ and α). These need to be fine-tuned to the specific application for the method to be successful. The method could also become more robust with the averaging of multiple randomly sampled gradients [32].
- **DIRECT**: Its main advantage is that it is an off-the-shelf algorithm. This usually comes at the cost of flexibility, but the implementation we use works very well with default parameters.
- **Bayesian Optimization**: Is probably the best choice when the computational budget is low (say < 30). The computational complexity scales rapidly with the number of function samples, so the assumption that function evaluations are the primary bottleneck may not hold if a large amount of data is considered per iteration, as this could render the algorithm intractable. The method is by far the slowest in terms of CPU time. On the positive side, it is arguably the best method for function evaluations that take hours or days (e.g. experiments or expensive simulations). We use an in-house implementation whose hyperparameters have been pre-tuned on similar case studies, which might skew the results in the algorithm's favor `https://github.com/UCL/Bayesian_opt_Pyro`.
- **SQSnobfit**: This is another off-the-shelf method with good performance using standard settings and minimal tuning. Additionally, the method is very flexible in allowing different modes of exploration, enabling more ways to leverage known information about the problem structure.

- **PyBOBYQA**: Works well off-the-shelf for deterministic problems with the flexibility to opt into bounds scaling, multiple restarts and a routine for handling noisy evaluations. The bounds scaling and multiple restarts options have been found to be especially useful, while the noise option does not dramatically improve performance.
- **CUATRO**: CUATRO's default hyperparameters work well off-the-shelf. These hyperparameters can also be tweaked to influence the degree of exploration. There are two versions of the algorithm following either a local or global search philosophy that decision-makers can choose based on problem characteristics and available budget. The solver is implemented in CVXPY [111]. While we use an early version of CUATRO in this work, the latest release can be found in `https://github.com/OptiMaL-PSE-Lab/cuatro` including the various extensions discussed in Chapter 6.

In the presence of constraints, algorithms that use the penalty method are very sensitive to the magnitude of the penalty parameter. For each case study, an initial parameter between 100 and 1,000 is picked depending on the magnitude of the objective and constraint evaluations. The parameters are increased until there are no constraint violations in the best solutions or until these are deemed acceptable (hard versus soft constraints). Choosing too high of a parameter can lead to ill-conditioning of the problem and numerical issues during surrogate fitting.

While it is assumed that computational time is limited by the number of function evaluations, as a rule of thumb, direct methods are significantly faster than model-based methods because they do not incur the cost of surrogate construction and optimization at each iteration. For instance, DIRECT takes milliseconds, CUATRO seconds, and Bayesian Optimization ten minutes to optimize the cheap Rosenbrock toy problem using an evaluation budget of 100. The rest of the methods' run times lie somewhere between that of DIRECT and CUATRO.

The code for the algorithm implementations and their comparisons on the case studies can be found in `https://github.com/OptiMaL-PSE-Lab/Expensive-Black-Box-Optim-ChemEng`.

Algorithm selection based on problem characteristics

While the previous section discusses each algorithm sensitivity to hyperparameter choices, this section discusses algorithm convergence based on problem characteristics.

As a rule of thumb, problem complexity increases significantly with the problem dimensionality. Purely bound-constrained problems are also easier to solve than their black-box constrained coun-

terparts. The difficulty of the problem also increases with the hardness of constraints. Additionally, complexity typically increases with the number of constraints, although a single complex constraint could potentially define the same feasible space as multiple simpler ones. The presence of stochasticity also considerably increases problem complexity, especially at higher amplitude. Lastly, the more 'well-behaved' the solution space is (with linear and convex being the most well-behaved), the easier the problem.

On top of the general observations, the following algorithm peculiarities are found. If any method is not explicitly mentioned , no clear dependence is observed for said characteristic.

- Problem dimensionality: Newton and BFGS scale particularly poorly due to their approximation of the Hessian (quadratic scaling). Adam scales better with problem dimensionality (linearly) than Newton and BFGS, but still performs worse compared with other methods because it requires over n_x evaluations at each iteration. Other methods only involve an expensive intitialization followed by fewer subsequent evaluations per iteration (CUATRO and simplex). Since our problem formulation assumes a tight evaluation budget, algorithms like BO, DIRECT and SQSnobfit that require fewer initial function evaluations seem to be less affected by the problem dimensionality.

- Constraints: As a rule of thumb, methods that only sample along the input dimensions (finite difference methods) rather than randomly tend to get trapped at constraints more often since they only capture information in a pre-defined set of directions. Simplex especially displays awkward behaviour near constraint boundaries. Bayesian Optimization deals best with constraints due to its explicit mapping thereof and its exploration routine. Accordingly, the performance of CUATRO along constraints improves with bigger trust region radii. The performance of other methods is very sensitive to the penalty parameter tuning.

- Stochasticity: Newton and BFGS break down in the presence of noise. Adam can leverage momentum to break through mild levels of stochasticity if fixed step sizes are used. Simplex and DIRECT seem to be little affected by stochasticity, while Nesterov's performance even sometimes improves with stochasticity. The model-based methods tend to perform well with stochasticity when they use regression instead of interpolation models. Consequently, SQSnobfit, CUATRO, and BO display less performance degradation than Py-BOBYQA in stochastic applications.

- Solution space topology: 'Global' model-based methods, such as BO and CUATRO global (or local with big trust region) display excellent performance in well-behaved problems, as

the solution space can be accurately captured by the surrogates. The performance of direct methods (DIRECT and simplex) depends on the specific solution space geometry. However, they tend to be more robust to ill-conditioning than model-based methods that rely on surrogate construction and minimization. The finite difference methods' performance, on a certain topology, is sensitive to the choice between backtracking and fixed step sizes, where backtracking (and less momentum for Adam) is usually preferred for ill-conditioned functions. If it is known that the problem is ill-conditioned, it is up to the decision-maker to decide if exploitative, local methods should be used to find a local optimum quickly, or if exploration should be leveraged in the hopes of escaping local minima.

The next section discusses algorithm convergence on specific case studies. Many subsequent observations might overlap with this section.

3.5.2 Performance of DFO algorithms on the case studies

Figures 3.6 to 3.14 show the best function evaluation versus evaluation budget convergence on the deterministic and stochastic versions of each case study, for each algorithm, using 50 to 100 function evaluations. In a nutshell, the following convergence performance is typically observed for the algorithms:

General algorithm performances: Nesterov makes consistent, but very slow progress. While some finite difference methods converge in the deterministic case for low-dimensional case studies, all of them break down with stochasticity. Adam is usually less affected by stochasticity, but tends to make slower progress in the deterministic case than its Newton and BFGS counterpart. The median simplex run usually converges to the optimum or to its vicinity, but runs tend to display much more variance in their performance, as they often get stuck at constraints. DIRECT often makes very fast initial progress due to its extensive search space exploration. However, the degree to which DIRECT is capable of fine-tuning its optimum is topology-dependent, on how close the initial partition-based sampling is to the actual optimum.

Similarly, BO and SQSnobfit also make significant progress early on due to their extensive exploration. BO is however better at navigating the exploration-exploitation trade-off in fine-tuning optima than SQSnobfit. SQSnobfit, due to its branching strategy, is less focused on exploitation resulting in a higher variation in its final performance. Py-BOBYQA usually displays excellent convergence for deterministic problems, but loses these properties on stochastic problems. This

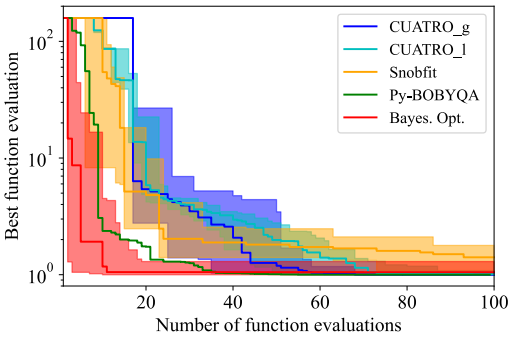
could be due to the method employing interpolation models which are more sensitive to noise, rather than regression models. Both CUATRO implementations tend to converge to the optimum for low-dimensional problems, while the local implementation performs better on higher-dimensional problems. Whether CUATRO local or global converges faster and more consistently depends on the problem topology, and on the initial trust region radius.

First, the algorithms are benchmarked on the constrained Rosenbrock test function, which was also used for hyperparameter tuning and to discuss convergence patterns (Section 3.5.1). Subsequently, the methods are benchmarked on specific chemical engineering case studies.

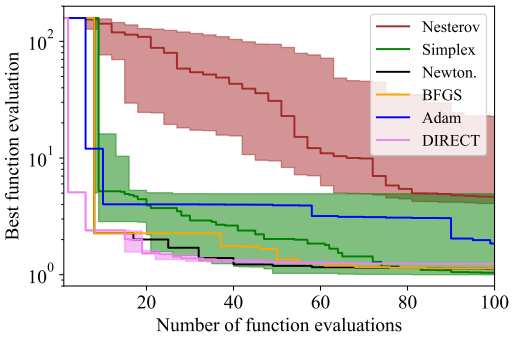
Rosenbrock: The problem is nonconvex. Its deterministic case shows the strengths of Newton and Adam when backtracking is used, while the stochastic case illustrates their degradation in performance with noise. The performance of Adam is slow, which is regularly observed for ill-conditioned problems. Simplex sometimes gets stuck at constraints increasing its min-max range. DIRECT and BO are quick because of their extensive initial exploration in the low-dimensional space. The performance of SQSnobfit is as expected: It converges to a near-optimum albeit with significant min-max range. Py-BOBYQA is excellent in the deterministic case, but becomes the worst model-based method in the presence of noise. CUATRO implementations have good convergence, with the global version being slightly faster. This suggests that in nonconvex lower-dimensional space, exploration leads to the optimum in fewer evaluations than with purely exploitative methods.

Real-Time Optimization: The discussion for the algorithm performances on this two-dimensional convex problem closely follows that for the Rosenbrock function. The main differences are as follows: Adam converges to the optimum in the deterministic case. CUATRO local converges faster than CUATRO global. CUATRO local and BO converge in around 15 evaluations, which is due to the problem being convex, making the problem easier to be approximated by the surrogates. DIRECT is also very quick to converge, finding a near-optimum in around 20 evaluations for the stochastic case. However it only finds a final median optimum of -60.6 compared with -76.0 for BO, which illustrates the dependence of DIRECT on the solution space topology. Figure 3.3a shows the two-dimensional convergence of the best methods on the case study. It illustrates the difficulty of even the best-performing methods to make progress along constraint boundaries, especially when these are almost parallel to the contour lines. As a result, BO is the only method whose median run converges to the optimum at the intersection of two constraints.

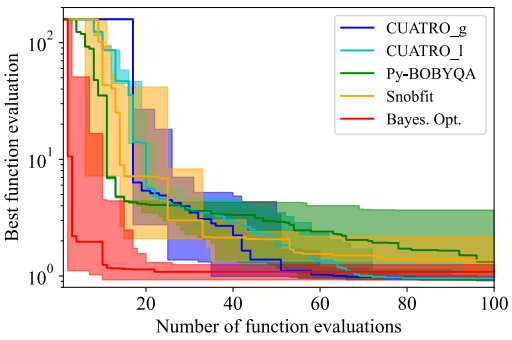
Self-optimizing reactors: This case study displays the same characteristics as the real-time



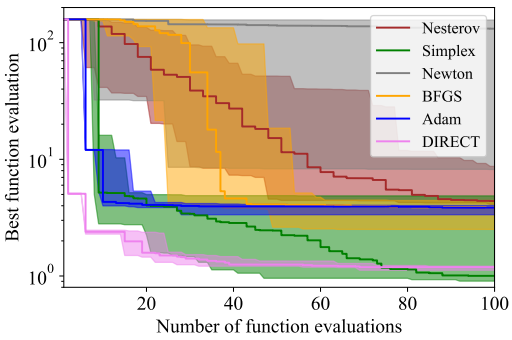
(a) Model-based, Deterministic



(b) Finite difference and direct, Deterministic

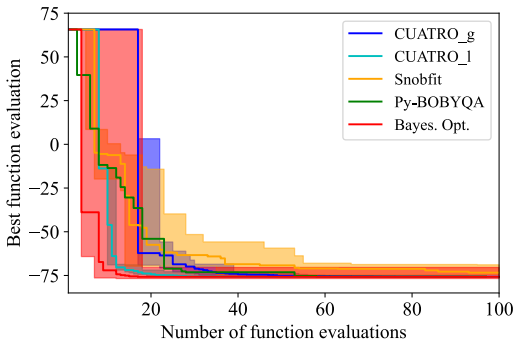


(c) Model-based, Stochastic

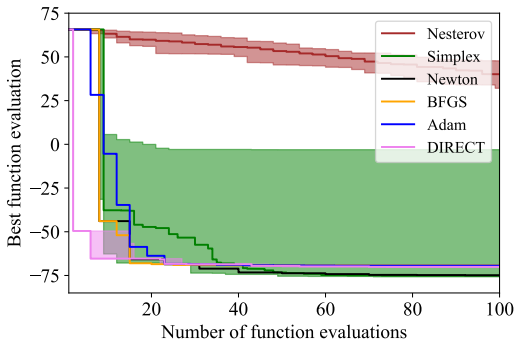


(d) Finite difference and direct, Stochastic

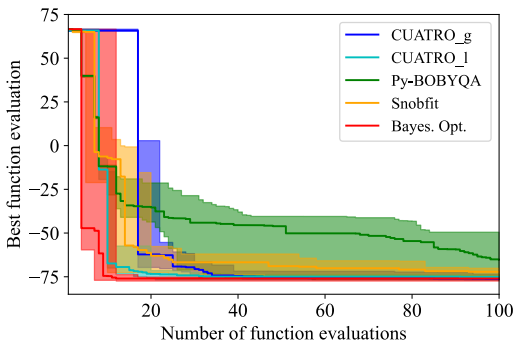
Figure 3.6: Convergence plots showing 10 best function evaluation versus number of function evaluation runs for all methods on the constrained Rosenbrock test function without and with stochasticity.



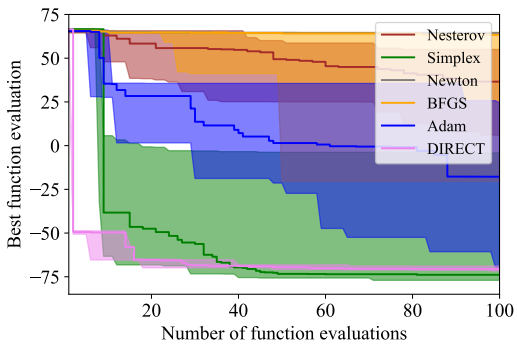
(a) Model-based, Deterministic



(b) Finite difference and direct, Deterministic



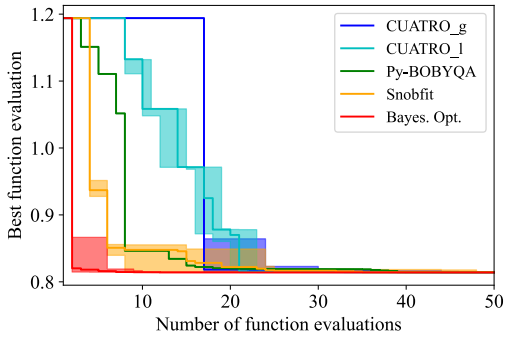
(c) Model-based, Stochastic



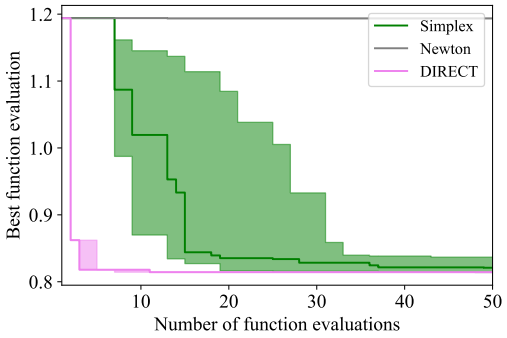
(d) Finite difference and direct, Stochastic

Figure 3.7: Convergence plots showing 10 best function evaluation versus number of function evaluation runs for all methods on the real-time optimization case study without and with stochasticity.

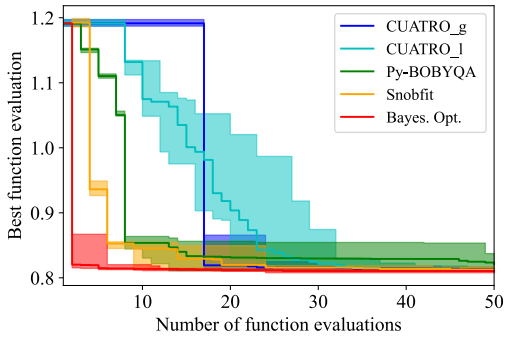
optimization case study, with similar results. DIRECT and BO are fast in finding the optimum in around 5 evaluations. In this well-behaved case study, the extensive initial search of CUATRO global (although more expensive initially) finds the optimum faster than the local implementation. Since self-optimizing reactions is the most expensive case study we investigate, and since we had initial numerical difficulties getting Nesterov, Adam and BFGS to work on the case study, we decided to omit them from the analysis. It is expected that their discussion would closely follow that of the real-time optimization case study. Figure 3.3b) shows the two-dimensional convergence of the best methods towards the optimum. BO is the only method that consistently cuts through the stochasticity and gets closest to the optimum near the constraints. The figure illustrates SQSnobfit's propensity towards exploration and that CUATRO global's first best evaluation update is already quite close to the optimum. It is also observed that DIRECT's first function update is set to the middle of the decision space, which is due to the method discarding initial points and just starting in the middle of the decision space.



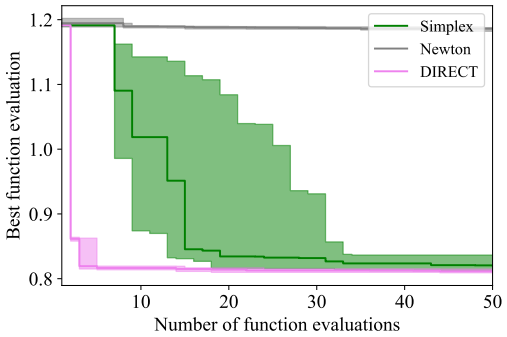
(a) Model-based, Deterministic



(b) Finite difference and direct, Deterministic



(c) Model-based, Stochastic

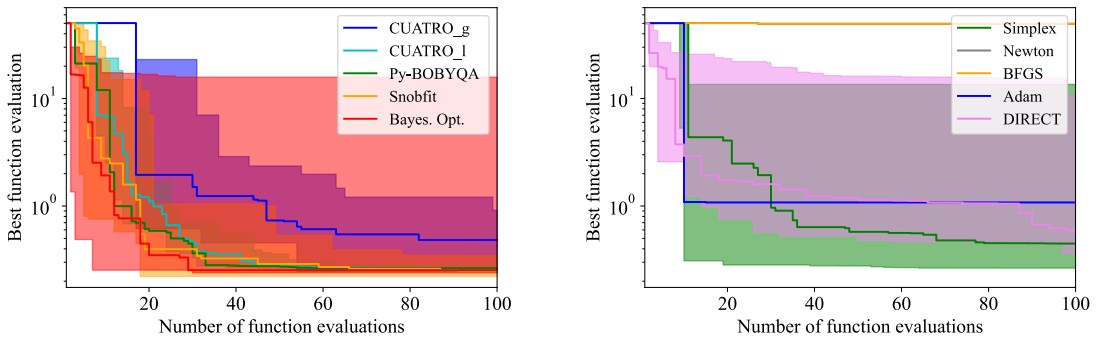


(d) Finite difference and direct, Stochastic

Figure 3.8: Convergence plots showing 10 best function evaluation versus number of function evaluations runs for all methods on the self-optimizing reactions case study without and with stochasticity

Model-based Design of Experiments: This case study differs from others in multiple ways:

First, this is the first case study to have more than two decision variables, making the analysis *a posteriori* of the optimization difficult due to a lack of visualisation. Additionally, the best optimum identified is active at the lower input bound in two out of four dimensions, whereas in previous case studies the optimum is active at one or two black-box constraints. DIRECT, as a partition-based method is slow with a big min-max range, converging to a median sub-optimum of 0.578 compared with the optimum at 0.231. Additionally, the problem is ill-conditioned, which can lead to problems with model-based methods. CUATRO local with small trust region radius performs better than its global implementation, potentially for similar reasons. This could also explain why BO fails to find the optimum consistently. While its median run finds the best optimum of all methods at 0.253, three out of ten runs get stuck early between evaluations of 15.5 and 16. Py-BOBYQA converges quickly to the optimum given the deterministic nature of the problem. Nesterov is left out of the analysis because no adequate hyperparameters could be found. Adam is the only finite difference method that makes significant progress given the poor Hessian approximation scaling of Newton and BFGS. Simplex does not reliably converge to the optimum and displays a lot of variance because some of its runs get trapped at bound-constraints.



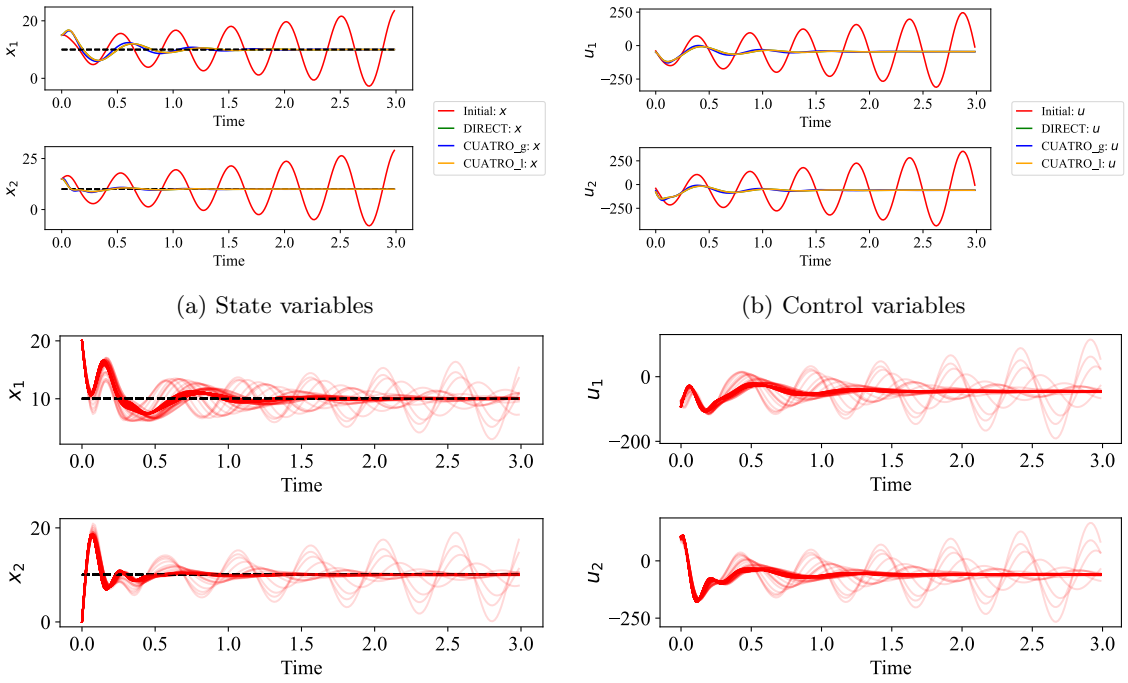
(a) Model-based

(b) Finite difference and direct

Figure 3.9: Convergence plots showing 10 best function evaluation versus number of function evaluations runs for all methods on the deterministic model-based design of experiments case study

PI controller tuning: Providing more background on the mathematical properties of the case study enhances our understanding of the results. The decision space is not only four-dimensional, but the objective function is also particularly ill-behaved. Figures (3.10a) and (3.10b) show the state and control trajectories resulting from the initial PI controller tuning and the median final optimal DIRECT, and CUATRO tunings respectively. While the initial controller tuning leads to a divergent response, the optima manage to bring back the system to the steady-state. The explosion in objective values between converging and diverging tunings causes ill-conditioning in the problem. While Figures (3.10a) and (3.10b) illustrate the difference between the initial and

final controller tuning responses, Figures (3.10c) and (3.10d) showcase the evolution of state and control trajectories corresponding to all evaluated CUATRO local PI controller tunings from the oscillating, diverging response to the near-optimal converging response. Experiments are performed on the same dynamic system using different initial states between the top and bottom figures in (3.10).



(a) State variables

(b) Control variables

(c) State variables

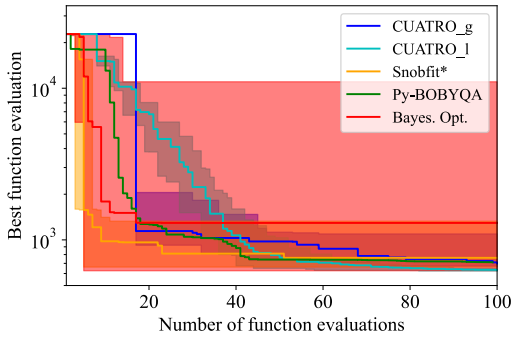
(d) Control variables

Figure 3.10: State (3.10a) and control (3.10b) trajectories corresponding to the initial and the median optimal DIRECT, CUATRO global and CUATRO local PI controller gains for the deterministic PI controller tuning case study. Evolution of the state (3.10c) and control (3.10d) trajectories corresponding to all sampled median CUATRO local PI controller tunings from the initial towards the best.

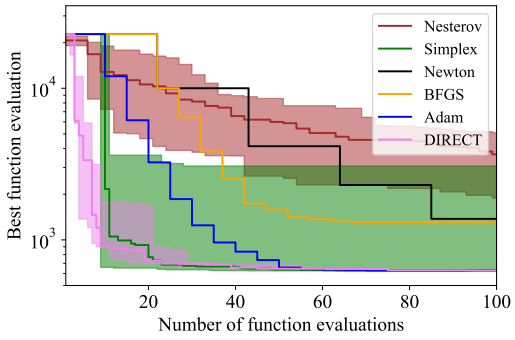
Additionally, stochasticity is introduced in a different way in this case study. Whereas in previous case studies we add randomly sampled noise to the objective and constraint functions, here, we add randomly sampled noise to the state variables to imitate measurement fluctuations due to process disturbances. This makes the numerical integration sensitive to high levels of stochasticity, and the whole problem ill-conditioned.

The performance of many algorithms is as expected: Nesterov makes slow, consistent progress. Adam makes surprising progress compared with Newton and BFGS in the deterministic case, but all methods lose efficiency in the stochastic case. The median runs of simplex and DIRECT approach a near-optimum in as little as 20 evaluations. While DIRECT is slightly worse than simplex at fine-tuning the optimum, simplex displays worse variance due to the method getting

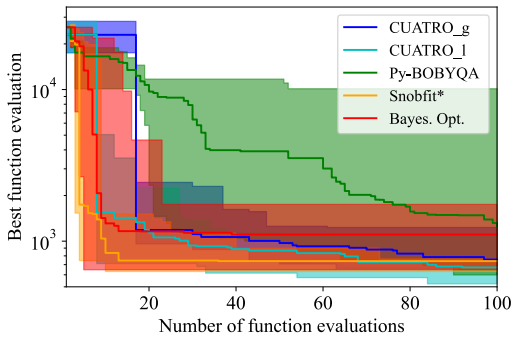
stuck at some input-bounds. SQSnobfit finds good optima but the algorithm returns an error in about 10% of the deterministic and 30% of the stochastic cases, making the method unreliable. This can again be explained by numerical issues causing ill-conditioning. The discussion for Py-BOBYQA follows that of the other case studies. BO displays a min-max range as high as 1,099, due to the occasional run getting trapped. The median run is outshone by all other model-based methods apart from Py-BOBYQA. The discussion boils down to ill-conditioning again. Hence, as with other highly ill-behaved functions, global surrogate construction should be discouraged. The discussion surrounding CUATRO follows that of previous case studies in terms of trust region radii, ill-conditioning, and the exploration-exploitation trade-off.



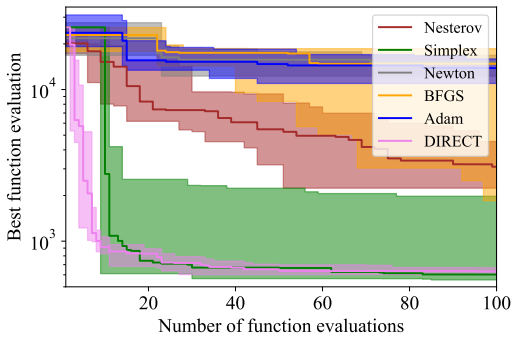
(a) Model-based, Deterministic



(b) Finite difference and direct, Deterministic



(c) Model-based, Stochastic



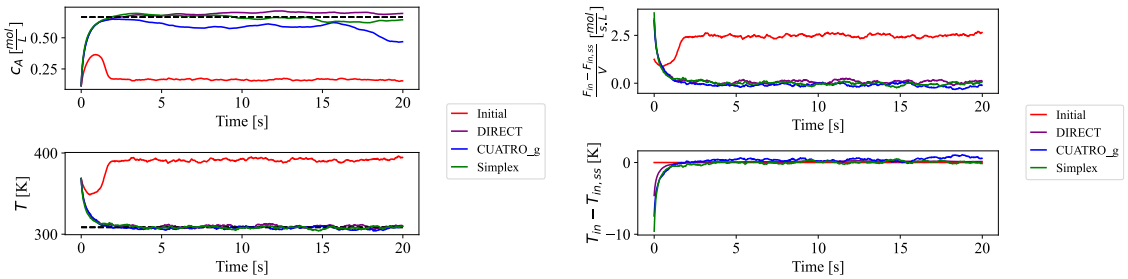
(d) Finite difference and direct, Stochastic

Figure 3.11: Convergence plots showing 10 best function evaluation versus number of function evaluation runs for all methods on the PI controller tuning case study without and with stochasticity. The asterisk in the label for Snobfit denotes that not all of its 10 runs were successful.

Quadratic control law synthesis: This case study involves ten decision variables. As a result, the finite difference approximation methods make little progress. The direct methods (DIRECT and simplex) converge reliably to a near-optimum in 20 evaluations. The discussion surrounding the model-based methods mirrors that of the previous PI tuning case study: BO 's median run is outshone by all other methods. SQSnobfit seems to find good solutions, but returns

an error in over half of the cases. Py-BOBYQA performs poorly in the presence of stochasticity. One might expect CUATRO local to outperform CUATRO global, like it did in the stochastic PI controller tuning case in Figures 3.11. One might expect consistent, smaller steps to be more important than extensive exploration for ill-behaved functions. However, both CUATRO variations exhibit similar performance after 20 evaluations.

This could be explained by significant differences in the scaling and bounds of the decision variables. Additionally, the two control problems exhibit distinct control dynamics and 'runaway' behaviors. Figure 3.12 shows the state and control trajectories resulting from the initial and the median best DIRECT, CUATRO global and simplex tunings. The initial coefficients fail to overcome the attraction of the dynamical system towards one of the stable optima. The near-optimal tunings manage to drive the system back to the meta-stable reference points, with the differences between the methods being mostly attributable to the stochasticity in the state variables rather than the quality of the optima. Similarly to the PI tuning solution space, this case study also displays ill-conditioning. Even though none of the control laws lead to diverging solutions, small differences in tunings can lead to the system converging to different optima, which can increase the objective function by orders of magnitude.

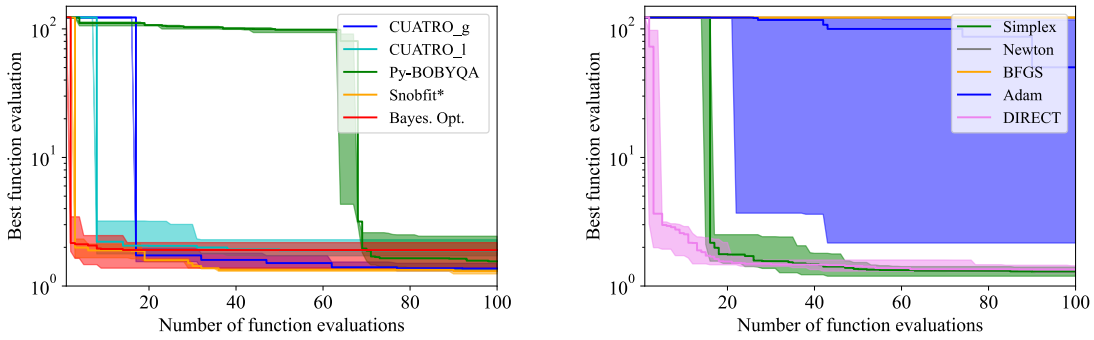


(a) State variables

(b) Control variables

Figure 3.12: State and control trajectories corresponding to the initial and best DIRECT, CUATRO global, and CUATRO local controller gains for the deterministic quadratic feedback regulator case study.

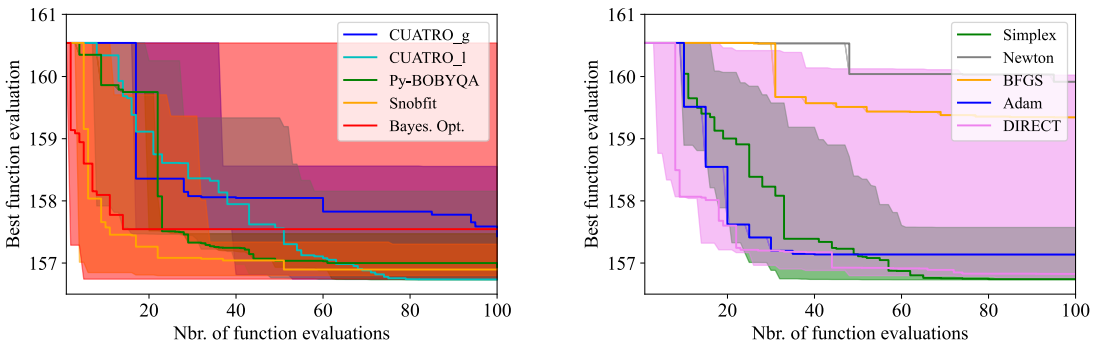
Flowsheet Optimization: The finite difference approximation methods perform as expected on the case study given its four-dimensional, deterministic nature. While both DIRECT and simplex find competitive median optima, some realizations of DIRECT stall, increasing its min-max range considerably. Py-BOBYQA and SQSnobfit behave as expected. CUATRO global performs poorly and displays a lot of variance with a range of 1.82 and a final median realization at 157.43, far from the optimum. CUATRO local fine-tunes the optimum consistently giving a median optimum of 156.74. However, some realizations become trapped at a constraint leading to a high min-max range of 1.42. BO, like in previous case studies with more than two decision



(a) Model-based, Stochastic (b) Finite difference and direct, Stochastic

Figure 3.13: Convergence plots showing 10 best function evaluation versus number of function evaluations runs for all methods on the stochastic quadratic control law synthesis case study. The asterisk in the label for Snobfit denotes that not all of its 10 runs were successful.

variables, displays significant variance, ranging from the optimum to the starting point, with the median final realization being close to that of CUATRO global. The difficulties for the MBDoE case study reside in its ill-conditioned objective changing over an order of magnitude, whereas the difficulties for flowsheeting arise from its 12 constraints. These either need to be modelled separately or can lead to ill-conditioning when using penalty methods.



(a) Model-based (b) Finite difference and direct

Figure 3.14: Convergence plots showing the 10 best function evaluation versus number of function evaluation runs for all methods on the deterministic flowsheeting case study

3.5.3 Performance of DFO algorithms at increasing levels of stochasticity

We are interested in observing the resilience against increasing levels of stochasticity of the best-performing methods as identified in the previous section. To this effect, we perform 20 runs of 50 to 100 evaluations at increasing levels of noise for the best algorithms of each stochastic case study.

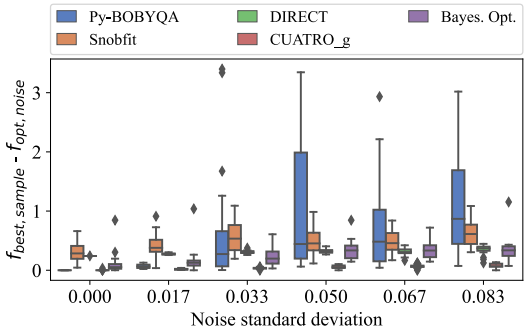
Stochasticity levels are defined by changing the standard deviation of the normally distributed noise, added to the case studies as discussed in Section 3.4. Boxplots show the difference between each sample optimum and the best found optimum at the given noise level among all considered methods (estimated optimality gap) in Figure 3.15.

Constrained Rosenbrock with stochasticity: The best methods identified in the previous section include: Py-BOBYQA, DIRECT, BO, SQSnobfit, and CUATRO global. Py-BOBYQA, though really good in finding the optimum in the deterministic case, becomes less reliable with stochasticity. SQSnobfit keeps a consistent min-max range around 0.75 despite only finding a near-optimal solution between 0.5 and 0.75 for all levels of stochasticity. DIRECT shows consistent, good performance at finding near-optimal solutions (optimality gap below 0.5 and similar min-max range) for all levels of uncertainty. BO displays a similar performance as DIRECT in terms of the quality of the optimum found, but has a higher min-max range closer to 1 for most stochasticity levels. CUATRO global stands out as the method finding the best optimum at each stochastic level with the highest consistency (highest min-max range of 0.13).

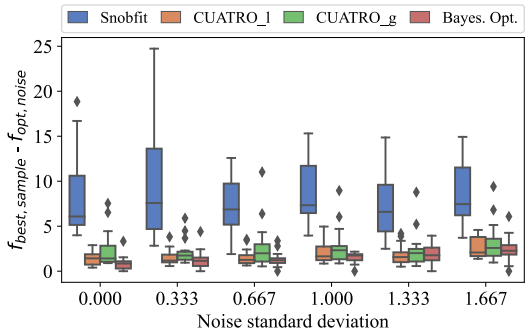
RTO in the presence of process disturbances: The best methods identified in the previous section include: SQSnobfit, CUATRO local, CUATRO global, and BO. SQSnobfit presents a median optimality gap between 5 and 7 for all noise levels while its min-max range remains mostly unaffected. This highlights the method's difficulty in fine-tuning optima. CUATRO methods display similar performance again, consistently finding good median optima with an optimality gap around 2. The global version displays higher min-max ranges up to 10, compared with a range of up to 5 for its local counterpart. Both methods are slightly affected by higher stochasticity levels. BO consistently finds the best optimum among all methods, while displaying a slightly higher min-max range than CUATRO local for all but one level.

Self-optimizing reactions in the presence of process disturbances: The best methods identified in the previous section include: SQSnobfit, DIRECT, CUATRO global, and BO. SQSnobfit is the method that finds the worst optimality gaps of all methods with the highest min-max ranges. BO finds the best median optima for all noise levels, with min-max ranges up to 0.01. This does not mean that it always finds the best sample given the variance in DIRECT and SQSnobfit at the two highest noise levels. DIRECT's median performance closely follows BO and displays slightly less variance than BO at most levels. CUATRO global displays median optima and ranges between that of SQSnobfit and DIRECT.

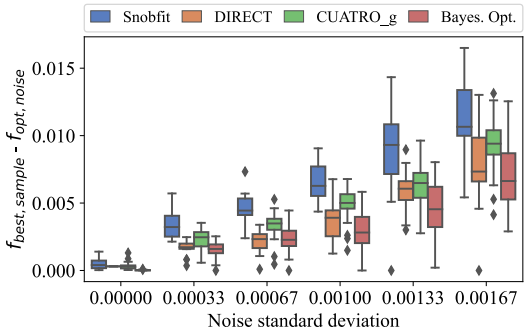
PI controller tuning in the presence of process disturbances: The best methods iden-



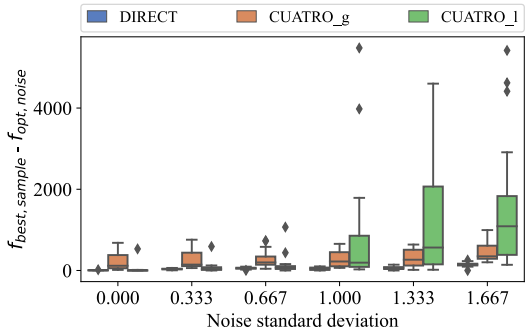
(a) Constrained Rosenbrock



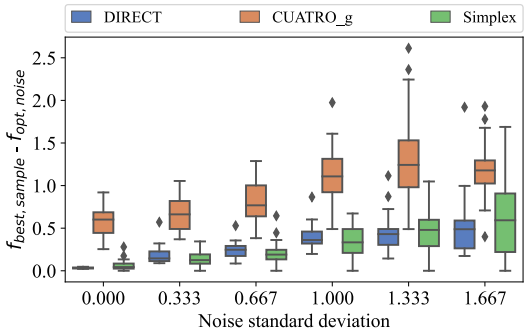
(b) Real-Time Optimization



(c) Self-optimizing reactions



(d) PID controller tuning



(e) Quadratic Control Law

Figure 3.15: Stochasticity convergence plots for the best methods per noisy case study: For each of the six stochasticity levels, the difference is shown between the best optimum found in a) 100, b) 50, c) 50, d) 100, or e) 100 function evaluations for each of the 20 runs and the best optimum found among all methods for said stochasticity level. The stochasticity levels refer to the noise standard deviations of the normally distributed noise added to each stochastic case study as discussed in Section 3.4

tified in the previous section include: DIRECT, CUATRO global, and CUATRO local. DIRECT stands out as the method to always find the best optimum at all stochasticity levels, and displays little variance in the optimum found. CUATRO global's median optimality gap closely follows that of DIRECT. CUATRO global also displays a worst-case optimality gap of 1,402, compared with DIRECT's worst range of 256 at the highest noise level. CUATRO local becomes less reliable with increasing levels of stochasticity with some realizations resulting in optimality gaps of 5,419 at the highest noise level. This suggests that extensive initial exploration (e.g. the global method) allows the method to make progress when high levels of stochasticity obscure a clear correlation for the surrogate to capture in the local trust region. On close inspection of the optima found, it is seen that three out of the four decision variables are at their upper bound. This suggests that DIRECT is not only useful for problem formulations where the optimum is close to the middle of the bound-constrained region. It also efficiently exploits more remote regions of the decision space.

Quadratic control law synthesis in the presence of process disturbances: The best methods identified in the previous section include: DIRECT, CUATRO global, and simplex. CUATRO global performs worse than the two other methods and displays more variance in the best optimum found with median optimality gaps up to 1.25 and min-max ranges up to 2.1. DIRECT is more consistent (min-max range up to 0.75) and displays similar median optima to the simplex method despite rarely finding the best optimum. Simplex displays similar median final realizations to DIRECT, but due to its higher variance (ranges up to 1.5) it has a higher chance of finding better optima. It is surprising to see that the two direct methods DIRECT and simplex deal better with this problem than the model-based ones. At this stage it is unclear if it is due to the higher dimensionality and/or the conditioning of the problem.

3.5.4 Constraint Satisfaction

In our last analysis, we investigate the constraint satisfaction versus convergence trade-off for all algorithms on the constrained, deterministic case studies, as well as the effect of sample average approximation on constraint satisfaction on the stochastic Rosenbrock test function.

The full treatment of constraint satisfaction and safe exploration under uncertainty in DFO algorithms is out of scope, and would require a full research contribution. There are specialized techniques and subroutines that can be added to make exploration inherently safer or to guarantee constraint satisfaction with high probability via chance constraints for example [362, 174, 55, 298].

Safe exploration

In Table 3.3, we report each algorithm's average percentage of infeasible samples per run on the deterministic constrained case studies alongside their average best function evaluation. The emphasis of this analysis is on model-based methods, and their ability to accurately synthesize constraints without having to violate them, but we also discuss direct methods for completeness.

Self-optimizing reactor: We see that all methods apart from Newton achieve a similar objective function (between 0.814 and 0.823). Among the direct DFO methods, DIRECT displays the worst constraint satisfaction (68.8%), mostly because it gets closer to the optimum, which is active at the constraint, in fewer evaluations than simplex (5 versus 30) as seen in Figure 3.8. This simple correlation does not hold for all model-based methods. Py-BOBYQA displays the best constraint satisfaction (90.0%) despite its fast convergence. This might be indicative of a well-tuned penalty parameter. BO also displays good constraint satisfaction (85.9%) despite the quality of the solution and the speed of convergence. This could be explained by the method's exploration strategy and its explicit handling of constraints. Despite explicitly handling constraints using quadratic surrogates, neither CUATRO local nor CUATRO global have superior constraint satisfaction properties (70.1% and 76.3%).

Flowsheeting: The same general trend is seen, namely methods converging faster, or with higher accuracy, sample proportionally more constraint-violating points. On top of this, exploratory methods such as CUATRO global and SQSnobfit have worse constraint satisfaction (39.4% and 34.0%) than similarly performing methods focused more on exploitation (CUATRO local at 63.2%).

Real-time optimization: Among the competitive methods, BO (64.0%) outperforms CUATRO in terms of constraint satisfaction (52.6% and 50.9% for global and local). This is again due to BO's exploration-exploitation routine, and its use of constraint surrogates in determining subsequent samples. Penalty methods fair particularly poorly on this problem. The poor constraint satisfaction properties of Py-BOBYQA, simplex and Adam (21.6%, 29.6% and 39.2% compared with the next-best of over 50%) are likely caused by a penalty coefficient that is too small. Essentially, objective improvements outweigh constraint violations at the boundary. Solution updates become infeasible early in the run and approach the optimum via the infeasible space. Increasing the penalty parameter, while improving constraint satisfaction, can lead to ill-conditioning and numerical challenges in model-based methods. Adaptive penalty strategies may alleviate this issue, but it can be more effective to avoid these problems altogether by using explicit constraint-handling

techniques.

Table 3.3: Average convergence-safe exploration trade-off on deterministic, constrained case studies

Constraint satisfaction ‖ Best optimum			
Algorithm	Real-time optimization	Self-optimizing reactions	Flowsheeting
Adam	39.2% ‖ -69.369	NA	100% ‖ 157.14
Bayes. Opt.	64.0% ‖ -75.436	85.9% ‖ 0.81398	81.3% ‖ 157.75
BFGS	NA	NA	81.0% ‖ 159.34
CUATRO_g	52.6% ‖ -75.744	70.1% ‖ 0.81422	39.4% ‖ 157.44
CUATRO_l	50.9 % ‖ -75.820	76.3% ‖ 0.81410	63.2% ‖ 156.93
DIRECT	55.1% ‖ -70.055	68.8% ‖ 0.81426	63.2% ‖ 157.96
Newton	72.2% ‖ -74.908	100 % ‖ 1.19359	100% ‖ 159.91
Nesterov	100 % ‖ 39.627	NA	NA
Py-BOBYQA	21.6 % ‖ -75.650	90.0% ‖ 0.81430	60.0% ‖ 157.47
Simplex	29.6 % ‖ -68.004	74.6% ‖ 0.82360	87.9% ‖ 156.91
SQSnobfit	53.5 % ‖ -71.874	76.5% ‖ 0.81430	34.0% ‖ 156.97

Robustness through Sample Average Approximation

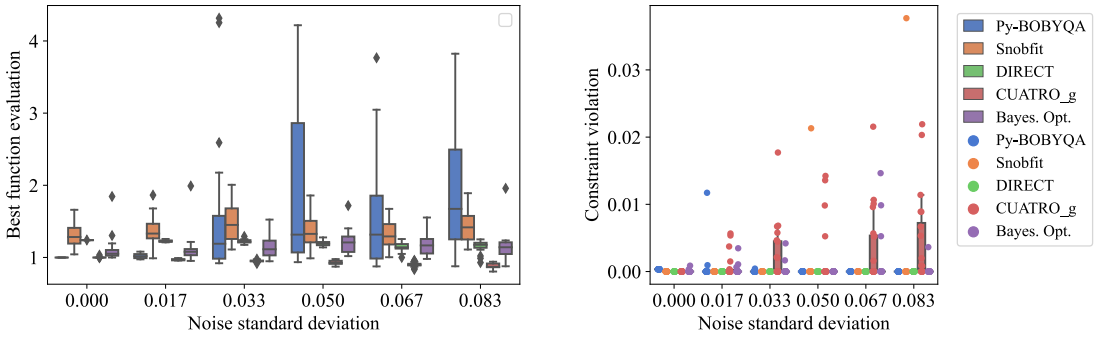
We investigate Sample Average Approximation (SAA) as a simple subroutine to mitigate constraint violations in stochastic problems. In theory, one could recover the 'true' objective and constraint evaluations in noisy problems by sampling new points infinitely many times [178]. In practice, decision-makers could perform multiple evaluations at the same point and use the average or a specific quantile thereof, depending on their risk tolerance. However, this comes at the price of additional expensive function evaluations, which are assumed to be the bottleneck.

In Figure 3.16, we investigate, for the best-performing methods on the constrained Rosenbrock case study, whether halving the evaluation budget and employing the sample average approximation (SAA) —the mean — of two sampled objectives per evaluation, is more effective than the default strategy of only sampling new evaluations once. Additionally, we employ a robust approach to constraint handling by only considering the 'most-violating' constraint evaluation among the two samples. This method aims to handle constraints conservatively rather than merely satisfy them in expectation.

Figure 3.16a is identical to 3.15a, but instead of showing the difference between the best evaluation of each run and the overall best found across all methods at a given stochasticity level, it only displays the function value of the best optimum identified after 100 evaluations for each method. Figure 3.16b shows how many final best function evaluations of the 20 runs for each noise level and algorithm are constraint violating in expectation. This explains why seemingly better optima are found with increasing stochasticity: A sample that is infeasible in expectation might

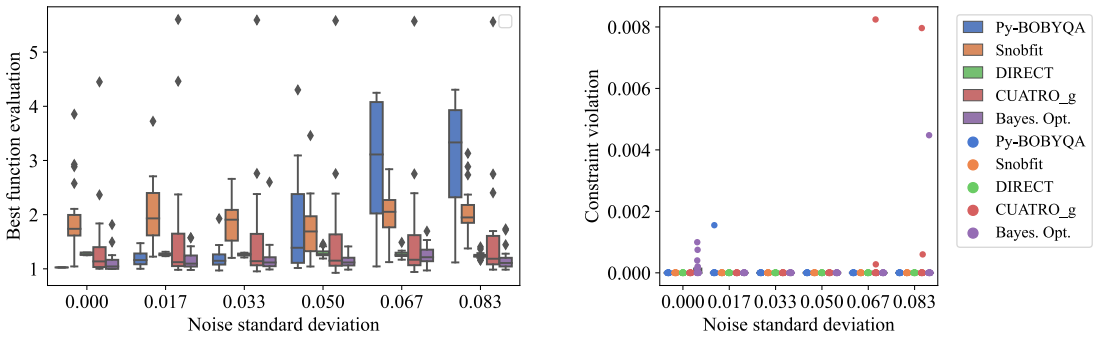
be accepted, returning optima that are in principle impossible to achieve in the expected feasible space. Escaping the infeasible space is also challenging, because any move back to the feasible space probably results in a degradation of observed solution quality.

Figure 3.16c shows a similar plot to Figure 3.16a. The main takeaway is that, at a fixed number of total evaluations, using SAA for more accurate information about the objective does not generally improve algorithm convergence given the effective halving of the evaluation budget. However, the robust constraint handling in our SAA approach does improve constraint satisfaction as evident in comparing Figures 3.16d and 3.16b. The occasional infeasible Py-BOBYQA and BO optima could be explained by weak penalty coefficients and algorithm-specific constraint tolerances respectively, or simply two successive optimistic constraint evaluations. As to CUATRO global, the occasional best evaluation is infeasible only at high stochasticity (around 3.33 % constraint satisfaction compared with the previous 17.5 %) but to a lesser extent: maximum constraint violation is decreased from 0.008 to 0.0038.



(a) Convergence, 100 SAA1 evaluations

(b) Constraints, 100 SAA1 evaluations



(c) Convergence, 50 SAA2 evaluations

(d) Constraints, 50 SAA2 evaluations

Figure 3.16: Constrained Rosenbrock convergence and proportion of constraint violation plots for PyBOBYQA, SQSnobfit, DIRECT, CUATROg and Bayesian opt. without and with sample average approximation at the same total evaluation budget.

3.6 Conclusion

Eight model-based and direct DFO algorithms are compared alongside three finite difference approximation implementations of popular gradient-based optimization algorithms on one constrained test function and five chemical engineering DFO problems. The convergence of the best function evaluation found with respect to the number of function evaluations is benchmarked for deterministic and potentially stochastic variations of the case studies. This is followed by a discussion of the best methods' ability to withstand increasing levels of stochasticity. Their convergence is also illustrated in the two-dimensional solution space whenever possible. As expected, the performance of the algorithms is widely dependent on problem-specific considerations such as conditioning of the black-box function, the presence of constraints, and problem dimensionality. While no method performs best in all scenarios, we can still give general recommendations.

Finite difference approximation methods are generally not recommended if the function evaluations are noisy, or if the number of decision variables is high. Among the finite difference methods, Adam has the most potential to deal with ill-behaved and noisy functions but is very sensitive to hyperparameter tuning.

Nesterov displays the worst performance out of all investigated methods. This is unsurprising as it is regarded as an all-purpose method, which cannot be expected to perform well for all subclasses of problems. Nesterov might shine for applications that are even noisier and less well-behaved than the considered applications.

Bayesian optimization and CUATRO global or local with big initial trust regions, should be consistent, resilient against stochasticity, and able to fine-tune optima if a function is believed to be smooth and well-behaved. BO seems to generally make astonishing progress in few function evaluations. However, Bayesian optimization has the worst run time of all algorithms. As a rule of thumb, Bayesian optimization should be used with a low function evaluation budget (around 30), and only if the function evaluations are more expensive than the per-iteration computational algorithm overhead. Expensive evaluations should take at least ten seconds. It might be the case that even for very expensive functions, Bayesian optimization becomes intractable. In Chapter 6 we discuss specialized high-dimensional BO implementations, relying for example on Thompson sampling [127] or sparse Gaussian processes.

SQSnobfit is recommended if extensive exploration is desired, or if no prior knowledge is available about a noisy function. However, SQSnobfit is not great at fine-tuning.

CUATRO's global variation, with its initial extensive exploration can be beneficial in smooth and well-behaved solution spaces. On the flip side, CUATRO global can stall in ill-conditioned problems or when the trust region is too big. If exploration is less of a concern and the aim is to make slow but steady progress, we recommend CUATRO local with a smaller trust region. This setup works particularly well in combination with quadratic discrimination constraint handling when dealing with simulation queries that might fail and cause ill-conditioning.

DIRECT would be our recommended 'all-purpose' solver if no prior knowledge is available. The partitioning strategy encourages consistent progress despite extensive exploration, while the biasing towards promising solutions ensures exploitation. Its only apparent drawback is its systematic solution space exploration, which can get in the way of fine-tuning when quick convergence to good optima is preferred. Since DIRECT and Bayesian optimization shine in making significant progress in early function evaluations, there is potential in combining them with a method focused on greedy optimization.

Finally, although most of the previous algorithms rely on the penalty method to address constraints, there is significant value in further developing explicit constraint-handling methods like CUATRO or Bayesian Optimization. These methods circumvent the challenge of penalty parameter tuning and facilitate safer exploration, particularly when minimizing the number and severity of constraint-violating queries is crucial. In the presence of noisy constraints, sample average approximation can be used in conjunction with conservative constraint handling to promote constraint satisfaction at the potential expense of solution quality.

3.6.1 Looking Back

With the benefit of hindsight, there are always areas for improvement. Many strong DFO solvers were left out of the analysis: the new SciPy [393] implementations of constrained trust region methods COBYLA and COBYQA, mesh-adaptive direct search NOMAD [229], and covariance matrix adaptation evolutionary search CMA-ES [161]. Recent research also suggests new, competitive finite difference schemes for noisy DFO applications [163, 342]. A critical reading might also reveal that only a few of the considered case studies could be truly considered 'expensive'. Although CUATRO generally performed well across various case studies, it was difficult to pinpoint a unique selling point that would justify further development into a fully-fledged software package. In subsequent benchmarking, we then realized that the sample efficiency of more established Bayesian optimization implementations is difficult to beat - apart from high-dimensional problems and high-

data regimes where they become intractable. My initial assumption that model-based methods would generally outperform direct methods was also not confirmed: Methods like DIRECT-L have negligible algorithm overhead and often 'just work', without the major hyperparameter tuning that can make or break the performance of some model-based DFO solvers.

However, considering that I was the sole developer at the time and that CUATRO was not the primary focus of the paper, its performance was impressive. The benchmarking process provided valuable insights into the challenges of black-box optimization and helped me develop intuition on when to use which DFO solver. Most importantly, it highlighted the shortcomings of common implementations. This led me to explore the applicability of the convex trust region optimizer to mixed-integer decision spaces, batch black-box optimization, high-dimensional domains, and various other exploitation-exploration heuristics. In Chapter 6, we discuss the successful avenues explored and an extension to high-dimensional decision spaces, which now marks CUATRO's most convincing selling point .

Most importantly, this process led me to consider other potential applications of DFO in the PSE literature. As I pursued my interest in decision-making architectures and how mathematical optimization structures can give rise to different decomposition schemes, I began to solidify ideas on how DFO could be used in place of decomposition algorithms and distributed optimization in integrated EWO problems.

4

Coordinating value chain planning under organizational considerations

Contents

4.1 Introduction 98
4.1.1 Decomposition and distributed optimization 99
4.1.2 Organizational considerations and limitations 100
4.1.3 Our contribution 101
4.2 Methodology 102
4.2.1 Problem statement 102
4.2.2 Problem reformulation 102
4.2.3 ADMM by consensus 103
4.2.4 Data-driven coordination 105
4.2.5 Algorithms and software implementation 111
4.2.6 Game-theoretical and other considerations 112
4.3 Case studies 112
4.3.1 Convergence plots 113
4.3.2 Motivating example 113
4.3.3 Collaborative model training 119
4.3.4 Facility location 123
4.3.5 Multi-objective coordination 128
4.4 General observations and discussion 132
4.5 Conclusion 135
4.5.1 Looking back 136

4

In this chapter, we address the challenge of coordinating decisions among agents in enterprise-wide optimization (EWO), where organizational constraints limit the solution of the resulting centralized formulation rather than numerical issues. Inspired by the business needs of the value chain coordination case study from Section 1.2, we propose a 'data-driven' coordination framework. This framework can achieve the same optimum as the equivalent centralized formulation while allowing coordinating agents to maintain autonomy, privacy, and flexibility over their problem formulation and data. Our approach updates the coordinated, or shared, variables using derivative-free optimization (DFO), relying solely on coordinated variables input to agent-level optimal subproblem objective output data. We compare the performance of our framework, using different DFO solvers including CUATRO, against conventional distributed optimization in three case studies: collaborative learning, facility location, and multi-objective blending. This chapter, along with the associated Appendix B, is adapted from work published in [387]. Figures in this chapter are reused from the original publication under the terms of the Creative Commons Attribution 4.0 International License.

4.1 Introduction

In Section 1.1.2, we introduce Enterprise-wide Optimization (EWO) as crucial for companies to remain competitive in an environment of increasingly stringent safety, environmental, and economic requirements [150]. EWO involves the integration of units across 1) all hierarchical levels of decision-making (from design, planning, scheduling, to control), and 2) all geographically distributed (plants, warehouses, etc.) or functional (sourcing, manufacturing, distribution) units. Conventionally, these separate entities are solved sequentially via one-way information flow [87]. For instance, higher-level planning might determine the setpoints of lower-level scheduling without explicitly accounting for lower-level constraints; or geographically separated plants might adjust their operations without explicitly considering the needs of downstream bottleneck plants in the value chain. While these heuristics do not guarantee optimality of the integrated problem, they are sometimes necessary for coordinated decision-making to remain practical and tractable. For example, integrated model-based optimization traditionally requires the solution of a larger-scale centralized optimization model, which quickly becomes computationally intractable in the number of decision variables and constraints [87]. A centralized formulation could also in practice be obviated by organizational complexity (antitrust, privacy, and more).

4.1.1 Decomposition and distributed optimization

One way to alleviate the computational burden of model-based integration is to relax constraints, or replace detailed formulations with surrogate models that are easier to handle by numerical solvers [38, 211, 40]. Usually, this would come at the expense of a degradation in solution quality. However, since EWO aims to coordinate previously decoupled decision-making, the resulting optimization formulations present mathematical structures that can be exploited (Figure 2.2). The resulting problems comprise only few complicating variables and constraints that lend themselves well to decomposition and distributed optimization schemes [289]. Decomposition techniques consist of the iterative solution of a relaxed upper- and reduced-order lower-level problem which can theoretically achieve the same solution quality as the original formulation, while saving computational time (Section 2.2.2). Bilevel [186, 116] and Benders decomposition [301, 383, 325] are among the most prominent techniques for addressing complicating variables, and typically decompose problems over time, and over stochastic realizations of uncertainty respectively. Lagrangean decomposition is particularly useful for tackling complicating constraints, as well as complicating variables by reformulation. Hence, it is also useful for decomposing problems by time, space, or products [187, 284, 369, 356].

Distributed optimization builds on the concepts of dual decomposition techniques (such as Lagrangean decomposition). It has many applications in problems that are separated by complicating constraints, such as the integration of geographically dispersed warehouses or plants along a supply chain [345]. Alternating Direction Method of Multipliers (ADMM) has received special attention as a powerful tool enabling considerable computational savings using minimal information exchange, especially in convex optimization [52]. ADMM repeatedly iterates between the solution of private, localized, lower-level subproblems, and an upper-level problem whose aim is to coordinate the solutions of the private subproblems. The possibility for solving the subproblems in parallel gives rise to significant potential computational savings, similar to what is achieved with decomposition schemes. Nevertheless, ADMM presents clear disadvantages. Despite often being applied in practice, ADMM loses its convergence guarantees on nonconvex problems [318]. Additionally, the method practically only leads to computational savings on continuous problems compared with the centralized solution under certain conditions, namely when the problem is decomposed into *numerous, convex* subproblems [52].

Similarly to how the convergence of first-order gradient descent solvers can be improved using acceleration or momentum, there are several ways to speed up the convergence of ADMM using

similar schemes [60]. Houska et al. [181] propose ALADIN, an algorithm to address ADMM's shortcomings: it speeds up - and includes theoretical conditions for - global convergence to local minimizers on nonconvex problems. ALADIN iterates between the parallel optimization of subproblems and sequential quadratic programming (SQP, Equation 2.21) steps for the coordination around the local subproblem solutions.

4.1.2 Organizational considerations and limitations

While distributed optimization appears beneficial from a computational perspective, much of the literature discussing model-based integration in EWO with relevant solution techniques fails to consider communication and business considerations that could hinder their practical applicability [132, 382, 411]. Yet, due to a significant increase in computational power over the past few decades, software and organizational rather than numerical considerations might become the bottleneck in the integration of computational decision-making [32]. We already consider some business scenarios that might complicate mathematical optimization in value chain coordination in Section 1.2. Specifically, distributed optimization is often approached from a 'top-down coordination perspective': Starting from a centralized model, a decomposition is applied that is expected to lead to computational savings. This presupposes that previously decoupled decision-makers 1) are willing to share their model expressions and data; 2) accept the risk of foregoing a certain degree of autonomy, flexibility, and Nash equilibria for the pursuit of the 'social optimum' of the centralized model; and 3) even have access to known, differentiable expressions as part of their optimization model. Current decision-making architectures were often established within a legal and organizational framework when operations were (and often still are) guided by heuristics rather than numerical optimization. Consequently, the considered problem is often rendered into a multi-agent coordination problem where each agent might represent a separate legal entity with its own autonomy, agenda, technical constraints, and organizational considerations [173, 117].

The organizational context matters when choosing the best coordination scheme. When all agents are willing to collaborate and share differentiable model expressions, powerful distributed optimization techniques can be leveraged for optimal numerical efficiency [124]. When coordinating agents only have access to black-box simulation tools for decision-making, 'data-driven' or 'black-box' optimization tools need to be adapted for the coordination. There are many reviews on data-driven or derivative-free optimization algorithms [228, 12]. In Chapter 3, we benchmark state-of-the-art methods on typical process systems engineering applications. DFO techniques

have also been introduced to solve multilevel formulations in PSE and EWO [422, 36, 35].

4.1.3 Our contribution

In this chapter, we investigate whether derivative-free optimization (DFO) can be used as a viable alternative to distributed optimization solvers for finding the complicating variables in the following coordination problems: each agent is willing to collaborate (i.e. sacrifice suboptimality in their own objective for a 'greater good') and has their own decision-making model, which does *not* have to be white-box. In the context of EWO, interesting scenarios may arise when plants within the same value chain need to coordinate material streams, while each agent optimizes its own objective using third-party, proprietary software (Section 1.2). In this case, gradient information cannot be exploited by methods like ALADIN, which require exact first-order gradient information of the subproblem expressions for the SQP coordination step. These scenarios then call for data-driven optimization approaches, or other distributed optimization techniques like ADMM.

While the performance of different distributed optimization algorithms have been compared with each other and with a centralized solution [366], we investigate under which conditions data-driven optimization outperforms typical distributed optimization solvers such as ADMM. Early results of our contribution were already published in van de Berg et al. [384], where we argue that (potentially imperfect) gradient information becomes increasingly valuable in higher-dimensional decision spaces. Since ADMM's upper-level coordination step involves subgradient information, we only expect DFO to be competitive under specific conditions, i.e. when the number of complicating variables is few relative to the number of private decision variables. Our aim is not necessarily to outperform centralized solution methods. In the methods we compare, computational efficiency is sacrificed for agent privacy, autonomy, and flexibility.

This chapter is organized as follows: In Section 4.2, we introduce our data-driven methodology, compare it with conventional ADMM, and justify our choice of DFO algorithms (CUATRO, Py-BOBYQA, DIRECT-L, and GPyOpt) for the proposed methodology. Section 4.3 presents a motivating mathematical test function and three case studies. Unlike Chapters 3, 5, and 6, we group the problem description, results, and discussion by case study, as each sensitivity analysis focuses on distinct case-specific characteristics, including the number of complicating variables, number of coordinating agents, and topology and size of the subproblem solution space. This is also the reason why we summarize general observations about algorithm convergence *after* the detailed discussion. Readers who prefer to begin with the summary are encouraged to read Section

4.4 before 4.3.

4.2 Methodology

4.2.1 Problem statement

We are interested in optimizing the equivalent of the following centralized, integrated coordination problem:

$$\begin{aligned} \min_{\mathbf{x}\in\mathcal{X}} \quad & \sum_{a=1}^{n_a} f_a(\mathbf{x}_{a,p}, \mathbf{z}) \\ \text{s.t.} \quad & \mathbf{g}_a(\mathbf{x}_{a,p}, \mathbf{z}) \leq \mathbf{0}, \quad a = 1 \dots n_a, \end{aligned} \tag{4.1}$$

where $\mathbf{x}$ refers to the entire decision variable vector within feasibility set $\mathcal{X} \subset \mathbb{R}^{n_x}$. Throughout this chapter, $\mathbf{x}$ contains the 'local' or private decision variables $\mathbf{x}_{a,p} \in \mathcal{X}_{a,p} \subset \mathbb{R}^{n_{x_a}}$ of all n_a agents, and the 'global', shared, or 'complicating' variables $\mathbf{z}$ within the (box-bound) feasibility set $\mathcal{Z} \subset \mathbb{R}^{n_z}$: $\mathbf{x} = [\mathbf{x}_{1,p}, \dots, \mathbf{x}_{n_a,p}, \mathbf{z}]$. The optimization is also subject to n_a sets of local agent constraints $\mathbf{g}_a : \mathbb{R}^{n_{x_a}} \times \mathbb{R}^{n_z} \to \mathbb{R}^{n_{g_a}}$.

This generic problem formulation also implicitly allows for the inclusion of equality constraints in Formulation (4.1) through a reduction in the degrees of freedom of the decision variables, or through an equivalent reformulation into two inequalities. Additionally, Equation (4.1) also allows for the incorporation of 'global', or shared, constraints and objectives. We would call any constraint set $\mathbf{g}_{global}$ 'shared' if it only depends on the shared variables $\mathbf{z}$. Similarly, shared objective terms might either manifest as a separate term $f_{global}(\mathbf{z})$ in a single agent objective, or be incorporated into the objectives of any $n_b \leq n_a$ agents as $\frac{f_{global}(\mathbf{z})}{n_b}$.

4.2.2 Problem reformulation

Problem (4.1) can be reformulated into:

$$\begin{aligned} \min_{\mathbf{z}\in\mathcal{Z}} \quad \min_{\mathbf{x}_{a,p},\, a=1..n_a} \quad & \sum_{a=1}^{n_a} f_a(\mathbf{x}_{a,p}, \mathbf{z}) \\ \text{s.t.} \quad & \mathbf{g}_a(\mathbf{x}_{a,p}, \mathbf{z}) \leq \mathbf{0}, \quad a = 1..n_a. \end{aligned} \tag{4.2}$$

After fixing $\mathbf{z}$, the problem becomes block-separable, which makes the problem amenable to decomposition and distributed optimization. This becomes evident when rewriting (4.2) as its

equivalent constrained (bi-level) optimization problem in (4.3). The coordination step involves an update in the shared variables $\mathbf{z}$. At each iteration, the subproblems are solved *in private* to find the optimal objective $F_a(\mathbf{z})$ and set of private variables $\mathbf{x}^*_{a,p} \in \mathbb{R}^{n_{x_a}}$ corresponding to a set of shared variables $\mathbf{z}$. Agents can maintain autonomy and flexibility by deciding on their own objective and constraint functions, which they do not need to share with other agents. The only information that agents share with a third-party coordinator is the optimal local copy of the shared variables $\mathbf{z}^*_a \in \mathbb{R}^{n_z}$ (4.2.3) *or* the optimal subproblem objective $F_a(\mathbf{z})$ (4.2.4) corresponding to a suggested set of shared variables $\mathbf{z}$. For simplicity's sake, we assume that the subproblems are solved to global optimality. This is a necessary condition to find the globally optimal proposed set of shared variables $\mathbf{z}^*$ in the upper level. In practice however, few industrial problems are provably solved to global optimality and we could still find a 'good' $\mathbf{z}$ in both frameworks if this condition is not met. The bottleneck in achieving global optimality in our considered case studies is usually the coordination step rather than violation of this assumption. The global optimality convergence considerations in ADMM and the suggested framework are addressed in further detail in Sections 4.2.3 and 4.2.4. On top of this, we do not assume that the lower-level problems have to be solved by exploiting known expressions. In fact, the subproblem optimization could involve black-box queries such as proprietary simulations.

$$
\begin{aligned}
\min_{\mathbf{z}\in\mathcal{Z}} \quad & F(\mathbf{z}) \\
\text{s.t.} \quad & F(\mathbf{z}) = \sum_{a=1}^{n_a} F_a(\mathbf{z}), \quad \text{where } \forall a = 1..n_a : \\
& F_a(\mathbf{z}) = \min_{\mathbf{x}_{a,p}\in\mathcal{X}_{a,p}} \quad f_a(\mathbf{x}_{a,p}, \mathbf{z}) \\
& \qquad\qquad \text{s.t.} \quad \mathbf{g}_a(\mathbf{x}_{a,p}, \mathbf{z}) \leq \mathbf{0}
\end{aligned}
\tag{4.3}
$$

4.2.3 ADMM by consensus

Readers without prior knowledge about Lagrangean, Augmented Lagrangean, or other dual decomoposition techniques are advised to read the relevant Section 2.2.2 in the background chapter first. The conventional method that our proposed approach is benchmarked against is the Alternating Direction Method of Multipliers (ADMM) in its consensus form, which is presented in Algorithm 7 and in more detail in Appendix Formulation (B.4). After initialization (**step 1**), ADMM iterates over **steps 2-7** until the evaluation budget is exhausted: This involves the solution of subproblems in private and parallel (**steps 3-5**) to get the local copy of shared variables $\mathbf{z}^{k+1}_a$, and an update

in the shared variables $\mathbf{z}^{k+1}$ and scaled dual variables λ_a^{k+1} based on $\mathbf{z}_a^{k+1}$ (**step 6**).

In **step 4**, each agent optimizes their copy of shared/complicating variables $\mathbf{z}_a$ that minimizes their private objective function, while penalizing any deviation from the *suggested* value of the complicating variables $\mathbf{z}^k$ corrected by the scaled dual variables λ_a^k:

$$\mathbf{x}_{a,p}^{k+1}, \mathbf{z}_a^{k+1} \leftarrow F_a(\mathbf{z}^k) = \underset{\mathbf{x}_{a,p}, \mathbf{z}_a}{\arg\min} . f_a(\mathbf{x}_{a,p}, \mathbf{z}_a) + \frac{\rho}{2}||\mathbf{z}_a - \mathbf{z}^k + \lambda_a^k||_2^2 \quad \text{s.t.} \quad \mathbf{g}_a(\mathbf{x}_{a,p}, \mathbf{z}_a) \leq \mathbf{0}, \tag{4.4}$$

where $||\cdot||_2$ refers to the Frobenius or L_2 norm, and λ_a^k to the scaled dual variables of agent a at iteration k. $\frac{\rho}{2}||\mathbf{z}_a - \mathbf{z}^k + \lambda_a^k||_2^2$ is known as the proximal or penalty term and is useful for stabilizing convergence in the shared variables $\mathbf{z}$. It also makes the formulation robust against local constraints: when there is no feasible set of $\mathbf{x}_{a,p}$ that satisfies all constraints for a given $\mathbf{z}^k$, the solution converges to the nearest feasible $\mathbf{z}_a$, incurring a penalization in the objective. After all subproblems are solved, the set of suggested complicating variables is then updated to $\mathbf{z}^{k+1}$ in the coordination (**step 6**) by averaging the set of optimal complicating variables resulting from the agent subproblems $\mathbf{z}_a^{k+1}$. While the update in the shared variables $\mathbf{z}^{k+1}$ aims to ensure asymptotic primal feasibility, the update in the dual variables λ_a^{k+1} aims to ensure asymptotic dual feasibility. Each agent's dual variables are updated to λ_a^{k+1} based on the difference between $\mathbf{z}^{k+1}$ and the local copy of shared variables $\mathbf{z}_a^{k+1}$, and could be interpreted as an integral error term often encountered in control.

Algorithm 7: Alternating Direction Method of Multipliers (ADMM) by consensus

Input: Agent objectives f_a and constraints $g_{a,l},\ l = 1..n_{g_a}, a = 1..n_a$, Initial shared variable guess $\mathbf{z}^0$, Penalty parameter ρ, Maximum number of function evaluations $N_{f,max}$

Initialisation: Initial dual variables $\lambda_a^0 = [0, \ldots, 0]$ for all n_a agents
for $k = 0..N_{f,max} - 1$ **do**
  for *agent* $a = 1..n_a$ *in parallel* **do**
    $\mathbf{x}_{a,p}^{k+1}, \mathbf{z}_a^{k+1} \leftarrow F_a(\mathbf{z}^k)$, by solving lower-level problem (4.4)
  end
  $\mathbf{z}^{k+1} \leftarrow \frac{1}{n_a}\sum_{a=1}^{n_a} \mathbf{z}_a^{k+1}, \quad \lambda_a^{k+1} \leftarrow \lambda_a^k + \mathbf{z}_a^{k+1} - \mathbf{z}^{k+1}$;
end

Common drawbacks of ADMM include the potential for solutions to require many iterations to converge to high accuracy, and the fact that, in theory, it is only proven to converge to the global optimum when the subproblems are convex [60, 148, 408]. This begs the question if the

coordination step in $\mathbf{z}$ could be improved for convergence speed or solution quality when limited by the number of subproblem calls.

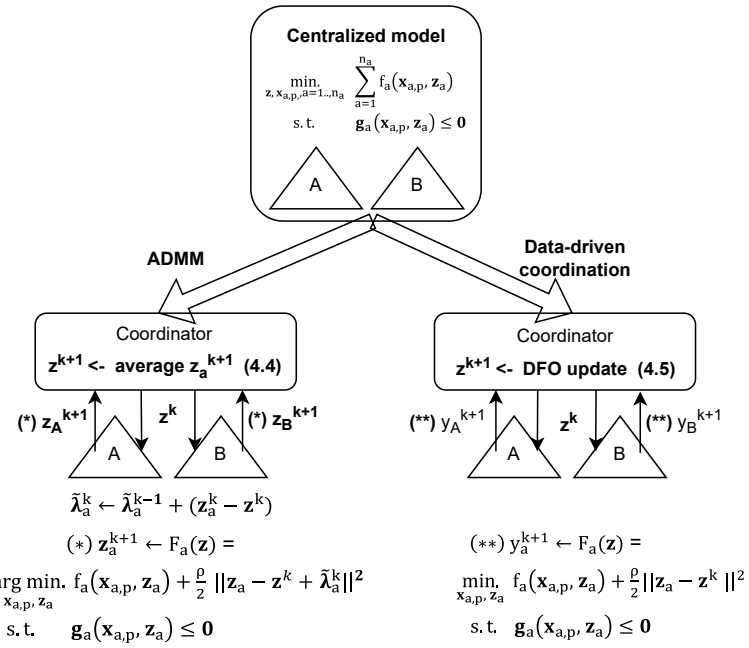


Figure 4.1: We can solve an equivalent centralized formulation either by ADMM or data-driven coordination (⇓). In either case, a coordinator (□) iteratively sends an updated proposed set of shared variables $\mathbf{z}^k$ (↓) to all coordinating agents . The coordinating agents (△) then optimize their private objective according to (*) or (**) for ADMM and data-driven coordination respectively. In ADMM, the subproblems return the optimal local copies of the proposed shared variables $\mathbf{z}_a^{k+1}$, whereas in our approach they return the optimal subproblem objective evaluations y_a^{k+1} (↑). In both frameworks, the penalty parameter ρ determines the extent to which the deviation between the suggested $\mathbf{z}^k$ and optimal set of private variables $\mathbf{z}_a^{k+1}$ is penalized. ADMM's proximal term also includes scaled dual variables λ_a^k, which serve as 'integral error terms' between $\mathbf{z}$ and $\mathbf{z}_a$. In the coordination step, ADMM updates the proposed set of shared variables $\mathbf{z}^{k+1}$ by averaging its local copies $\mathbf{z}_a^{k+1}$, while our data-driven framework uses derivative-free optimization (DFO) to determine $\mathbf{z}^{k+1}$ based on shared variable $\mathbf{z}^{k+1}$ to the sum of optimal subproblem evaluation y_a^{k+1} data. In this case, we only show 2 coordinating agents, but this scheme can be generalized to any number of coordinating agents.

4.2.4 Data-driven coordination

Problem (4.3) views the coordination formulation as a bilevel optimization instance. The successful application of derivative-free optimization (DFO) to multilevel problems [422, 36, 35] suggests that exploring DFO as an alternative to ADMM's subgradient update step is a worthwhile pursuit. The difference between the data-driven coordination framework and ADMM is illustrated in Figure 4.1 and the data-driven coordination framework is illustrated in more detail in Algorithm 8: After initialization (**step 1**), our framework iterates over **steps 2-11** until the evaluation budget is

exhausted: In **step 3**, the upper level aims to find the set of complicating variables that minimize the objective function, constrained on the optimal solution in parallel of the agent-level subproblems in the private variables (**steps 4-9**).

Step 3 uses a DFO algorithm to update the shared variables $\mathbf{z}$ with the aim to minimize the 'black-box' upper-level objective $F(z)$ in (4.3):

$$\min_{\mathbf{z}\in\mathcal{Z}} \quad F(\mathbf{z}), \tag{4.5}$$

where the decision variables $\mathbf{z}$ are subject to box-bound constraints $\mathcal{Z}$. Any box-constrained derivative-free, black-box, data-driven, or 'zeroth-order' optimization algorithm can be used for the solution of the upper level [228, 12, 385]. Since the black-box evaluations are the result of optimizations, these evaluations are considered expensive. The number of evaluations n_{next} that are sampled at each iteration in **step 3** depends on the exploitation-exploration trade-off as well as the sampling strategy of the DFO method used.

$F(\mathbf{z})$ is obtained in **steps 4-9** in a similar manner to (4.3). $F_a(\mathbf{z})$ is treated as the result of private black-box simulations and $F(\mathbf{z})$ is equivalent to the sum of all optimal subproblem solutions, similar to (4.4). The only difference in the subproblem formulation lies in the omission of the dual variables from the proximal term:

$$F(\mathbf{z}) = \sum_{a=1}^{n_a} F_a(\mathbf{z}) \quad \text{where} \quad F_a(\mathbf{z}) = \min_{\mathbf{x}_{a,p},\mathbf{z}_a} f_a(\mathbf{x}_{a,p},\mathbf{z}_a) + \frac{\rho}{2}||\mathbf{z}_a - \mathbf{z}^k||_2^2 \quad \text{s.t.} \quad \mathbf{g}_a(\mathbf{x}_{a,p},\mathbf{z}_a) \leq \mathbf{0} \tag{4.6}$$

The scaled dual is omitted as it only enhances convergence within the rigorous stability scheme of ADMM [282], and can even degrade convergence performance. The reason why we need the proximal term and use local copies $\mathbf{z}_a$ in $f_a(\mathbf{x}_{a,p},\mathbf{z}_a)$ and $\mathbf{g}_a(\mathbf{x}_{a,p},\mathbf{z}_a)$ still holds: $\mathbf{g}_a(\mathbf{x}_{a,p},\mathbf{z}^k)$ might not be feasible for the current iteration, which would lead to a failure of the subproblem.

Notes on convergence guarantees: The attentive reader might recognize that the omission of the dual term throws away any guarantees of convergence to the centralized optimum. In 4.3.2, we discuss the dilemma that ensues when increasing ρ: we lower the degree of acceptable deviations between $\mathbf{z}_a$ and $\mathbf{z}$ and consequently, potential constraint violations in the final solution, but we also lower the speed of convergence. In this work, we handle this trade-off by increasing the penalty term until at least one DFO solver finds a solution with small enough dual residual to be generally deemed acceptable to practitioners. The final solutions (coordinating variables and sum of subproblem objectives) are typically found to be close to those of the centralized solution.

When theoretical convergence guarantees to the centralized optimum are important, we could consider alternative approaches such as using L_1 rather than L_2 proximal terms. In parallel, Krishnamoorthy and Paulson [221] considered an alternative way of using DFO for coordination problems. While we use DFO to solve for the coordinating variables in the upper level, they perform Bayesian optimization of standard acquisition terms in the subproblems, augmented with coordinating terms as derived from ADMM. In follow-up work, they generalize this approach to coordinating terms derived from other decomposition techniques and perform regret analysis [220].

For problems with connected feasible spaces in the coordinating variables, we can stick with a slight modification of our framework. We can first try to find a set of coordinating variables $\mathbf{z}$ that have feasible subproblem solutions when $\mathbf{z}$ is fixed, which can be fed as initial guess to constraint-handling DFO solvers that also handle binary subproblem feasibility information to explore the solution space. This is discussed in more detail in formulation (4.9).

Algorithm 8: Data-driven coordination framework

Input: Agent objectives f_a and constraints $g_{a,l}, l = 1..n_{g_a}, a = 1..n_a$, Box-bound constraints $\mathcal{Z}$, Initial shared variable guess $\mathbf{z}^0$, Penalty parameter ρ, Maximum number of function evaluations $N_{f,max}$

Initialisation: Function evaluation counter $n_f := 1, \mathbf{z}_{best} = \mathbf{z}^0, y_{best} = F(\mathbf{z}^0)$ where $F(\cdot)$ is obtained from (4.6), initial data set $Z := [\mathbf{z}^0], \mathbf{y} := [y_{best}]$

while *($N_{f,max} \geq n_f$)* **do**

  Obtain n_{next} samples Z_{next} from DFO update step at $\mathbf{z}_{best}$ using (4.5) ;

  for $\mathbf{z}_j = \mathbf{z}_1 \dots \mathbf{z}_{next} \in Z_{n_{next}}$ **do**

    for *agent* $a = 1..N$ *in parallel* **do**

      Obtain $F_a(\mathbf{z}_j)$ from (4.6)

    end

    Update datasets: $Z \leftarrow [Z, \mathbf{z}_j], \mathbf{y} \leftarrow [\mathbf{y}, y_j]$, where $y_j = \sum_a^{n_a} F_a(\mathbf{z}_j)$

  end

  Update best iterate: $n_f \leftarrow n_f + n_{next}, \quad y_{best}, \mathbf{z}_{best} \leftarrow \min_{j=1..n_f} . \mathbf{y}_j$;

end

When the subproblems are solved to global optimality, the whole optimization problem can be solved (heuristically or rigorously) to global optimality depending on the function evaluation budget and the convergence certificate of the DFO solver. Among the proposed DFO solvers, only DIRECT-L has this asymptotic global convergence property when no assumptions on the subproblem convexity are made. Since convergence is limited by the number of expensive sub-

problem calls, we do not include overly exploratory methods, such as particle swarm methods. In the next section, we explore any analogies to 'data-driven' ADMM and ALADIN when quadratic surrogates (CUATRO) are used for the DFO step. Additionally, we introduce the other DFO algorithms used, whose choice is informed by the benchmarking in Chapter 3: Py-BOBYQA as the trust region model-based method, DIRECT-L as the direct method, and GPyOpt for Bayesian Optimization. Figure 4.2 shows our selection of data-driven as well as distributed optimization algorithms, and their mutual relations.

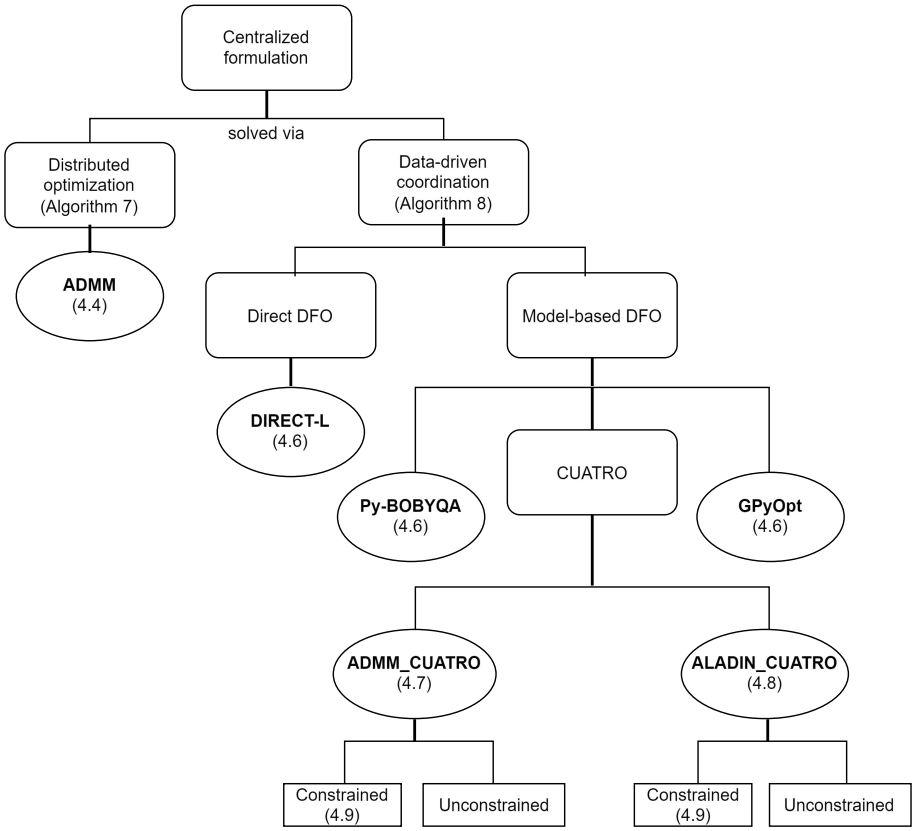


Figure 4.2: Classification of considered algorithms with associated problem formulations in parentheses: We can decide to solve our equivalent centralised formulation either via distributed optimization, namely ADMM, or via our proposed data-driven coordination framework. Within our framework, we can choose any derivative-free optimization (DFO) algorithm. We choose DIRECT-L as a direct DFO algorithm. Among model-based DFO algorithms, we consider Py-BOBYQA and CUATRO as quadratic trust region frameworks and GPyOpt as Bayesian optimization. We also distinguish between ADMM_CUATRO and ALADIN_CUATRO in the way quadratic surrogates are formulated. Each CUATRO version also has the choice of explicit constraint satisfaction

Data-driven distributed optimization

The classification of DFO solvers is discussed at length in Section 3.1.1. We group DFO algorithms into direct and model-based methods. Direct methods optimize an expression by directly handling function evaluations, while model-based methods rely on the intermediate construction

and optimization of surrogates [228]. While Formulation (4.6) is applicable to both direct and model-based DFO solvers, model-based DFO methods can address Problem (4.5) by introducing surrogates $\hat{F}(\mathbf{z})$ in two different ways: One option would be to fit a single surrogate over the sum of the subproblem evaluations for a given proposed set of shared variables $\mathbf{z}$.

$$\min_{\mathbf{z}\in\mathcal{Z}} \quad \hat{F}(\mathbf{z}) \quad \text{where} \quad \hat{F}(\mathbf{z}) \approx \sum_{a}^{n_a} F_a(\mathbf{z}) \tag{4.7}$$

Alternatively, we could allow for separate surrogates to be fitted for each subproblem before the sum of surrogates is optimized in the upper level.

$$\min_{\mathbf{z}\in\mathcal{Z}} \quad \sum_{a}^{n_a} \hat{F}_a(\mathbf{z}) \quad \text{where} \quad \hat{F}_a(\mathbf{z}) \approx F_a(\mathbf{z}) \tag{4.8}$$

For example, we could use convex quadratic surrogates to approximate $F(\cdot)$ as $\hat{F}(\mathbf{z}) = \mathbf{z}^\intercal P\mathbf{z} + \mathbf{q}^\intercal\mathbf{z} + r, \quad P \succcurlyeq 0 \in \mathbb{R}^{n_z \times n_z}, \mathbf{q} \in \mathbb{R}^{n_z}, r \in \mathbb{R}$. In this case, the approach used in Eq. (4.7) is similar to Li et al. [235], and could be loosely referred to as 'Data-driven ADMM'. The approach in (4.8) could then be viewed as 'Data-driven ALADIN', with a crucial difference: ALADIN's quadratic surrogate coefficients are given by the gradient and Hessian (obtained via automatic differentiation) of a second-order Taylor expansion around the local subproblem solutions; in our data-driven counterpart, the quadratic surrogates are obtained via regressing the sum of subproblem evaluations $F_a(\cdot)$ on the shared variables $\mathbf{z}$. Whenever possible, we should leverage automatic differentiation to obtain gradient information with respect to the shared variables as this is expected to yield better convergence, especially in higher dimensions. However, we omit the original ALADIN in the benchmarking. Its use of gradient expressions would ultimately address a different problem statement. We remain agnostic to the subproblem expressions, and focus solely on mappings from the suggested shared variables to their optimal local copies or the optimal subproblem objectives.

CUATRO

We modify the quadratic surrogate trust-region optimizer CUATRO, as presented in Section 3.3, to be used within the 'data-driven ADMM' (ADMM_CUATRO) and 'data-driven ALADIN' (ALADIN_CUATRO) framework. We choose CUATRO as our quadratic surrogate-based DFO algorithm because we expect its safe exploration and flexibility in constraint handling to become useful in dealing with infeasible subproblem evaluations.

When CUATRO is used with explicit constraint handling, the local copies of the shared variables $\mathbf{z}_a$ in the objective evaluation and constraints from (4.6) are replaced by the exact shared variables $\mathbf{z}$:

$$F_a(\mathbf{z}) = \min_{\mathbf{x}_{a,p}} . f_a(\mathbf{x}_{a,p}, \mathbf{z}) \quad \text{s.t.} \quad \mathbf{g}_a(\mathbf{x}_{a,p}, \mathbf{z}) \leq \mathbf{0}, \tag{4.9}$$

where $F_a(\mathbf{z})$ becomes a tuple consisting of the objective and binary feasibility evaluation: $F_a : \mathbb{R}^{n_z} \to \mathbb{R} \times \{0, 1\}$. 1 indicates that the evaluation for $\mathbf{z}$ is feasible. This approach makes the subproblem less robust to constraints, but by returning solver status as feasibility information in addition to objective evaluations, we can use quadratic discrimination in CUATRO (Equation 3.13) to classify the binary feasibility outcomes from the success or failure of the subproblem, thereby concentrating the search within the expected feasible space. Quadratic discrimination - closely related to linear discrimination but using quadratic features - finds the 'hyperellipsoid' in the input space that best separates samples belonging to different classes [50]. ADMM_CUATRO and ALADIN_CUATRO are used with explicit constraint handling if a feasible starting point can be found.

In summary, ADMM_CUATRO and ALADIN_CUATRO leverage Formulations (4.7) and (4.8) in their surrogate construction respectively. In both cases, the subproblems are formulated using (4.6), unless a feasible initial guess is found that enables the use of (4.9). While we benchmark data-driven ADMM and ALADIN against conventional ADMM for the coordination of upper-level, shared variables, we also include the data-driven methodology with other DFO algorithms. Our chosen DFO solvers (Py-BOBYQA, GPyOpt, and DIRECT-L) are presented in the next section, and used in the general data-driven coordination framework (4.6) without explicit constraint handling. The choice between (4.7) and (4.8) is not relevant for DIRECT-L, and Py-BOBYQA and GPyOpt are only used in the single-surrogate framework (4.7).

Py-BOBYQA

We include a competing quadratic trust-region DFO solver. In Section 3.5.2, we show that Py-BOBYQA shines in higher-dimensional, deterministic case studies. It iteratively constructs a linear-quadratic regression-interpolation model of the objective, and determines the next step by minimizing said model within a trust-region framework. We use Py-BOBYQA in its default configuration, but enable the multiple restarts heuristic to avoid getting stuck in local minima. More details on Py-BOBYQA can be found in Section 3.2.3 and in Cartis et al. [72].

GPyOpt

To contrast the 'local', exploitative Py-BOBYQA, we also include a Bayesian optimization (BO) implementation. BO constructs 'global' Gaussian process surrogates over the entire solution space, and leverages these in the optimization of the acquisition function to navigate exploitation and exploration. BO is generally regarded as the go-to framework for expensive black-box optimization within chemical engineering given its sample efficiency [343, 128, 285, 80, 294, 247], but suffers from poor scaling with the number of dimensions and evaluation samples. Informed by Cartis et al. [72], we opt for the GPyOpt implementation, as we prioritize convergence within the low-accuracy regime given our tight budget. GPyOpt [20] is a Python open-source library of BO and builds on GPy, a Python framework for Gaussian process modelling. We use GPyOpt with its default hyperparameters. The interested reader is referred to Section 3.2.8 and Garnett [143] for more information on Gaussian processes and Bayesian optimization.

DIRECT-L

Finally, we also include a 'direct' (model-free) DFO method. Informed by our observations in the previous chapter, we choose DIRECT-L for its consistency in convergence, its interesting exploitation-exploration trade-off, and its asymptotic global convergence properties. We refer the reader to Section 3.2.6 for more details on DIRECT-L and the chosen implementation [194].

4.2.5 Algorithms and software implementation

We use Pyomo [164, 63] as Python-based optimization software to solve the subproblems in (4.4), (4.6, and (4.9). As our numerical solvers, we either opt for Ipopt [394] for continuous, or Gurobi [157] for mixed-integer formulations. Information from these problem instances is then extracted to be used in the upper-level distributed optimization or DFO. We use readily available Python packages for GPyOpt, Py-BOBYQA, and DIRECT-L, and in-house Python implementations of ADMM and CUATRO. The generalized framework for our proposed methodology, along with its comparison to ADMM, is found in Figure 4.1. Figure 4.2 classifies all methods and pairs them with the formulation they are used with. The code for the algorithms and benchmarking is available under `https://github.com/OptiMaL-PSE-Lab/Data-driven-coordination`.

4.2.6 Game-theoretical and other considerations

Coordination problems are interdisciplinary in nature, and are rooted in a rich body of literature within the field of game theory [136, 245]. While we are less interested in the game-theoretical underpinnings of these problems, we need to state some assumptions that justify our proposed method and investigated case studies. First, ADMM and our proposed 'data-driven coordination' techniques involve an upper-level, centralized 'coordination' step. This presumes the existence of a coordinator agent or software that is acting in good faith, which should be a reasonable assumption in EWO. We are also assuming that all agents are honest-but-curious, i.e. that no agent is trying to trick the coordinator or launch any adversarial 'attacks', which is the scope of a different subfield of literature [16] .

Finally, we want to acknowledge that coordination within business settings is subject to many other kinds of considerations: While we investigate algorithms that share as little information as possible, the coordinator-agent and indirectly agent-agent exchange requires an involved legal framework and software infrastrucutre [198]. While ADMM and our proposed data-driven coordination algorithms in principle allow for privacy preservation, supporting such claims would require a thorough investigation into differential privacy and cryptography schemes. The interested reader is referred to Rodríguez-Barroso et al. [319] and Yao [412] for a related discussion.

In the next section, we group problem description and discussion of convergence results by case study, as each has distinct characteristics suitable for sensitivity analysis.

4.3 Case studies

Data-driven coordination is expected to shine in applications that are low-dimensional and non-convex. For this reason, we start with a motivating example before presenting three EWO-specific applications. We use the motivating example in Section 4.3.2 to provide an ideal use case for data-driven coordination and illustrate how decomposition might work in practice for distributed optimization and data-driven coordination. We then benchmark all solvers on this motivating example, investigating the effect of starting points and the penalty parameter ρ on convergence, and comparing the algorithm solution times and overhead of all methods. In the collaborative model training in Section 4.3.3, we analyze the effect of increasing the number of agents and shared variables on convergence. In the facility location problem in Section 4.3.4, we consider different

organizational scenarios and increase subproblem size. Finally, in the multi-agent coordination in Section 4.3.5, we analyze the effect of (non)convexity and solution topology. In Section 4.4, we then summarize the case-study specific results.

4.3.1 Convergence plots

For each application, we first present the case study along with any relevant background, followed by the numerical experiments and their results. For each case study configuration, we provide plots showing the best function evaluation versus number of function evaluations, as well as convergence to the centralized solution optimum versus number of function evaluations, of all algorithms. We use number of function evaluations interchangeably with the number of iterations, as the coordinator usually queries one evaluation $F(\cdot)$ per iteration. We benchmark based on the evaluation budget rather than solution time, since evaluations $F(\cdot)$ involve the expensive solution of optimization problems, making the algorithm overhead negligible. Consequently, this implies that the relative convergence of algorithms does not depend on the size or solution time of the subproblems, when the evaluation budget and solution space topology of the black-box input $\mathbf{z}$ to output $F(\cdot)$ are fixed. We discuss a scenario in Section 4.3.4, where an increase in subproblem size at fixed shared variables is not expected to impact the solution space.

Since the underlying subproblems are deterministic, we only need to include a single realization of the deterministic methods ADMM and Py-BOBYQA. While CUATRO involves some stochasticity in its sampling, we only include a single realization of its two versions (ADMM_CUATRO and ALADIN_CUATRO) at the default seed. For DIRECT-L and GPyOpt however, we include 5 realizations each on all case studies. We plot their median evaluation with their min-max range shaded. While the best function evaluation plots better illustrate low-accuracy convergence (especially relevant for a tight function evaluation budget), convergence plots are sometimes needed to compare high-accuracy convergence.

4.3.2 Motivating example

We first consider the following synthetic toy problem:

$$\begin{aligned} \min_{x_1,x_2,x_3} \quad & (x_1-7)^2+(x_1x_3-3)^2+(x_2+2)^2+(x_2x_3-2)^2 \\ \text{s.t.} \quad & x_1 \geq 0, \quad x_1+x_3=5, \\ & -10 \leq x_1,x_2,x_3 \leq 10. \end{aligned} \tag{4.10}$$

Problem (4.10 becomes trivially separable into 2 subproblems when fixing x_3. This means that this problem can be reformulated into a one-dimensional DFO problem: We replace x_3 with z, and introduce local copies z_A and z_B. Then, we penalize the deviation between z and the local copies using a proximal term in the objective:

$$\begin{aligned} 1) \quad F_A(z) = \min_{x_1,z_A} \quad & (x_1-7)^2+(x_1z_A-3)^2+\frac{\rho}{2}(z_A-z)^2 \\ \text{s.t.} \quad & x_1 \geq 0, \quad x_1+z_A=5, \quad -10 \leq x_1,z_A \leq 10, \\ 2) \quad F_B(z) = \min_{x_2,z_B} \quad & (x_2+2)^2+(x_2z_B-2)^2+\frac{\rho}{2}(z_B-z)^2 \\ \text{s.t.} \quad & -10 \leq x_2,z_B \leq 10. \end{aligned} \tag{4.11}$$

Py-BOBYQA, DIRECT-L, and GPyOpt aim to find z that optimizes $F_A(z)+F_B(z)$. ADMM uses the same subproblems with the addition of $\tilde{\lambda}_A$ and $\tilde{\lambda}_B$ in the proximal term following (4.4). For a trivial problem like this, we can easily find a starting point ($z=4.5$), for which there exist feasible values x_1 and x_2 that satisfy all of the constraints in the subproblems. For this reason, we use ADMM_CUATRO and ALADIN_CUATRO in their constrained form: we omit z_A and z_B as decision variables, omit the proximal term in the subproblem objectives, replace z_A and z_B with z in the objective and constraints, and return the solver status as a binary feasibility evaluation on top of the objective as described in (4.9):

$$\begin{aligned} 1) \quad F_A(z) = \min_{x_1} \quad & (x_1-7)^2+(x_1z-3)^2 \\ \text{s.t.} \quad & x_1 \geq 0, \quad x_1+z=5, \quad -10 \leq x_1,z \leq 10, \\ 2) \quad F_B(z) = \min_{x_2} \quad & (x_2+2)^2+(x_2z-2)^2 \\ \text{s.t.} \quad & -10 \leq x_2,z \leq 10. \end{aligned} \tag{4.12}$$

Figure 4.3 plots the upper-level objective evaluation $F(\cdot)$ of the bilevel formulation (4.3) as a function of the shared variable z. The resulting curve shows an inflection point at $z=3.75$, which can hinder the convergence of ADMM.

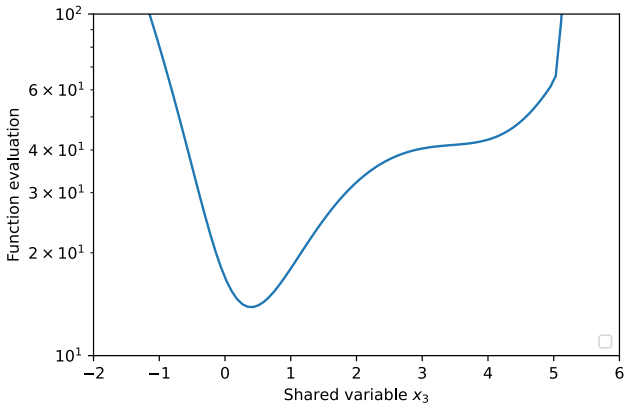


Figure 4.3: 'Black-box' objective $F(z)$ of the motivating example as a function of the shared variable z.

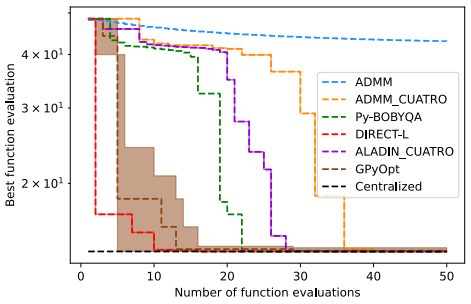
Base case

The nonconvexity of the motivating example calls for data-driven coordination. The problem uses a penalty parameter ρ of 1,000, is one-dimensional with a starting point at $z = 4.5$, and an inflection point around $z = 3.75$ that might hinder convergence of purely exploitative methods. Figure 4.4c shows the solution space convergence of CUATRO and ADMM. While ADMM fails to pass the inflection point, the two CUATRO versions converge to a near-optimal solution. The best function evaluation plot (Figure 4.4a) shows that the DFO variants converge to a low-accuracy solution in the following order from first to last: DIRECT-L, GPyOpt (Bayesian optimization), Py-BOBYQA, ALADIN_CUATRO and ADMM_CUATRO. In this one-dimensional case study, initial exploration in DIRECT-L and GPyOpt encourages escaping the saddle point as quickly as possible. Figure 4.4b then shows that both CUATRO versions achieve a convergence of around 10^{-8} and 10^{-10} for the ADMM and ALADIN variations, respectively, while the other DFO variants only achieve a median convergence up to 10^{-3} or 10^{-5}. This is due to the small trust region radius of the two CUATRO versions in later iterations favouring fine-tuning.

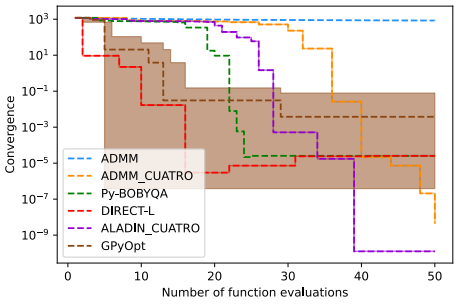
Effect of the starting point

In general, the relative performance of distributed optimization and data-driven coordination depends on the initial guess. We first repeat the default analysis on the same motivating example using the same penalty parameter ($\rho = 1,000$), but starting from an initial guess of $z = -5.0$ instead. When comparing Figures 4.4a and 4.5a, we see that the low-accuracy convergences of the DFO variants are very similar between the two starting points. When starting from $z = -5.0$, the

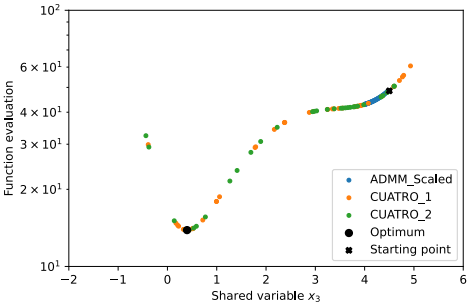
4



(a) Best function evaluation versus number of function evaluations for all methods.



(b) Convergence versus number of function evaluations for all methods.

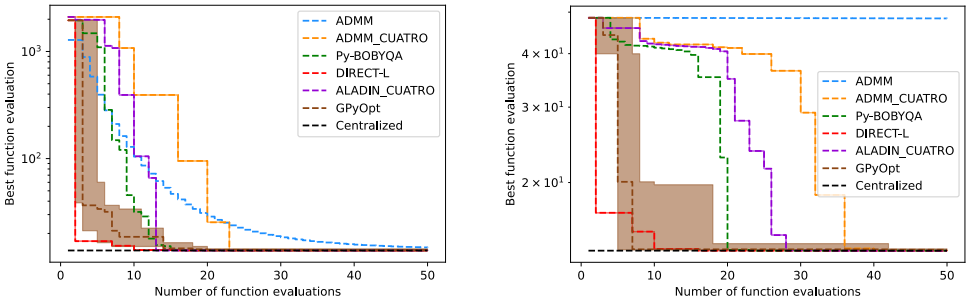


5

(c) One-dimensional solution space convergence for ADMM and both CUATRO versions

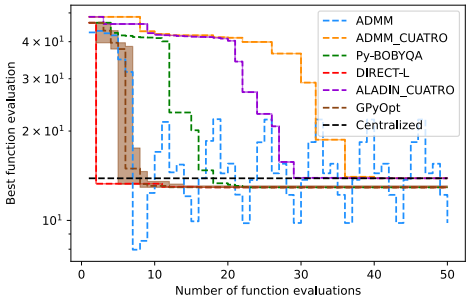
Figure 4.4: Convergence plots for the motivating example starting at $z = 4.5$ with $\rho = 1,000$. For DIRECT-L and GPyOpt, the median best evaluation with shaded min-max range is given over 5 runs

more exploitative solvers ADMM_CUATRO, ALADIN_CUATRO and Py-BOBYQA converge in around ten fewer evaluations. ADMM however displays better performance and manages to converge to a low-accuracy solution in 50 evaluations, since it does not get stuck at an inflection point. While there might be a specific penalty parameter tuning that would result in ADMM outperforming most DFO methods, the DFO methods can be said to more efficiently explore the solution space when the number of shared variables is this small.



(a) $z = -5.0$ and $\rho = 1,000$

(b) $z = 4.5$ and $\rho = 100,000$



(c) $z = 4.5$ and $\rho = 10$

Figure 4.5: Motivating example: Best function evaluation versus number of function evaluations convergence plots under different configurations. For DIRECT-L and GPyOpt, the median best evaluation with shaded min-max range is given over 5 runs

Effect of the penalty parameter

A major drawback of ADMM is its sensitivity to the penalty parameter ρ. In theory, convergence is guaranteed on convex problems for all values of the penalty parameter. In practice however, if the penalty parameter ρ is too high, the local copies of the shared variables $\mathbf{z}_a$ are restricted in how much they can move away from the suggested $\mathbf{z}$ at each iteration leading to slow convergence. If the penalty parameter ρ is too low, the local copies are allowed to move too much, resulting in oscillating, and in nonconvex problems, even diverging behaviour. These phenomena can be observed in Figures 4.5b and 4.5c for $\rho = 100,000$, and $\rho = 10$ respectively compared with the

base penalty parameter of $\rho = 1,000$. Ideally, we would want to schedule ρ, such that a small value is used initially to escape the saddle point, but then gradually increase it to stabilize convergence in later iterations for higher accuracy. Since our computational resources are limited, we do not perform extensive hyperparameter tuning to find the perfect ρ for each subsequent case study. Instead, we start from a small ρ and increase it until an acceptable solution is found.

The relative performance of the DFO methods generally displays less sensitivity to ρ. When comparing Figures 4.4a and 4.5b, we see that the convergence of DFO algorithms is only slightly slowed down on the motivating example. However, the trade-off can manifest in different ways: High penalty parameters can lead to ill-conditioned solution spaces when some suggested shared variables are excessively penalized, resulting in different orders of magnitudes in $F(\cdot)$. Low penalty parameters present another problem: when deviations between the local copies $\mathbf{z}_a$ and their proposed set $\mathbf{z}$ are encouraged (for example, when $\mathbf{z}$ is not feasible), these deviations might be penalized less than the corresponding gain in objective value achieved at $\mathbf{z}_a$ instead of $\mathbf{z}$. This could lead to cases where the sum of subproblem solutions 'outperforms' the optimal evaluation of the centralized formulation, manifesting as sudden increases in the convergence plots, such as with DIRECT-L in Figure 4.4b after around 20 evaluations. In practice, the deviation between the local copies and proposed set of shared variables should be kept within accepted numerical tolerances for industrially relevant applications.

A note on computational time

The overhead of ADMM and DFO solvers in coordinating the solution of the subproblems becomes negligible as the size of the subproblems increases. Even on very cheap problems, the algorithm overhead of most methods is insignificant compared with the total subproblem solution time: In the motivating example, we have two subproblems that require on average 0.3583 seconds (in sequence) to be solved at each iteration or simply 'evaluation'. Since we are using an evaluation budget of 50 in the motivating example, the average total subproblem solution time is around 17.92 seconds. The total algorithm solution and overhead times of all methods are given in seconds on the motivating problem as follows: DIRECT-L (17.95/0.03), ADMM (18.37/0.45), Py-BOBYQA (18.76/0.84), ADMM_CUATRO (20.95/3.04), ALADIN_CUATRO (24.95/7.04), GPyOpt (50.04/32.13).

DIRECT-L and ADMM have the best solution times since they do not require surrogate optimization in their coordination step. Bayesian optimization solvers are also known to present significant overhead when the dimensionality and evaluation budget are high (Section 3.5.1). Thus,

the overhead of some methods, especially GPyOpt, might become the bottleneck in smaller case studies. However, it is safe to say that the algorithm overhead becomes negligible in most practical subproblems where total subproblem solution times exceed tens of seconds.

4.3.3 Collaborative model training

Federated Learning

Our first case study is motivated by Federated Learning (FL) [410]. FL is a subfield within Machine Learning (ML), where multiple clients collaborate under the supervision of a centralized coordinator to train a model under privacy considerations. For example, the aim could be to train a text prediction ML model on decentralized edge devices' data while preserving user privacy. For deep neural networks, this usually involves an iteration over the following steps as described in the FedAVG and FedSGD [256] algorithms: The model is first broadcast to a selection of training agents. The agents then perform a model parameter update step using their data only based on a stochastic gradient descent step obtained by backpropagation. These model parameter updates are then averaged among all participating agents, potentially preceded by an encryption or differential privacy step, before dissemination of the new shared model. The interested reader is referred to Kairouz et al. [198] for an overview of typical FL challenges.

Cross-silo 'learning'

We are more interested in a 'cross-silo' [198] rather than a 'cross-device' setting, where the number of participating agents is fewer and the primary bottleneck resides in the model update computation rather than in the communication. Additionally, first-order methods such as stochastic gradient descent may not always be applicable if models cannot be (cheaply) differentiated for gradient information, as is the case when they are for example represented by a large system of differential equations. To address these issues, we investigate a generalized coordination scheme for collaborative model training using distributed optimization or data-driven coordination. Similar to ADMM, the conventional FL scheme also involves an averaging step of the model parameters as 'shared variables'. But as opposed to FedAVG [256], the subproblem local variable updates usually cannot be obtained under closed form. Instead, we fall back on the more general optimization formulation used in (4.4).

4

Case study

Our case study is based on [400, 409] and addresses collaborative linear regression with a nonconvex truncated loss term augmented by a L_1 norm regularization term. The centralized problem is trivially separable into unconstrained subproblems:

$$F_a(\mathbf{z}) = \min_{\mathbf{z}_a} . \frac{\zeta}{2M_a} \sum_{j=1}^{M_a} (\log\left(1 + \frac{(o_{a,j} - \mathbf{z}_a^\intercal \mathbf{x}_{a,j})^2}{\zeta}\right)) + \xi_a ||\mathbf{z}_a||_1 + \frac{\rho}{2} ||\mathbf{z}_a - \mathbf{z}||_2^2. \tag{4.13}$$

The truncated loss term makes the regression more robust against outliers, while the regularization term encourages sparsity in the regression coefficients. In the linear regression, $\mathbf{x}_{a,j} \in \mathbb{R}^{n_z}$ denotes the jth sample's predictors of the ath agent, and are normally distributed. $\mathbf{z}$ denotes the regression coefficients. $o_{a,j} \in \mathbb{R}$ denotes the jth observed data sample of the ath agent, and is synthesized according to $o_{a,j} = \mathbf{z}^{*\intercal} \mathbf{x}_{a,j} + v_{a,j}$ where $v_{a,j}$ represents random Gaussian noise with a standard deviation spanning a tenth of the number of dimensions. $\mathbf{z}^*$ denotes the ground truth model coefficients, sampled uniformly from $[-1, 1]^{n_z}$, where n_z denotes the dimensionality of the problem, namely the number of model coefficients. We use 3,000 data samples in total, such that each of the n_a agents has $M_a = 3,000/n_a$ data samples. ζ and ξ_a, which control the level of truncation and regularization are set to 3 and 0.01 respectively.

Experiments

The objectives in (4.13) are again tailored towards the implementation of ADMM following (4.4). CUATRO is used in its standard, non-constrained form because the problem itself is not constrained. Starting from an initial solution of $\mathbf{z} = [0, ..., 0]^\intercal \in \mathbb{R}^{n_z}$, we investigate the following configurations of the case studies: We first explore how the comparison of the different algorithms changes with increasing dimensionality ($n_z = 2, 10, 50$) at a fixed number of agents ($n_a = 2$). The second set of experiments explores how an increase in the number of agents ($n_a = 2, 4, 8$) affects convergence when the dimensionality of the prediction coefficients is fixed ($n_z = 6$). All configurations use a penalty parameter ρ of 10.

Effect of the number of shared variables

Much of the algorithm convergence discussion in this section follows that of the motivating example because both problems present a nonconvex objective in the shared variables. The DFO

variants perform particularly well on the two-dimensional case study (Figure 4.6a), taking up to 20 evaluations to converge compared with the 100 of ADMM. The convergence plot (Figure 4.6b) shows that Py-BOBYQA and both CUATRO variants converge to a high degree of accuracy in 20 evaluations, which takes DIRECT-L around 40 evaluations to reach. ADMM and GPyOpt only reach a (median) convergence of 10^{-5} and 10^{-7} respectively compared with 10^{-10} of the other three methods. Like in the motivating example, GPyOpt displays significant variance in its final convergence.

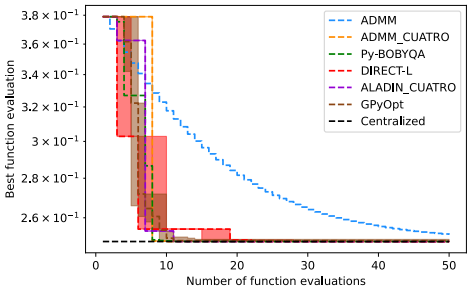
Figure 4.6c shows that when the dimensionality is increased to ten, the CUATRO variants lose their competitiveness with ADMM. Figure 4.6d illustrates how DIRECT-L displays a similar median convergence speed to ADMM. Py-BOBYQA and GPyOpt make substantial progress in the first 20 evaluations. The best final convergence realization of GPyOpt matches that of ADMM (10^{-7}), while Py-BOBYQA is the best at fine-tuning solution accuracy (10^{-10}).

In the 50-dimensional case, we would expect ADMM to significantly outperform all DFO variants given its competitive advantage as a subgradient method, which becomes increasingly important in higher dimensions. However, in Figure 4.6e, we see that after around 100 evaluations, Py-BOBYQA is the only variant to find the optimum. This makes sense given that the starting point of the case study $\mathbf{z} = [0, \dots, 0]^\intercal$ is already quite close to the optimal solution. This gives exploitative methods (ADMM and Py-BOBYQA) the upper hand. Finally, ADMM, despite making consistent progress, is slower at fine-tuning the optimum than Py-BOBYQA.

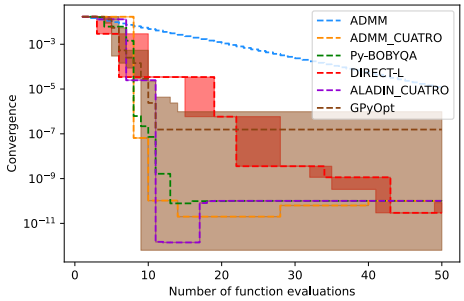
Effect of the number of agents

Starting with a dimensionality of six and two coordinating agents, we observe a similar convergence pattern in Figures 4.7a and 4.7b to the ten-dimensional case in Figures 4.6c and 4.6d. ADMM, Py-BOBYQA, ALADIN_CUATRO and GPyOpt display a similar relative performance, while ADMM_CUATRO makes consistent progress and matches the convergence found for BO and Py-BOBYQA in 35 and 55 evaluations respectively.

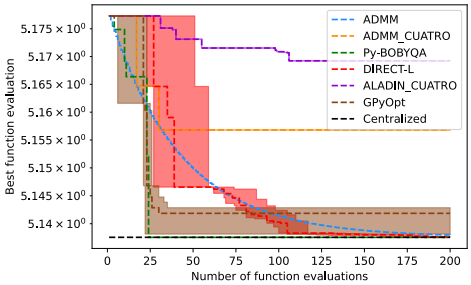
Overall, we observe that with an increasing number of coordinating agents, the optimality gap increases between the final total function evaluation and the centralized optimum. There will always be small numerical differences between $\mathbf{z}_a$ and $\mathbf{z}$, which are penalized in the proximal terms $\frac{\rho}{2}||\mathbf{z}_a - \mathbf{z}||_2^2$. With an increasing number of agents, the relative importance of these proximal terms is strengthened, which becomes even more apparent when the optimal objective evaluation is close to the starting point as is the case for these problems.



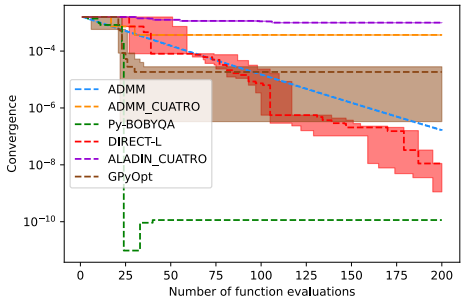
(a) Best function evaluation, $n_a = 2$ and $n_z = 2$.



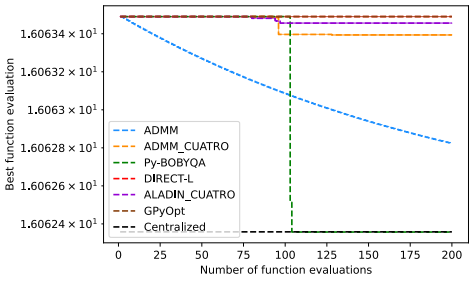
(b) Convergence to optimum, $n_a = 2$ and $n_z = 2$.



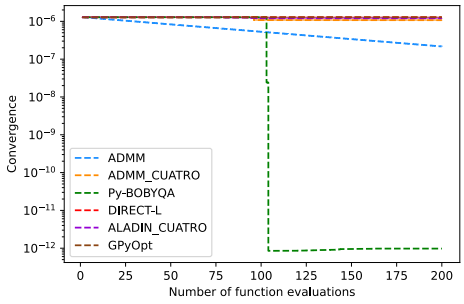
(c) Best function evaluation, $n_a = 2$ and $n_z = 10$.



(d) Convergence to optimum, $n_a = 2$ and $n_z = 10$.



(e) Best function evaluation, $n_a = 2$ and $n_z = 50$.



(f) Convergence to optimum, $n_a = 2$ and $n_z = 50$.

Figure 4.6: Truncated regression case study: Best function evaluation versus evaluation budget and convergence versus evaluation budget plots at a fixed number of agents ($n_a = 2$), and an increasing number of shared variables ($n_z = 2, 10, 50$). For DIRECT-L and GPyOpt, the median best evaluation with shaded min-max range is given over 5 runs

The *relative* performance however of the DFO algorithms with respect to each other and ADMM does not seem to change with an increasing number of coordinating agents. ALADIN_CUATRO makes poor progress, DIRECT-L tracks the convergence of ADMM, while Py-BOBYQA and ADMM_CUATRO quickly find the best function evaluations. GPyOpt again makes quick initial progress but displays a lot of variance in the best evaluation found. It is interesting to see that for this particular case, the convergence speeds of Py-BOBYQA, ADMM_CUATRO, and GPyOpt do not tend to deprecate with an increasing number of agents.

4.3.4 Facility location

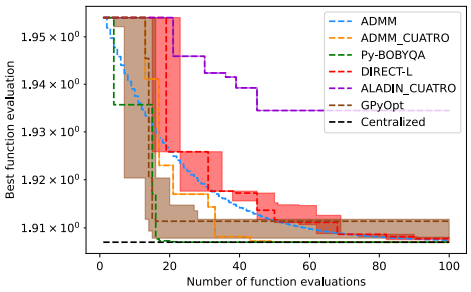
Value chains

A key part of EWO is the design and operation of supply chains [338]. In an idealized setting, all stakeholders within a given value chain are willing to collaborate and share model information with a centralized coordinator. In practice however, antitrust and game-theoretical considerations might prevent stakeholders from fully collaborating. There is ample literature about 'Stackelberg Leader-Follower games' [414], where some supply chain agents 'take the first step' in deciding on the optimal location of their plants subject to other players reacting optimally with respect to their private objective. Yet, there is much value to be captured in moving away from these 'Nash equilibria', and approaching a coordinated optimum along the 'Pareto front'. To this end, a coordinator can optimize a (fairness-guided) game-theoretical operator that scalarizes and trades off the conflicting criteria of competing stakeholders [81, 87, 415, 9, 219].

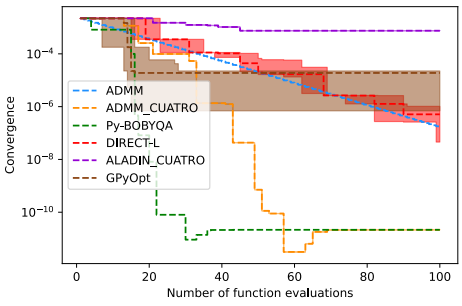
This is especially relevant for the design of emerging supply chains with distinct characteristics, such as biomass [322, 139] or (bio)pharmaceutical value chains [337, 328]. These 'social optima' are often obtained as the result of centralized optimization formulations, which can be decomposed for numerical tractability, or in our case to fit organizational considerations.

Case study

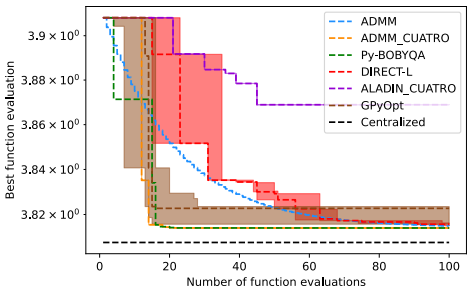
We consider a continuous facility location problem in two-dimensional continuous space, which belongs to the general class of Capacitated Multi-facility Weber Problems [8]. The objective is to find the location, production, and connecting flows of all facilities that minimize a total cost. These 'shared' variables are few relative to the number of private variables and parameters, the latter including local cost parameters, technical upper and lower bounds, binary variables, and distances



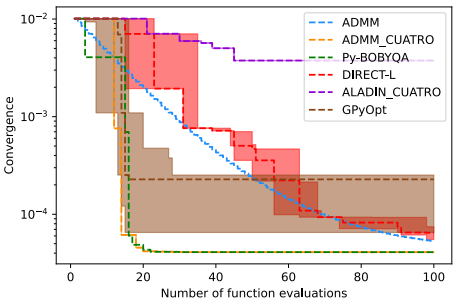
(a) Best function evaluation, $n_a = 2$ and $n_z = 6$.



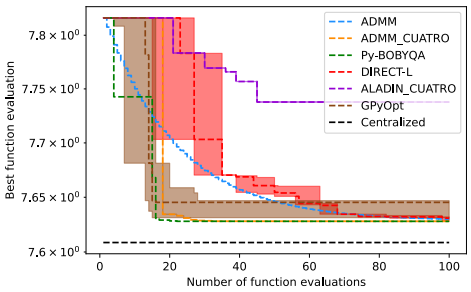
(b) Convergence to optimum, $n_a = 2$ and $n_z = 6$.



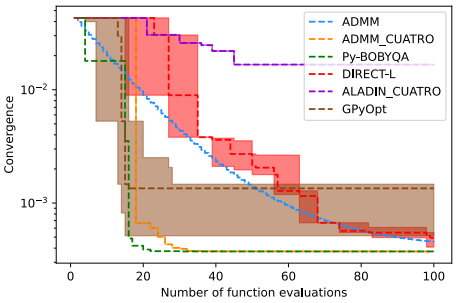
(c) Best function evaluation, $n_a = 4$ and $n_z = 6$.



(d) Convergence to optimum, $n_a = 4$ and $n_z = 6$.



(e) Best function evaluation, $n_a = 8$ and $n_z = 6$.



(f) Convergence to optimum, $n_a = 8$ and $n_z = 6$.

Figure 4.7: Truncated regression case study: Best function evaluation versus evaluation budget and convergence versus evaluation budget plots at an increasing number of agents ($n_a = 2, 4, 8$), and a fixed number of shared variables ($n_z = 6$). For DIRECT-L and GPyOpt, the median best evaluation with shaded min-max range is given over 5 runs

to/from facilities to name but a few. We use the same formulation as Lara et al. [227] with some key differences. In particular, we assume the presence of two suppliers and markets each. We fix the number of facilities to be built to either one or two, which still gives rise to a Generalized Disjunctive Problem (GDP). We also define the distances between agents and facilities using the L_1 norm, rather than the L_2 norm for computational efficiency. The GDP is finally reformulated into an MINLP using big-M constraints, implemented in Pyomo [164, 63], and solved using Gurobi [157]. The detailed formulation can be found in B.2.

Decomposition

In our first experiment, we compare two different decompositions, motivated by separate business scenarios. In the first scenario, we consider the two node types (suppliers and markets) to represent separate agents, companies, and legal entities. This means that the two nodes of the same type are allowed to share model expressions with each other, but not with others. For each problem, we need to find the following shared variables: the two-dimensional location of the facilities ($2n_p$ variables) and their production (n_p variables). We set the number of processing facilities to one or two ($n_p = 1, 2$), resulting in three or six shared variables ($n_z = 3, 6$). Detailed formulations can be found in B.2

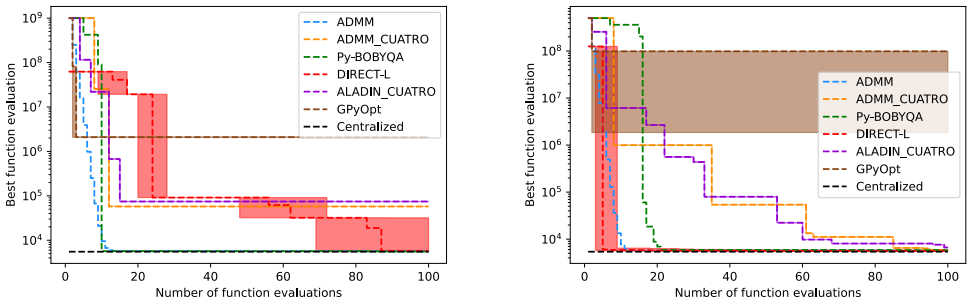
In our second scenario, we consider all four nodes (suppliers and customers) to constitute their own separate legal entity with privacy considerations. This means that we need to decompose the problem into four. Unfortunately, the presence of complicating constraints, i.e. on the total amount transported to and from a facility, prevents each agent from independently deciding on the material flow between their node and the facilities. Consequently, the transport variables $f_{s,p}$ and $f_{p,c}$ become part of the set of shared variables, where s, p, c denote the supplier, processing, and market node indices respectively. This would in principle add another $4n_p$ shared variables. However, some complicating equality constraints only rely on the shared transport variables, and can be used to reduce the degrees of freedom in the shared variables from $7n_p$ to $5n_p$. Detailed formulations can again be found in B.2.

In our second experiment, we investigate the effect of increasing subproblem size on the relative algorithm convergence at a constant number of shared variables. In our first scenario, we fix the number of processing facilities to two, leaving us with six shared variables in the presence of two coordinating agents. Then, we increase the number of suppliers and customer nodes from two to five each ($n_s = n_c = 2, 3, 4,$). This increases the number of private variables and constraints in

the subproblems and the CPU time required to solve them. Yet, we expect this to have no major effect on the solution topology given the similar nature of the subproblems.

Two agents: nodes of the same type can share model expressions

Figure 4.8a shows that for the two-agent three-dimensional case, the exploitative methods ADMM and Py-BOBYQA display a similar convergence speed and are the only methods to converge quickly to the optimum. The two CUATRO versions converge to the same similar suboptimal point. The median DIRECT-L run manages to find the same optimum as ADMM and Py-BOBYQA after around 90 evaluations. GPyOpt only makes little progress. Figure 4.8b then shows that in the six-dimensional case, Py-BOBYQA and ADMM again outperform both CUATRO variants and Bayesian Optimization. As expected, the subgradient information of ADMM leads to faster convergence compared with Py-BOBYQA when the dimensionality is increased. Interestingly, DIRECT-L outperforms both exploitative methods, suggesting that the optimum is close to the center of one of the initial partitions used by DIRECT-L, which depends mostly on the user-given box bounds on the shared variables. The convergence versus number of function evaluation plots are omitted since they do not provide any additional information, as ADMM, Py-BOBYQA, and DIRECT-L converge to about the same accuracy.



(a) Best function evaluation, $n_a = 2, n_p = 1, n_z = 3$.

(b) Best function evaluation, $n_a = 2, n_p = 2, n_z = 6$.

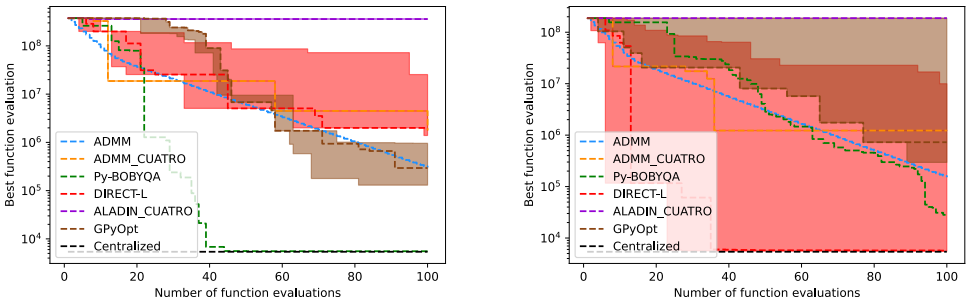
Figure 4.8: Facility location convergence plots when increasing the number of processing facilities ($n_p = 1, 2$) and shared variable dimensionality ($n_z = 3, 6$) between two agents ($n_a = 2$). For DIRECT-L and GPyOpt, the median best evaluation with shaded min-max range is given over 5 runs

Four agents: no model expressions are shared

Figure 4.9 shows a markedly different relative algorithm performance in the four-agent case compared with the two-agent case in Figure 4.8. In Figure 4.9a, only Py-BOBYQA manages to navigate

the solution space more effectively than ADMM and is the only method that converges to the optimum. GPyOpt finds a similar optimum to ADMM, while the other DFO variants display worse performance than ADMM. Figure 4.9b shows a peculiar convergence pattern in the ten-dimensional case for all methods aside from ALADIN_CUATRO, which makes no progress whatsoever. Py-BOBYQA displays similar convergence patterns to ADMM, and both find a slightly better optimum than ADMM_CUATRO. GPyOpt's final evaluations compete with ADMM. DIRECT-L is the only method to converge to the optimum in its median evaluation but displays significant variability in its convergence. Like in the higher-dimensional two-agent case in Figure 4.8b, the solution space topology and input-bounds give rise to partitions whose center is close to the optimum. Convergence versus number of function evaluation plots are omitted again as they provide no additional information.

The differences between Figures 4.9 and 4.8 are partly due to the inclusion of additional shared variables in the form of facility-to-agent transport links, which must be coordinated between all supplier and customer nodes. The presence of complicating equality constraints introduces a trade-off between shared variable dimensionality and solution space complexity. In the multi-agent coordination case study, complicating constraints are handled by maintaining the number of shared variables and penalizing infeasibilities in the shared constraints through the proximal term. In this case study, complicating constraints are addressed by reducing the degrees of freedom in the shared variables using material balances on the facility nodes. The choice of how best to handle complicating constraints is case-specific. However, reducing the degrees of freedom with constraints typically favours convergence for data-driven methods.



(a) Best function evaluation, $n_a = 4, n_p = 1, n_z = 5$

(b) Best function evaluation, $n_a = 4, n_p = 2, n_z = 10$

Figure 4.9: Facility location convergence plots when increasing the number of processing facilities ($n_p = 1, 2$) and shared variable dimensionality ($n_z = 5, 10$) between four agents ($n_a = 4$). For DIRECT-L and GPyOpt, the median best evaluation with shaded min-max range is given over 5 runs.

Effect of the subproblem size

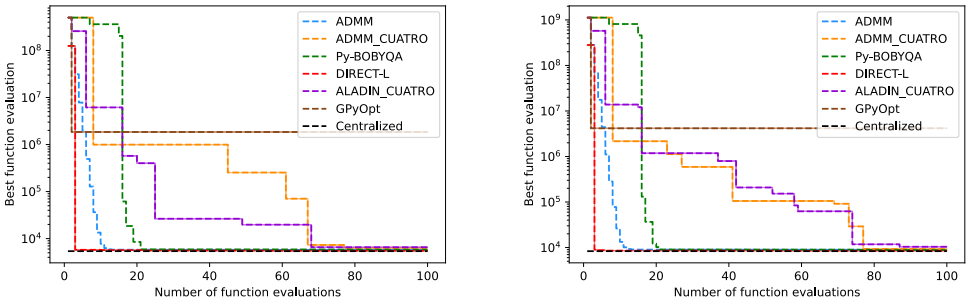
In previous experiments, the performance of most DFO solvers deteriorates relative to ADMM as the number of shared variables increases. In Figure 4.10, we examine the effect of increasing subproblem size while keeping the number of shared variables constant, by increasing the number of supplier and customer nodes ($n_s = n_c = 2, 3, 4, 5$) rather than the number of processing facilities ($n_p = 2$). The performances of ADMM, Py-BOBYQA, and GPyOpt remain largely consistent, while the performance of DIRECT-L decreases slightly in one run. The relative performances of ADMM_CUATRO and ALADIN_CUATRO compared with ADMM deteriorate only in the runs with three and four supplier and customer nodes. This could be explained by the introduction of additional private variables and constraints in the subproblems, which may lead to a restriction in the feasible shared variable space, a stricter penalization of certain proposed shared variables, and more ill-conditioning. However, the general solution topology should not change significantly since the subproblems remain mixed-integer quadratic programs with similar structure.

Our analyses suggest that DFO algorithms scale better relative to ADMM when subproblem size increases than when the number of shared variables increases. These results also strengthen our argument that the relative performance of ADMM and DFO solvers does not depend on subproblem size when controlling for evaluation budget, number of shared variables, and solution topology. Thus, our conclusions on the relative performance of ADMM and DFO solvers on smaller case studies extend to larger subproblems, as long as the number of shared variables and the solution space topologies can be maintained.

4.3.5 Multi-objective coordination

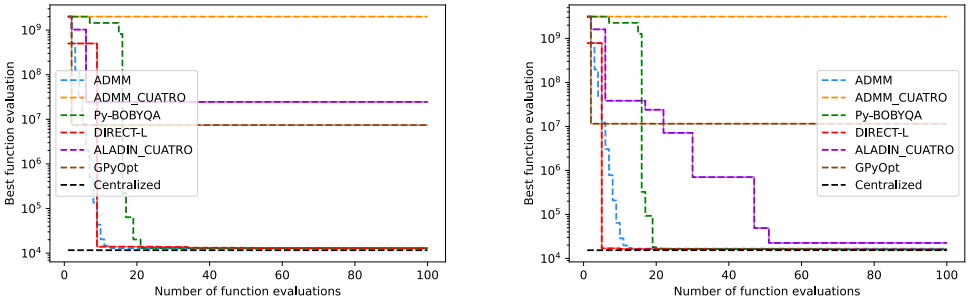
Case study

This case study is inspired by a scenario, where two supply chain stakeholders want to coordinate a plant's feed composition that minimizes conflicting economic and environmental objectives. We consider an animal feedstock optimization problem described in earlier conference proceedings [384], which differs from our motivating scenario only in the interpretation of the quality and environmental impact coefficients. The two agents are secretive about the intricacies of their proprietary optimization and simulation software used in pursuing their private objectives, making ADMM or data-driven coordination necessary for effective collaboration. The centralized problem becomes trivially decomposable after fixing the feedstock composition variables $\mathbf{z}$. Agent A op-



(a) $n_s = n_c = 2$.

(b) $n_s = n_c = 3$.



(c) $n_s = n_c = 4$.

(d) $n_s = n_c = 5$.

Figure 4.10: Facility location convergence plots with two agents ($n_a = 2$) and two processing facilities ($n_p = 2$ and $n_z = 6$) when increasing the number of supplier and customer nodes per subproblem from two to five ($n_s = n_c = 2, 3, 4, 5$). For DIRECT-L and GPyOpt, the median best evaluation with shaded min-max range is given over 5 runs

timizes an economic blending problem, while Agent B optimizes the output of an environmental impact simulation:

$$F_A(\mathbf{z}) = \min_{\mathbf{z}_A \in \mathbb{R}^{n_z}, \mathbf{y} \in \{0,1\}^{n_z}} \quad \mathbf{z}_A^\intercal \mathbf{c} + \frac{\rho}{2} ||\mathbf{z}_A - \mathbf{z}||_2^2 \tag{4.14a}$$

$$\text{s.t.} \quad \mathbb{1}^\intercal \mathbf{z}_A = 1, \quad \mathbf{q}^L \leq \mathbf{z}_A{}^\intercal Q_{qual} \leq \mathbf{q}^U, \tag{4.14b}$$

$$\mathbf{0} \leq \mathbf{z}_A \leq \mathbf{y}, \quad \mathbb{1}^\intercal \mathbf{y} \leq n_{int}, \tag{4.14c}$$

$$F_B(\mathbf{z}) = \min_{\mathbf{z}_B \in \mathbb{R}^{n_z}} \quad \sum_{i}^{n_z} (e_i z_{B,i}^{a_i} + \sum_{i' \in \mathcal{L}_i} z_{B,i} z_{B,i'} + \frac{\rho}{2} (z_{B,i} - z_i)^2). \tag{4.14d}$$

$\mathbb{1} \in \mathbb{R}^{n_z}$ refers to the identity vector. Essentially, the economic blending problem $F_A(\cdot)$ minimizes the feedstock cost given component costs $\mathbf{c}$, quality matrix Q_{qual}, as well as upper and lower bounds $(\mathbf{q}^U, \mathbf{q}^L)$ on the feedstock mix quality. The dummy composition variables $\mathbf{z}_A$ of agent A are also subject to the corresponding binary variables $\mathbf{y}$, which are active if the associated feedstock is non-zero. The number of active composition variables is constrained by n_{int}. Agent A's formulation is a mixed-integer convex quadratic problem. Agent B's objective term is composed of a sum of linear, bilinear, and quadratic terms. Each feedstock i has its environmental impact coefficient e_i, its power coefficient $a_i \in \{1, 2\}$, and its sparse set of bilinear interactions $\mathcal{L}_i$.

Black-box simulations in the subproblems

If the proximal term in the environmental subproblem is strongly penalized with a high ρ, its optimal solution tends towards $\mathbf{z}_B = \mathbf{z}$. In other words, since this environmental subproblem does not involve any local constraints, $F_B(\mathbf{z})$ could theoretically be the result of a black-box simulation instead of an optimization problem. Our data-driven framework could readily handle problems where the lower level is obtained via simulation, since progress only relies on objective evaluations. ADMM however requires optimization approaches to update the local copies of the shared variables. In approaching the collaborative optimum with $\mathbf{z}$, ADMM relies on the pull that both subproblems exert on $\mathbf{z}_A$ and $\mathbf{z}_B$ to move the solution closer to their 'selfish' economic and environmental objectives $F_A(\cdot)$ and $F_B(\cdot)$ respectively. Otherwise, we would end up only optimizing $F_A(\cdot)$. For this reason, we do not benchmark the case study when $F_B(\cdot)$ is given by simulation.

Experiments

We perform the mixed-integer, nonconvex coordination problem given by (4.14) on an increasing number of shared feedstock variables ($n_z = 10, 25$). Additionally, we perform the same experiment on a nonlinear but convex version of the previous problem. This is obtained by relaxing all mixed-integer constraints in (4.14c), and by omitting the bilinear terms in the environmental problem (4.14d).

Figure 4.11 gives the best function evaluation and convergence to optimum plots for the 10- and 25-dimensional multi-agent coordination problems in their convex and nonconvex variants. All configurations use a penalty parameter ρ of 5,000. Figures 4.11a and 4.11c show that all methods manage to find at least a low-accuracy optimum in the convex ten-dimensional variant. However, ADMM converges considerably faster than the fastest DFO variant (20 and 200 evaluations for ADMM and Py-BOBYQA to achieve the same accuracy respectively). It makes sense that both ADMM and Py-BOBYQA outperform more explorative methods for the considered configuration given their purely exploitative behaviour. This case also highlights the respective strengths of ADMM and Py-BOBYQA. ADMM is in general very fast to converge to the neighbourhood of a local optimizer. However, ADMM struggles to fine-tune the optimum. When the evaluation budget allows for it, Py-BOBYQA takes more evaluations to find this neighbourhood, but is more efficient at fine-tuning the solution quality. Figure 4.11c shows that ADMM's final convergence is orders of magnitude worse than that of Py-BOBYQA (10^2 and 10^0 respectively). The discussion of the 25-dimensional convex configuration follows that of the 10-dimensional convex one. The relative performance of the algorithms is very similar, with the exception that ADMM's final convergence still displays a considerable optimality gap (Figure 4.11b). Py-BOBYQA is the only method to converge to a high-accuracy solution given its exploitative nature and its ability for fine-tuning.

The ten-dimensional nonconvex configuration presents conditions that favour data-driven approaches. The relative performance of the algorithms in Figure 4.11e is similar to that of its convex counterpart in Figure 4.11a with two notable exceptions: ADMM and Py-BOBYQA - both purely exploitative methods - converge to a *local* minimizer in 100 evaluations. ADMM makes faster initial progress, but Py-BOBYQA finds a slightly better solution. ADMM_CUATRO and the median run of DIRECT-L, due to their extensive initial exploration, manage to escape the local minimizer and converge to a low-accuracy neighbourhood of the global optimum. The discussion of the higher-dimensional nonconvex case again follows that of its convex counterpart. ADMM and Py-BOBYQA are the only methods again to converge to at least a near-optimal solution.

Py-BOBYQA finds a better solution quality, but takes significantly longer than ADMM. This highlights the importance of (sub)gradient information in higher-dimensional solution spaces and underscores that, when constrained by evaluation budget, prioritizing a *good* solution is often more practical than striving for the *best* one.

4

4.4 General observations and discussion

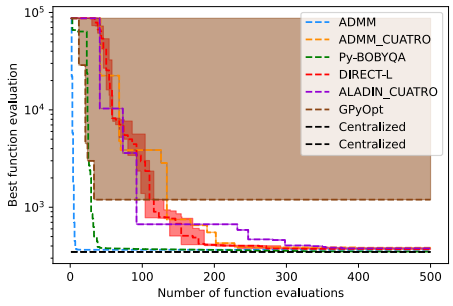
In this section, we present general observed trends that summarize the case-specific observations of Figures 4.4 to 4.11. A high-level comparison between the performance of the data-driven coordination framework and ADMM is summarized in Table 4.1.

Table 4.1: Performance comparison between ADMM and data-driven coordination based on mathematical problem and desired solution characteristics

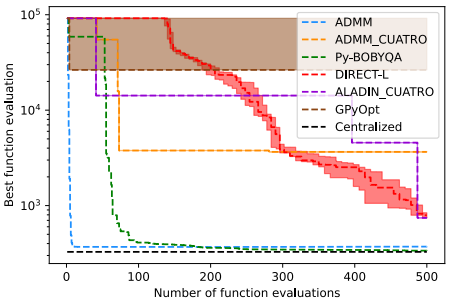
Consideration	**ADMM**	**Data-driven coordination**
Number of shared variables **z**	**Scales better** with dimensions	Shines in lower dimensions
Convergence speed	**Quick initial progress** but dependent on penalty parameter ρ	Can use exploitative DFO solver to **better fine-tune optimum**
Convergence guarantee	Guaranteed **for convex subproblems**.	**Depends on DFO solver** e.g. DIRECT-L guarantees asymptotic global convergence
Solution space topology in **z**	Can get stuck at nonconvexities	Can use **explorative DFO** solver to **escape local minima**
Organizational and software	Requires numerical optimization for the subproblem solution	**More flexibility** in the subproblem solution (black-box simulations, heuristic evaluations, ...)

ADMM: manages to converge to at least a *local* minimizer if given enough function evaluations. If the proposed starting point is far from the optimum, initial progress with ADMM is generally fast. However, ADMM is found to be ill-suited for fine-tuning near-optimal solutions, which is in line with literature [52, 60].

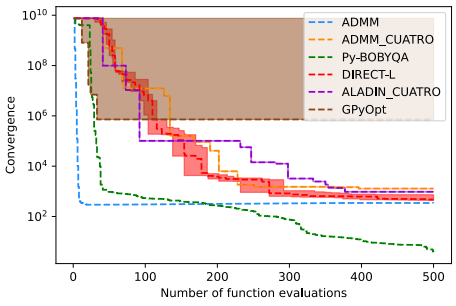
Data-driven coordination: The performance of all data-driven coordination alternatives improves with respect to ADMM the more ill-behaved and low-dimensional the solution space in the shared variables is. For the EWO case studies, we examine both lower- and higher-dimensional configurations. In the lower-dimensional case, there is consistently at least one DFO variant that outperforms ADMM. Understanding the way these algorithms approach the exploration-exploitation trade-off is key to this observation. The coordination step in ADMM is purely exploitative. It



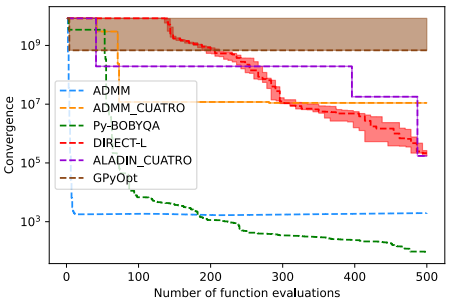
(a) Best function evaluation, $n_z = 10$, convex.



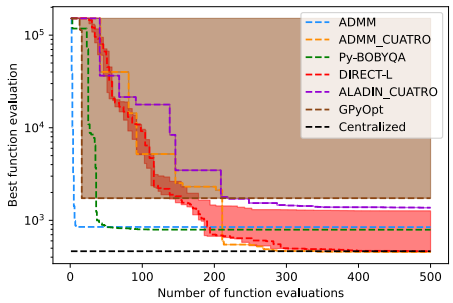
(b) Best function evaluation, $n_z = 25$, convex.



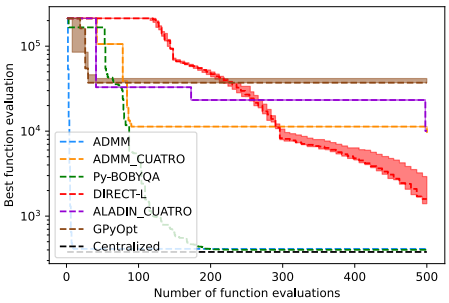
(c) Convergence, $n_z = 10$, convex.



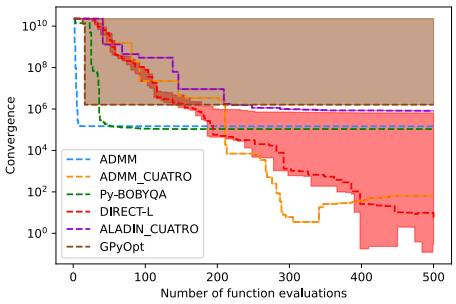
(d) Convergence, $n_z = 25$, convex.



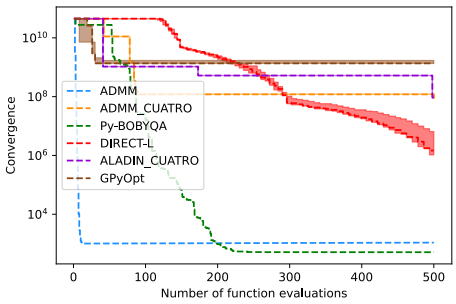
(e) Best function evaluation, $n_z = 10$, nonconvex.



(f) Best function evaluation, $n_z = 25$, nonconvex.



(g) Convergence, $n_z = 10$, nonconvex.



(h) Convergence, $n_z = 25$, nonconvex.

Figure 4.11: Multi-objective coordination case study: best function evaluation versus evaluation budget and convergence versus evaluation budget plots for the convex and nonconvex variations with an increasing number of shared variables ($n_z = 10, 25$). For DIRECT-L and GPyOpt, the median best evaluation with shaded min-max range is given over 5 runs

cheaply extracts subgradient information from the subproblems to approach the coordinated optimum as quickly as possible. The relative performance of DFO algorithms against ADMM and of explorative versus exploitative methods is determined by the mathematical properties of the case study.

DFO variants: Highly explorative frameworks like Bayesian optimization perform well in lower-dimensional applications, where thorough exploration is more likely to be rewarded by faster convergence to the optimum. Other DFO exploration routines can also be useful in escaping local optima. DIRECT-L, as a global optimization algorithm, usually makes slow but consistent progress as its function evaluations are used to thoroughly explore all partitions of the solution space - unless the optimum happens to be in the center of one of the initial partitions. CUATRO, with its decreasing trust region framework, encourages initial exploration and later exploitation. ADMM_CUATRO usually displays superior performance over ALADIN_CUATRO. However, the choice between these two variants is in practice more motivated by organizational considerations: whether agents are comfortable with sharing detailed subproblem objective or only approximate surrogate information.

Py-BOBYQA: The previously discussed DFO algorithms tend to outperform ADMM only in lower-dimensional applications. Py-BOBYQA is the only DFO algorithm that has the potential to be competitive with ADMM in higher-dimensional applications given its highly exploitative nature. This is why Py-BOBYQA tends to converge to the same local optima as ADMM. While ADMM displays faster convergence to low-accuracy regimes of the solution, Py-BOBYQA can find higher-accuracy solutions at the expense of more function evaluations.

Significance of the penalty parameter ρ: The value of the penalty parameter ρ should *in principle* not influence the quality of the solutions found. In fact, the penalty term should approach zero at the optimum, since the local copies of the shared variables $\mathbf{z}_i$ should approach the optimal suggested shared variables $\mathbf{z}$. In practice however, the choice of the penalty parameter ρ influences the accuracy and speed of convergence. The weaker ρ is, the more deviation is allowed between $\mathbf{z}_i$ and $\mathbf{z}$, and the less this deviation is penalized in the objective. If the gain in solution quality by evaluating expressions at $\mathbf{z}_i$ instead of $\mathbf{z}$ exceeds this penalty at the theoretical optimum, this can lead to potential infeasibility in the returned solution. This also explains how better evaluations $F(\cdot)$ can be found than would theoretically be possible from the centralized solution. However,

increasing ρ can slow down convergence for ADMM and the DFO methods, by tying $\mathbf{z}_a$ closer to $\mathbf{z}$ at each iteration or introducing ill-conditioning for model-based DFO methods respectively.

As a result, the conclusions on the relative convergence of the considered methods are influenced by ρ. There is ample literature on how ρ influences the convergence of ADMM [148]. There are multiple heuristics that can speed up ADMM, such as iteratively increasing ρ to allow for more exploration and faster convergence initially while encouraging fine-tuning and feasibility in later iterations [408]. However, we keep ρ constant across our algorithm benchmarking, since our analysis is based on comparing function evaluations. In ADMM, each iteration makes consistent progress towards a local minimizer. For the DFO methods however, we have no guarantee that the points we are sampling in later iterations present better function evaluations. By changing ρ between iterations, the same sample would, if resampled in a later iteration, potentially return a worse evaluation due to a higher ρ. Hence, from the perspective of DFO solvers, changing ρ between iterations would hinder convergence by introducing 'noise' or 'stochasticity'.

4.5 Conclusion

Our proposed 'data-driven' framework is able to find a solution close to the equivalent centralized formulation for optimization-based coordination problems in all our considered case studies. Our approach differs from ADMM in that it uses derivative-free optimization (DFO) to find the shared variables that minimize the sum of lower-level optimal subproblem objectives. We include CUATRO, Py-BOBYQA, DIRECT-L, and GPyOpt as DFO solvers, and benchmark them against ADMM as a distributed optimization solver on a motivating example and three case studies with expensive subproblems. We examine the effect that dimensionality and solution topology in the shared variables have on the relative algorithm performance. We also discuss organizational considerations and how they inform the choice of coordinating algorithm: autonomy and flexibility, privacy, software, black-box subproblems, and organizational structure. We show that our approach outperforms ADMM when the number of shared variables between agents is few, and when the shared variable to coordinated objective evaluations call for exploration rather than exploitation. As opposed to distributed optimization, our method does not need the capacity for numerical optimization at the agent-level, since the subproblems can also be evaluated as the result of black-box objective simulations. We argue that our approach is especially relevant when the decomposition is limited by organizational rather than numerical considerations.

While the relative performance of DFO algorithms is in line with current literature, there are several avenues for future work: the deployment of the suggested data-driven workflow would require a thorough investigation into privacy (differential privacy, cryptography) and into how much agent-level information can be inferred from optimal subproblem evaluations and variables. ADMM loses its convergence guarantees when the subproblems are ill-behaved. Consequently, we could test our intuition that the data-driven approach might have a competitive advantage when a) the subproblem evaluations are not solved to global optimality, b) objective evaluations are obtained by querying human decision-makers as opposed to optimization or simulation software, or c) evaluations are noisy, inconsistent, or non-stationary. Our framework could potentially even enable coordination between business units, where some decision-makers still use expert-guided heuristics rather than numerical optimization.

4.5.1 Looking back

The promise of machine learning, specifically supervised learning, lies in its ability to abstract complicated physical or mathematical phenomena into input-output mappings. The challenge then becomes finding the model family, architecture, and parameters that can best explain the data. The proposed framework of data-driven coordination follows a similar rationale: coordination in EWO can be subject to various organizational considerations that complicate the use of off-the-shelf tools for non-expert decision-makers. Thus, the idea of simplifying decision-making into input-output mappings, where complicating considerations are contained within the black-box, has creative merit.

However, throughout the course of this project, I realized I was overlooking the fundamental reasons why mathematical and operations research techniques are used in the first place: given a set of assumptions, these tools are designed to exploit mathematical structure for numerical efficiency. For example, while ADMM performance can be sensitive to the choice of the penalty parameter, it generally makes rapid progress toward the vicinity of a local minimizer. This makes ADMM hard to surpass in terms of sample efficiency. ADMM owes its superior scaling properties with respect to the number of shared variables to its exploitation of problem information in the form of subgradients in the coordination step.

Additionally, ADMM is only outperformed when assumptions are violated, when it gets stuck in local optima of nonconvex problems. While the proposal of data-driven coordination has merit, and exploitative DFO solvers do outperform ADMM in fine-tuning solutions, it is challenging to

find these kinds of problems with a manageable number of complicating variables at an industrially relevant scale. Finally, for a fair comparison, it would be more appropriate to also compare our framework to nonconvex implementations of ADMM, which might offer a more nuanced perspective on the relative strengths and weaknesses of each approach.

After this proof of concept, I approached my next challenge differently: given an intractable multi-level formulation, I wanted to identify the most suitable data-driven technique that would render the problem tractable. Instead of devising an interesting solution and then searching for use cases with industrial collaborators, I started from the problem itself and determined the best method to address it.

4

5

Hierarchical planning-scheduling-control

Contents

5.1 Introduction . . . 140
5.1.1 Relaxation or aggregation techniuqes . . . 141
5.1.2 Decomposition techniques . . . 141
5.1.3 The impact of data-driven techniques . . . 142
5.1.4 Our contribution . . . 142
5.1.5 Related works . . . 144
5.2 Case study . . . 148
5.2.1 Planning . . . 149
5.2.2 Scheduling . . . 149
5.2.3 Control . . . 150
5.2.4 Integration . . . 151
5.3 Methodology . . . 152
5.3.1 Using data to solve a canonical bi-level problem . . . 153
5.3.2 Hierarchical planning-scheduling-control as tri-level formulation . . . 156
5.3.3 Metrics . . . 157
5.3.4 Partitioning into DFO variables and lower-level scheduling-control problems 158
5.3.5 Methodology 1: Derivative-free optimization . . . 161
5.3.6 Methodology 2: Optimality surrogates . . . 162
5.3.7 Methodology 3: Combining both approaches . . . 163
5.3.8 Software and practical considerations . . . 163
5.4 Results and Discussion . . . 164
5.4.1 Methodology 1: Integration into planning via DFO . . . 165
5.4.2 Methodology 2: Integration into planning via surrogates . . . 167
5.4.3 Comparing the DFO and surrogate approaches . . . 171
5.4.4 Methodology 3: Bringing out the best in both approaches . . . 172
5.5 Conclusion . . . 176
5.5.1 Looking back . . . 177

In this chapter, we tackle the challenge of coordinating decisions between hierarchical agents involved in integrating planning, scheduling, and control. Accounting for lower-level feasibility or optimality in the integrated formulation results in generally intractable large-scale, potentially multi-level problems. We present a step-by-step workflow for using data-driven techniques, such as optimality surrogates and derivative-free optimization, to derive a tractable solution for a tri-level formulation of a multi-site, multi-product case study. Finally, we synthesize our findings into a methodology that combines the strengths of derivative-free optimization and optimality surrogates. Our method remains agnostic to level-specific formulations when linking variables are identified, and retains the heuristic sequential solution as a fallback option. This chapter, along with the associated Appendix C, is adapted from work published in [389]. Figures in this chapter are reused with permission under the author's rights in accordance with Elsevier's policies.

5.1 Introduction

In Section 1.1.2, we introduce Enterprise-wide Optimization (EWO) as crucial for companies to remain competitive in an environment of increasingly stringent safety, environmental, and economic requirements [150]. EWO has the ultimate goal to coordinate all decision-making within a company [152, 153]. Significant value can be captured by integrating units across all hierarchical levels of decision-making (from design, planning, scheduling, to control). We adopt the terminology of Chu and You [87] and distinguish between the sequential, monolithic, and hierarchical integration of optimization models as shown in Figure 5.1. Conventionally, decision-making happens *sequentially*: upper-level decisions are taken while disregarding lower-level considerations, and then fed as setpoints to the lower levels. However, as we already discuss in Section 1.2, there is no guarantee that these setpoints are feasible in lower-level problems. Sequential decisions historically arise out of necessity due to tractability and organizational constraints.

In the *monolithic* approach, lower-level feasibility considerations are included into upper-level optimization problems. This comes at the expense of a significant drop in computational tractability, due to heterogeneity in model formulations and vastly different time horizons between levels. Lower-level quantities can appear in the upper-level objective, but this does not guarantee lower-level optimality. While the monolithic approach could for example return planning setpoints that minimize total planning including changeover costs for a specific feasible scheduling assignment of said production, the scheduling layer might decide on another assignment that optimizes a different

objective such as makespan.

In the *hierarchical* approach, upper-level decision-makers consider not only lower-level feasibility, but also optimality, accounting for how lower-level decisions affect upper-level objectives. This formulation arises when closed-loop behavior of the system is considered or a game-theoretic approach is followed, and can be avoided if the integrated model is used in an MPC architecture [264, 159]. This results in multi-level formulations, which are numerically intractable and mathematically difficult [214]. There are two ways that we can alleviate the computational burden of integrated optimization problems: we either simplify the expensive, detailed formulations, or we keep the formulation and use more sophisticated solution techniques.

5.1.1 Relaxation or aggregation techniuqes

The first set of techniques [14] aim to relax the constraints, granularity, or detail of the integrated optimization models. This includes the replacement of some model parts - for example the lower level - by surrogate models [379, 376, 85] that are easier-to-handle by numerical solvers. This concept is inherently linked to the model reduction techniques discussed in Section 2.3.3. Introducing surrogates into the original formulation inevitably incurs the risk of losing solution quality. Therefore, decision-makers inherently trade-off surrogate accuracy with optimization tractability when they choose the type and size of surrogate.

5.1.2 Decomposition techniques

The second approach exploits the mathematical structure of integrated decision-making problems via decomposition algorithms [288, 266]. Hierarchical problems could be viewed as coordination problems between one or multiple leader(s) and follower(s): each player has decision freedom over their respective problem but needs to coordinate on a sparse set of complicating variables. Traditional sequential decision architectures often lead to the coordinated variables to be few relative to the upper- or lower-level specific variables. As a result, Lagrangean, and Benders decomposition [207, 190] lend themselves well to tractably solving hierarchically integrated problems.

5.1.3 The impact of data-driven techniques

The interest and successes in Machine Learning (ML) over the last decade have successfully diffused through the chemical engineering literature [335, 270]. These advances have also inspired new developments in hierarchical planning, scheduling, and control. While surrogates - also known as meta-, scale-bridging, or reduced-order models [396, 118, 379] - have been established tools in the process systems engineering literature (Section 2.3.3), ML has shifted attention to other approximation models [38]: artificial neural networks, (linear model) decision trees, Gaussian Processes, and support vector machines. We use 'feasibility surrogates' and 'optimality surrogates' as umbrella terms for any approximation model with the aim of mapping lower-level feasibility, or lower-level optimal response variables (objective or decision variables) of upper-level complicating variables or 'setpoints'. As discussed at length in Section 2.3.4, this approach has analogues and similarities across many disciplines, from multi-parametric model predictive control and explicit optimal policies [22], to amortized learning [13] in ML, and value functions [321] in Reinforcement Learning (RL).

The developments in ML have also heavily influenced the use of surrogates in derivative-free optimization (DFO) [228]. DFO deals with the optimization of systems without having explicit or cheap access to gradient information. Although there are subtle differences, here, we use the term DFO synonymously with black-box, simulation-based, zeroth-order, or gradient-free optimization [12]. In Chapters 3 and 4, we discuss the application of DFO to traditional chemical engineering problems, and to enterprise-wide optimization (EWO) coordination problems that traditionally are targeted via decomposition algorithms, respectively. Similarly, DFO has been applied to the solution of multi-level problems in process systems engineering (PSE) [422] and even to the hierarchical integration of process operations [37]. For more background information on DFO solvers, interested readers are referred to Section 3.1.1.

5.1.4 Our contribution

In Section 5.1.5, we review the broad body of literature related to data-driven solution methods for integrated planning-scheduling-control problems. In earlier conference proceedings, we already attempt a tri-level solution of the integrated Planning-Scheduling-Control (iPSC) problem by using DFO to optimize planning simulations, where lower-level information is accounted for via calls to the integrated scheduling layer with embedded control surrogates. In the remainder of this chapter,

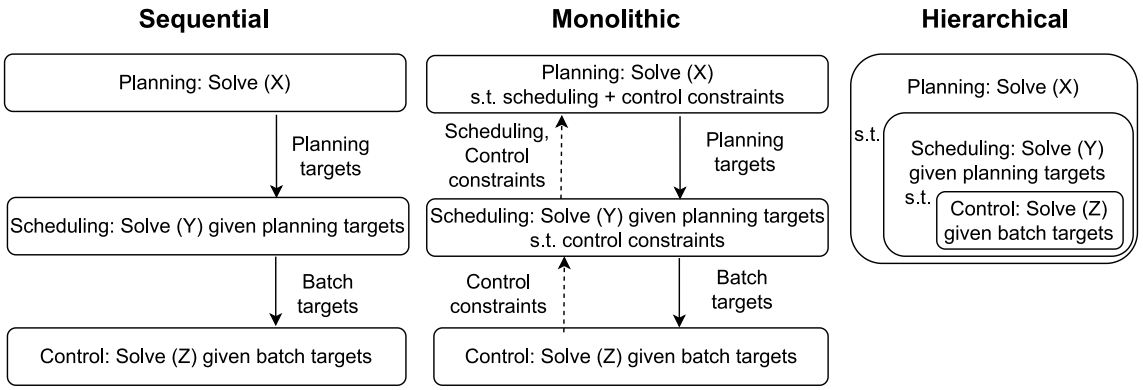


Figure 5.1: Hierarchical decision-making architectures: In the sequential approach, higher-level solutions feed into lower levels as setpoints. Each level solves their own optimization problem to meet these setpoints without explicit considerations of the lower levels. The monolithic approach resembles the sequential approach, but feasibility in lower levels is satisfied via explicit incorporation of their constraints. The hierarchical approach could be viewed as a multi-level leader-follower game. A single 'tri-level' optimization problem is solved, where the planning is subject to the optimal solution of the scheduling level, and the scheduling is constrained on the optimal solution of the control level.

we strive to answer the questions that are left unanswered. We provide three methodologies that outperform the sequential approach on a small-but-realistic iPSC formulation of the value chain motivating hierarchical integration in Section 1.2:

- We provide a tutorial-like exposition of the DFO and surrogate approaches and how we can evaluate solution qualities based on increasingly complex single- to tri-level evaluations.
- We investigate if we can integrate the scheduling-control layers as surrogates into the planning and if this surrogate should be trained on scheduling-only, approximate scheduling-control, or exact scheduling-control data.
- We investigate if we can use DFO to find the tri-level solution without relying on model approximations.
- We explore practical techniques for balancing solution accuracy and computational time through combining the two methods.

After reviewing iPSC solution methods in Section 5.1.5, we illustrate our multi-site, multi-product, hierarchical planning-scheduling-control case study in Section 5.2. In Section 5.3, we present in tutorial fashion three combinations of the DFO and optimality surrogate approaches that can be leveraged towards a tractable tri-level planning-scheduling-control solution of our problem. We also highlight practical considerations that are crucial to making these approaches scalable. In Section 5.4, we present our results on the tractability-accuracy trade-off between the

different methods on multiple solution quality and solution time metrics. We emphasize the role that understanding the interplay of ML and optimization pipelines has on navigating the inherent accuracy-tractability trade-off before concluding in Section 5.5.

5.1.5 Related works

There is ample review literature about solution strategies and applications in the general domain of integrated planning-scheduling-control (iPSC). Some works narrow in on the integration of scheduling and control (iSC) [102], others focus on uncertainty [112], their application to sequential batch processes [86], or the difference between top-down and bottom-up approaches [26, 74]. Andres-Martinez and Ricardez-Sandoval [14] present the most recent review at the time of writing and classify works according to their application to integrated planning and scheduling (iPS), integrated scheduling and control (iSC), or iPSC. While most reviews are structured following the scope of integration, we are more interested in coming up with a typology of solution approaches. We discuss different solution methods and highlight their relationship to the general paradigms of surrogate modelling, decomposition, and derivative-free optimization (DFO). While we initially zone in on monolithic or simultaneous formulations, we argue that the same typology of solution approaches can be applied to hierarchical, multi-level formulations.

Surrogates

We use the term 'surrogate models' to encompass all approximation models that reduce model complexity for the sake of tractability (Section 2.3.3). Surrogates could be loosely regarded as relaxation or aggregation techniques and as such inevitably incur the risk of losing solution quality by forfeiting model accuracy. Tsay and Baldea [376] argue that data presents a natural bridge between different hierarchical layers and consequently advocate for the use of data-driven 'time-scale bridging models' to integrate operations and control. Scale-bridging models are compatible with the 'aggregate method' [85], where surrogates are used to map the linking functions between subproblems arising from the decomposition of different hierachical layers.

Surrogates differ by the amount of detail they capture from lower levels. Consequently, Tsay and Baldea [376] distinguish between static and dynamic models. Static surrogates for instance do not explicitly account for transient data or dynamics. They could for example be used to learn only scheduling-relevant quantities from the control problem in iSC [118]. They can also be used to map the feasible space of lower levels [407], which is related to the concept of feasibility analysis

[24]. Convex region surrogates are especially popular as they use historical operating data to map feasible sets as mixed-integer linear programming (MILP) constraints, avoiding mixed-integer nonlinear formulations. Nonlinear regression techniques could also be used as static models in mapping continuous outputs rather than binary feasibility: Gaussian Processes, support vector machines, piece-wise affine regression, or artificial neural networks would be well-suited to map the (optimal) cost of a lower level.

Dynamic surrogates explicitly account for time as an additional input. They are necessary when the dynamics of the lower level become relevant. Data-driven dynamic surrogates are overwhelmingly influenced by the system identification literature [208, 378, 380]. Santander et al. [324, 323] argue that deep learning follows similar workflows to linear state identification methods, captures nonlinearities, and reduces computational expense when integrating advanced process control with production planning or scheduling respectively. The difference between learning dynamic surrogates or only response variables relevant to the upper level divides the community: The system identification literature champions the learning of dynamic models from data which can subsequently be deployed within optimization-based feedback control such as model predictive control (MPC) [55]; The RL literature advocates for the approximation of optimal policies or feedback laws directly from data. Rawlings and Maravelias [314] argue that Q-Learning is not competitive with standard system identification on a simple, linear system in the presence of small amounts of Gaussian noise. However, they find promise in the use of neural networks to learn the response of an MPC controller on a linear dynamic model with quadratic stage cost.

Decomposition

Planning, scheduling, and control problems are often tackled by different business units within the same company, since they cover widely different timespans and types of decisions. As a result, the interactions between hierarchical decision-making layers, also called linking or complicating variables, naturally end up being sparse. Examples include planning targets and batch processing times in iPS or iSC problems respectively. When moving away from sequential to monolithic or even hierarchical integrated problem formulations, this gives rise to mathematical optimization structures that can be exploited by decomposition techniques [288]. While these techniques describe a wide range of approaches, they typically consist of solving problems by breaking them up into smaller subproblems, which are solved in parallel or sequentially. This often involves many iterations over the subproblems coordinated by the solution of a 'master problem'.

Lagrangean decomposition can be used to break up the iPSC problem into separate, more manageable iPS and control problems [266]. The coupling variables can even be learned via systematic network approaches. Mitrai and Daoutidis [261] use community detection to find the mathematical structure in an iSC problem and the linking variables that can be exploited using generalized Benders decomposition. As expected, they recover the scheduling problem in the first level and dynamic optimization subproblems in the second level. Similarly, Mitrai and Daoutidis [262] use nested stochastic blockmodelling to uncover different types of complicating variables and structures that arise at the different hierarchical levels. In later work, Mitrai and Daoutidis [263] use Multi-cut generalized Benders decomposition for iPSC by learning the problem structure and find that the planning-scheduling and control constraints are coupled only via the transition times.

Decomposition can also be combined with surrogates. Kelley et al. [207] use Hammerstein-Wiener models to integrate control dynamics into scheduling and solve the resulting formulation using Lagrangean decomposition.

Derivative-free optimization

Derivative-free optimization (DFO) is used to optimize systems without explicit gradient expressions [228], bypassing the need for explicit model formulations. DFO leverages input-output evaluations, or 'data', directly to find the optimum of the problem at hand through direct or model-based methods. Direct methods comprise of a wide range of techniques from random search, to grid search, and more sophisticated algorithms like NOMAD [229], or even evolutionary methods [352]. Developments in model-based DFO are closely related to surrogate techniques, as they rely on the intermediate construction and optimization of approximation models to find the optimum. Popular methods range from exploitative trust region methods [306, 307] to more explorative Bayesian optimization using Gaussian processes [134] or gradient-boosted trees [370]. For a thorough classification of different DFO techniques, the reader is referred to Section 3.1.1.

DFO can exploit the same mathematical problem structure as decomposition approaches in iPSC. DFO approaches are especially prevalent in iPS, when the number of complicating variables (i.e. the planning targets) is few. Early attempts mostly use genetic algorithms [425, 241] or surrogates/metamodels [396] under the name of simulation-based optimization [395]. Hybrid DFO and decomposition approaches also exist. Ji and Gu [190] use a combination of generalized Benders decomposition and genetic algorithms in the simultaneous solution of iSC.

DFO and single-level reformulation for multi-level problems

Accounting for game-theoretical considerations in the optimization of integrated supply chains and process systems often involves multi-level formulations [250]. While sometimes simultaneous formulations for iPSC can be solved without resorting to decomposition, surrogates, or other techniques meant to alleviate computational expense [158], the nested decision-making architectures of bi-level formulations require specialized approaches. Colson et al. [93] discuss general bi-level optimization solution approaches. If no suitable decomposition technique is available, solving multi-level problems typically requires their reformulation into a form compatible with conventional single-level optimization. These techniques can be categorized into two main approaches: DFO and single-level reformulation.

DFO extends to hierarchical and bi-level formulations by using data-driven optimization to find optimal planning or scheduling setpoints subject to the optimal response of the scheduling [37] or control level [113] respectively. On the other hand, single-level reformulation involves transforming the lower-level optimization problem into its Karush-Kuhn-Tucker (KKT) conditions. These can then be embedded into the upper level as big-M constraints, or by using branching techniques on the complementarity conditions [213]. KKT reformulation has been used to embed linear MPC into scheduling [348], and operations into supply chains design [414]. KKT reformulation is already computationally expensive even for linear cases and generally loses any theoretical guarantees in nonconvex iPSC.

Surrogates for multi-level problems

Rather than embed lower levels into the upper level using KKT reformulations, we can 'learn' the optimal lower-level or follower response as a function of the leader's variables [265]. This could be considered an extension of the surrogate approach: rather than learn lower-level *feasibility* from historical operating data, we can learn lower-level *optimality* from the optimal solution of scheduling or control problems. We call 'feasibility surrogates' and 'optimality surrogates' the use of time-scale bridging models applied to simultaneous and hierarchical approaches respectively (Section 2.3.3). The line between monolithic and hierarchical formulations can be blurry in feasibility analysis [24] and often boils down to practical details - if feasible operating data is available or needs to be obtained via solving feasibility problems.

The application of 'optimality surrogates' in the integration of design, operations, and con-

trol appears in different domains under various names, from 'response functions' [84], to multi-parametric programming [21, 61, 303]. More recently, successes in Reinforcement Learning and Machine Learning have introduced new terminology and different perspectives. Sachio et al. [321] use Reinforcement Learning to learn an optimal policy on MPC simulations before embedding into the design problem. Software advances such as OMLT [381, 77, 76] are streamlining the integration of optimization and ML pipelines by automating reformulations of neural networks and decision trees and their subsequent embedding into Pyomo as constraint blocks that can be handled by MILP solvers. This could catalyze new advances in the safe deployment of learnt optimality surrogates to mitigate model approximation risks, like adversarial approaches or safe Reinforcement Learning [333, 62].

5.2 Case study

In this section, building on the motivating example from Section 1.2, we present the planning, scheduling, and control layers from our multi-site, multi-product problem as shown in Figure 5.2. We then discuss how the decisions between the levels are interlinked with the help of Figure 5.3. We refer to Section C.1 for detailed optimization formulations of the high-level planning, scheduling, and control formulations presented in (5.1), (5.2), and (5.3). The implementations of the planning, scheduling, and control formulations can also be found in the Github repository under https://github.com/OptiMaL-PSE-Lab/DD-Hierarchical.

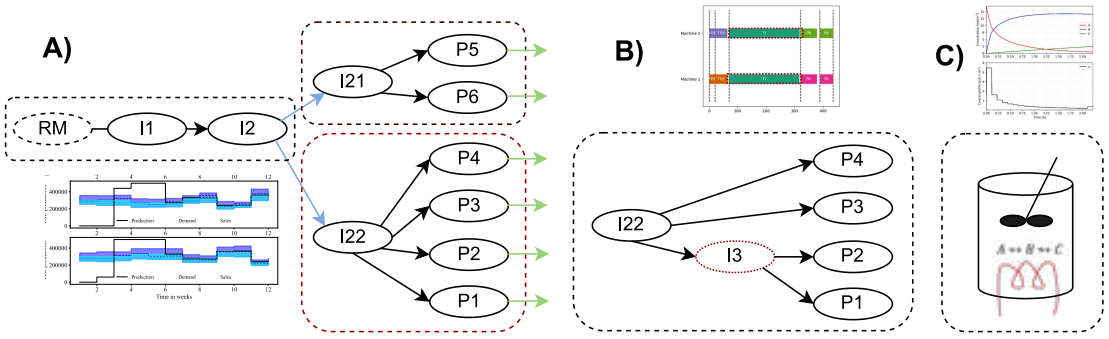


Figure 5.2: Outline of the hierarchical A) planning, B) scheduling, C) and control levels in the multi-site, multi-product case study. A) shows the planning state network denoting the materials (○) as well as processing (→), transport (→), and sales (→) across the three sites (□). Scheduling and control are only relevant to the third site. B) shows the state network of the scheduling problem that includes an additional intermediate material. The production of I3 and P1-P4 needs to be scheduled in batches on two machines in seven event time points. C) Each batch production is then subject to an optimal control problem.

5.2.1 Planning

The highest level of decision-making in our case study is planning. While planning model formulations can vary, production planning usually involves the solution of a (mixed-integer) linear program (LP or MILP) to determine the optimal production targets that optimize an economic objective over a specific planning horizon. Our planning problem (5.1) involves the solution of an LP over a planning horizon of 12 months. The state network of the planning is illustrated in Figure 5.2A, and is essentially identical to the motivating value chain from Figure 1.1. We consider three sites: In the first site, raw material RM (assumed unlimited) undergoes two intermediate processing stages (I1 and I2). I2 is then transported to the second and third sites, where it is referred to as I21 and I22 respectively. Both intermediates undergo further processing into one of two or one of four products (P5-P6 and P1-P4) respectively, which are then sold to meet external customer demand. The goal of the planning is to determine the production of each material given processing yields (black arrows), the material transport given lead times (blue arrows), and the sales given customer demand (green arrows) that maximize sales and minimize transportation, storage, and production cost. The LP is subject to inventory material balances on each material, site-wide resource utilization limits, and upper and lower bounds on inventory for each material. The full formulation can be found in (C.1).

$$
\begin{aligned}
\min. \quad & \text{Transport + Storage + Production - Sales} \\
\text{s.t.} \quad & \text{Initial conditions} \\
& \text{Inventory mass balances with production and transportation} \\
& \text{Resource limits} \\
& \text{Sales limits} \\
& \text{Safe storage constraints}
\end{aligned}
\tag{5.1}
$$

5.2.2 Scheduling

The optimal planning targets for each of the 12 planning timesteps are then fed to the scheduling layer. At each planning timestep, a scheduling problem determines the resource allocation required to reach the planning targets in minimal time, i.e. for each of the 7 scheduling timesteps (events), for each of the two machines, which production (job) happens and for how long. Our scheduling formulation is based on Maravelias and Grossmann [249], and relies on the state-task network ap-

proach with a common continuous-time representation for all units. The formulation also accounts for variable batch sizes, variable processing times and sequence-dependent changeover times. In our integrated framework, the scheduling layer involves the solution of a MILP (5.2) at each planning step with discrete variables in the assignment constraints (if a batch production happens in a specific machine at a given time) and changeover constraints (if we switch from one production to another at a given event time point and machine) on top of the planning-level mass balances. The batch constraints, production recipes, production duration, and changeover duration are machine- and sequence-specific. There is also a more accurate estimate of the inventory and resource limits available compared with the planning layer. The scheduling layer is only considered for site three given the simplicity of the state network in the other sites. Explicitly accounting for scheduling in the other sites is compatible with our proposed methodologies at a potential increase in solution time, especially since the scheduling problems are only loosely connected via few planning variables. The state-task network corresponding to site 2 is depicted in Figure 5.2B, where on top of the state network from Figure 5.2A, we also consider a perishable, bottleneck intermediate I3 between I22 and products P1 and P2. I3 is produced on-demand for each planning timestep with any unused production after the planning period going to waste. The full formulations can be found in (C.2) to (C.6).

$$
\begin{aligned}
\min.\quad & \text{Makespan} \\
\text{s.t.}\quad & \text{Inventory mass balances with production} \\
& \text{Batch constraints} \\
& \text{Duration constraints} \\
& \text{Assignment constraints} \\
& \text{Changeover constraints} \\
& \text{Resource limits} \\
& \text{Meet monthly planning production targets}
\end{aligned}
\tag{5.2}
$$

5.2.3 Control

The control formulations are based on Mishra et al. [260]. Each batch control takes the batch targets for each job-machine allocation as determined in the scheduling layer, and determines the optimal cooling flowrate that achieves said batch target while minimizing a combination of processing time and energy cost. This involves the solution of an optimal control problem (5.3)

subject to quality conditions and nonlinear differential expressions of the kinetics and energy transfer. The full formulation can be found in (C.7).

$$\begin{aligned} \min. \quad & \text{Processing time \& energy cost} \\ \text{s.t.} \quad & \text{Mass balances given by differential equations} \\ & \text{Kinetics dependent on state and control variables} \\ & \text{Energy cost dependent on control variables} \\ & \text{Final time quality conditions} \end{aligned} \tag{5.3}$$

5.2.4 Integration

iPSC problems are either solved sequentially by ignoring all lower-level considerations, monolithically by only accounting for lower-level feasibility, or hierarchically by explicitly accounting for lower-level optimality as depicted in Figure 5.1. While the sequential and monolithic approaches are often employed for tractability or practical reasons, the hierarchical approach most accurately reflects organizational decision-making. Figure 5.3 shows the information flow between the layers. These connecting, coordinated, or complicating variables link the three decision-making units that can otherwise be solved autonomously by each level. At the highest level, the 12 optimal monthly planning variables feed as setpoints into separate scheduling problems. For each planning timestep, all batch assignment targets as determined in the scheduling layer then feed separately as setpoints into the control level. In the scheduling layer, we need to decide on and consequently explicitly consider all 5 possible productions for each of the 14 (7 events by 2 machines) optimal control problems. Each optimal processing time from the control problems feeds back into the scheduling, correcting the makespan corresponding to a given assignment sequence; and the optimal changeover and cooling costs from the scheduling and control problems feed back into the planning, correcting the economic objective.

A hierarchical solution is not just intractable but mathematically difficult given the nested layers of decision-making. Each tri-level optimization requires the solution of up to 840 control problems: the top layer (planning) is subject to the optimal solution of 12 scheduling optimization problems that themselves are constrained on the optimal solution of up to 70 optimal control problems each. Even the monolithic formulation remains intractable given the embedding of 12 sets of mixed-integer *nonlinear* scheduling constraints containing the dynamics of 70 sets of batch production dynamics.

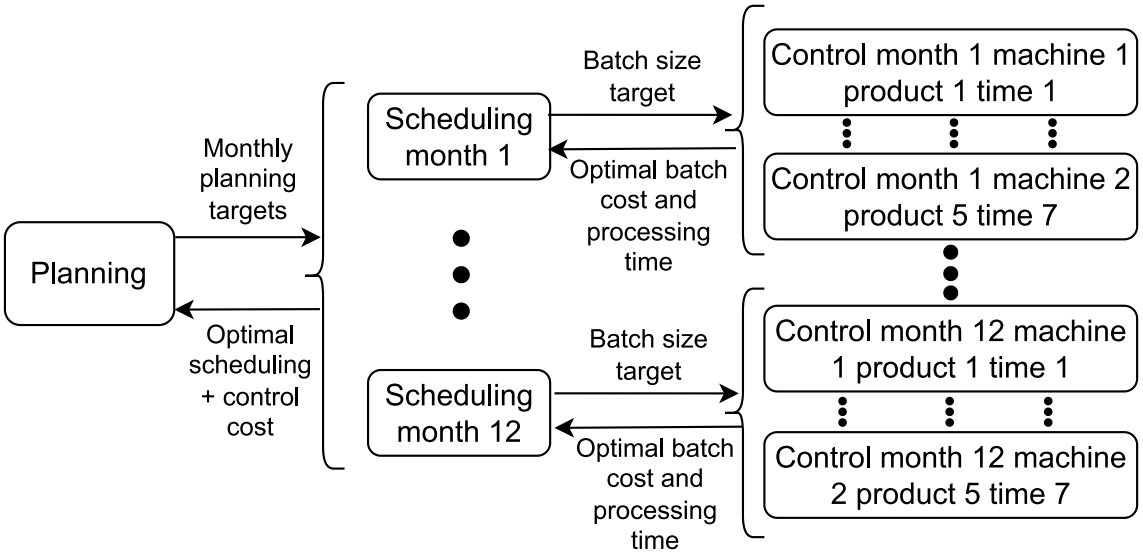


Figure 5.3: Linking variables between the planning, scheduling, and control levels. Each planning problem feeds the planning targets for its 12 timesteps to separate scheduling problems to be solved in parallel. Each of the 12 scheduling problems, in deciding which of the 5 productions to schedule on the 2 machines at the 7 events, obtains the optimal batch processing time and cost as a function of the batch size targets for each of the 70 possible batch assignments. The optimal scheduling and control finally feed back into the planning.

Data-driven techniques have become essential to the solution of iPSC formulations. In the next section, we show how we can exploit data-driven techniques to solve hierarchical (multi-level) formulations. While it is unrealistic to expect to find the optimal solution to tri-level optimization formulations (even the simplest linear-linear bilevel optimization problems are NP-hard [214]), our aim is to find a tractable solution to the tri-level hierarchical formulation that outperforms the conventional sequential solution. In this work, we prioritize finding an approximate solution to an accurate model over finding the optimal solution to an approximate model. We acknowledge however that there are many reasons to use the sequential or monolithic approach, i.e. when the lower-level objectives are aligned with upper-level objectives, or when any potential gain in the tri-level formulation does not justify the increase in computational effort.

5.3 Methodology

To streamline the exposition of how we can use surrogates and derivative-free optimization (DFO) to solve our iPSC problem, we first show how we can use both techniques in solving general bi-level problems.

5.3.1 Using data to solve a canonical bi-level problem

Problem (5.4) presents a general bi-level formulation where the leader determines the set of variables that appear only in their level $\mathbf{x}_{up}$, and the connecting variables $\mathbf{x}_{u\to l}$ that appear as setpoints in the lower level. Given the linking variables $\mathbf{x}_{u\to l}$, the follower optimizes their objective $f_{low}(\cdot)$ by manipulating the variables that are specific to the lower level $\mathbf{x}_{low}$ and the connecting variables that also appear in the upper level $\mathbf{x}_{l\to u}$. Therefore, the leader optimizes their objective $f_{up}(\cdot)$ while explicitly accounting for the optimal response $\mathbf{x}^*_{l\to u}$ of the follower to $\mathbf{x}_{u\to l}$. We consider only continuous instances of the complicating, also called linking or coordinated, variables $\mathbf{x}_{u\to l}$ and $\mathbf{x}_{l\to u}$, even though no such restrictions need to be placed.

$$
\begin{aligned}
\min_{\substack{\mathbf{x}_{up},\mathbf{x}_{u\to l},\\ \mathbf{x}^*_{low},\mathbf{x}^*_{l\to u}}} \quad & f_{up}(\mathbf{x}_{up},\mathbf{x}_{u\to l},\mathbf{x}^*_{l\to u}) && && \text{(5.4a)}\\
\text{s.t.} \quad & \mathbf{h}_{up}(\mathbf{x}_{up},\mathbf{x}_{u\to l},\mathbf{x}^*_{l\to u}) = \mathbf{0} && && \text{(5.4b)}\\
& \mathbf{g}_{up}(\mathbf{x}_{up},\mathbf{x}_{u\to l},\mathbf{x}^*_{l\to u}) \le \mathbf{0} && && \text{(5.4c)}\\
& \mathbf{x}^*_{low},\mathbf{x}^*_{l\to u} \in \underset{\mathbf{x}_{low},\mathbf{x}_{l\to u}}{\arg\min}. \quad & f_{low}(\mathbf{x}_{low},\mathbf{x}_{l\to u},\mathbf{x}_{u\to l}) && && \text{(5.4d)}\\
& \qquad\qquad \text{s.t.} \quad & \mathbf{h}_{low}(\mathbf{x}_{low},\mathbf{x}_{l\to u},\mathbf{x}_{u\to l}) = \mathbf{0} && &&\\
& \qquad\qquad & \mathbf{g}_{low}(\mathbf{x}_{low},\mathbf{x}_{l\to u},\mathbf{x}_{u\to l}) \le \mathbf{0} && &&
\end{aligned}
$$

We now present two different ways that data-driven techniques can be exploited towards a solution of bi-level problems.

Optimality surrogates

We first present optimality surrogates as a solution approach to multi-level problems. Crucial to solving bi-level optimization problems is their reformulation into a single level, such that the problem can be exploited using standard optimization solvers. In the optimality surrogate approach, we first sample various combinations of the complicating variables $\mathbf{x}_{u\to l}$ and solve the corresponding lower-level optimization problem to extract the optimal response $\mathbf{x}^*_{l\to u}$. Then, we can use any supervised learning technique to construct a model - we use neural networks - to map $\mathbf{x}^*_{l\to u}$ to $\mathbf{x}_{u\to l}$. The explicit expression $\mathbf{x}^*_{l\to u} = \mathcal{NN}(\mathbf{x}_{u\to l})$ can then be embedded as a constraint into the upper level, replacing the lower-level optimization problem, and collapsing the bi-level into a single-level formulation (5.5) that can be readily implemented in standard optimization software.

$$
\begin{aligned}
\min_{\substack{\mathbf{x}_{up},\mathbf{x}_{u\to l},\\ \mathbf{x}^*_{l\to u}}} \quad & f_{up}(\mathbf{x}_{up},\mathbf{x}_{u\to l},\mathbf{x}^*_{l\to u}) \\
\text{s.t.} \quad & \mathbf{h}_{up}(\mathbf{x}_{up},\mathbf{x}_{u\to l},\mathbf{x}^*_{l\to u}) = \mathbf{0} \\
& \mathbf{g}_{up}(\mathbf{x}_{up},\mathbf{x}_{u\to l},\mathbf{x}^*_{l\to u}) \leq \mathbf{0} \\
& \mathbf{x}^*_{l\to u} = \mathcal{NN}(\mathbf{x}_{u\to l}) \qquad \text{(neural network)}.
\end{aligned} \tag{5.5}
$$

In principle, it is possible to construct surrogates that present exact 'explicit control laws' for certain kinds of parameterized optimization problems [315]. However, parametric programming explodes in complexity with the size of the lower levels [304]. Consequently, we train approximate optimality surrogates, forfeiting theoretical guarantees that the lower-level is solved to optimality. Therefore, care should be taken in choosing the type and architecture of the surrogate when trading off solution accuracy and tractability.

We suggest using neural networks with piecewise linear activation functions (ReLU) as optimality surrogates since their expressions can be reformulated into mixed-integer *linear* constraints [182], avoiding the introduction of nonlinear terms into upper-level objectives. Other techniques like decision trees could be used to the same end. There is ongoing debate whether it is favourable to use mixed-integer linear formulations with discrete solvers or to use global solvers on the full-space formulations of the surrogates [333].

The optimality surrogate approach mirrors the way 'feasibility' surrogates would be used. However, in monolithic formulations (binary) data on the feasible operating region would be used to map and embed constraints as $\mathcal{NN}(\mathbf{x}_{u\to l}) \leq 0$.

Derivative-free optimization

In our second approach, we leverage recent advances in derivative-free optimization (DFO). The idea of using derivative-free optimization (DFO) for bi-level formulations is to fix as many upper-level variables as necessary such that the upper level becomes fully determined. First, we observe that when $\mathbf{x}_{u\to l}$ is fixed, we can obtain $\mathbf{x}^*_{l\to u}$ by solving the lower-level optimization problem. On top of this, we can split the upper-level variables $\mathbf{x}_{up}$ into a set of decision variables $\mathbf{x}_{DFO}$ and $\mathbf{x}_{sim}$, such that $\mathbf{x}_{sim}$ can be obtained by solving the system of equality constraints $\mathbf{h}_{up}(\cdot) = \mathbf{0}$ with $\mathbf{x}_{DFO}$ and $\mathbf{x}_{u\to l}$ fixed. This means that we can use DFO to find $\mathbf{x}_{u\to l}$ and $\mathbf{x}_{DFO}$ that minimize the black-box objective consisting of the upper-level objective $f_{up}(\cdot)$ augmented by a penalization of any violation of $\mathbf{g}_{up}(\cdot) \leq \mathbf{0}$. The extent of penalization can be tuned via the penalty parameter

ρ. We essentially solve the general bi-level formulation (5.4) as a single-level DFO problem (5.6), where each evaluation consists in the expensive solution of the lower-level optimization problem $\mathcal{LOW}(\cdot)$ equivalent to (5.4d):

$$
\begin{aligned}
\underset{\mathbf{x}_{DFO},\mathbf{x}_{u\to l}}{\min.} \quad & \mathcal{BB}(\mathbf{x}_{DFO},\mathbf{x}_{u\to l}) && \text{(Black-box)} \\
\text{where} \quad & \mathbf{x}^*_{l\to u} \leftarrow \mathcal{LOW}(\mathbf{x}_{u\to l}) && \text{(lower problem)} \\
& \mathbf{x}_{sim} \leftarrow \mathbf{h}_{up}(\mathbf{x}_{sim},\mathbf{x}_{DFO},\mathbf{x}_{u\to l},\mathbf{x}^*_{l\to u}) = \mathbf{0} \\
& \mathbf{x}_{up} = [\mathbf{x}_{sim},\mathbf{x}_{DFO}] \\
& \text{penalty} = \rho||\max(\mathbf{0},\mathbf{g}_{up}(\mathbf{x}_{up},\mathbf{x}_{u\to l},\mathbf{x}^*_{l\to u}))||^2 \\
& \mathcal{BB}(\mathbf{x}_{DFO},\mathbf{x}_{u\to l}) = f_{up}(\mathbf{x}_{up},\mathbf{x}_{u\to l},\mathbf{x}^*_{l\to u}) + \text{penalty}.
\end{aligned}
\tag{5.6}
$$

The left arrows $\leftarrow$ denote that we obtain $\mathbf{x}_{sim}$ and $\mathbf{x}^*_{l\to u}$ from the solution of the set of equality constraints $\mathbf{h}_{up}(\cdot) = \mathbf{0}$ and of the lower-level optimization instance $\mathcal{LOW}(\cdot)$ within the black-box simulation respectively. In principle, we can use any kind of DFO solver for the single-level DFO problem (5.6). However, our DFO problem is considered expensive and high-dimensional, which limits the practical applicability of many over-explorative methods like evolutionary search, and Bayesian optimization. We suggest using exploitative trust-region methods and leveraging as much problem knowledge as possible in finding a good initial guess.

Differences between the two methods

Model-based DFO and the surrogate approach could be easily confused as they both rely on surrogates. The surrogate approach constructs relevant input-output models of the lower level before embedding these surrogates into the upper level, where a single optimization formulation is solved using algebraic modelling languages such as Pyomo [164] or GAMS [98]. In model-based DFO, surrogates are only used in the 'outer level' to determine the next DFO candidate to sample, and could be replaced by direct DFO routines. In the 'inner level', the simulation calls the original lower-level optimization problem (formulated in Pyomo) as a black-box to return relevant quantities to the upper level.

More importantly, both approaches present vastly different model accuracy versus solution tractability trade-offs. While the surrogate approach inevitably runs the risk of losing solution quality through model inaccuracies, DFO scales poorly with the expense of lower-level problems and the number of DFO dimensions. This motivates a careful investigation into the intricacies of

our case study and how these can be used to integrate the planning-scheduling and the scheduling-control layer.

5.3.2 Hierarchical planning-scheduling-control as tri-level formulation

Problem (5.7) formalizes the hierarchical iPSC formulation described in Section 5.2.4. The upper level consists of an LP formulation to determine the planning-specific and complicating planning target variables ($\mathbf{x}_p$ and $\mathbf{x}_{prod}$) that minimize the planning-level economic objective $f_p(\cdot)$ augmented by the optimal scheduling and control costs c_s^* and c_c^*. c_s^* is obtained by constraining the upper level on solving 12 MILP scheduling problems. Their aim is to find the scheduling-specific and batch target variables ($\mathbf{x}_s$ and $\mathbf{x}_{batch}$) that minimize the makespan required to fulfil $\mathbf{x}_{prod}$. The optimal batch processing times t_f^* however rely on optimal control operation. Each of the 12 scheduling problems are subject to 70 nonlinear optimal control problems, where the control-specific variables $\mathbf{x}_c$ and processing times t_f are determined that minimize a mixture of energy cost and processing time $f_c(\cdot)$ while meeting the batch target setpoints ($\mathbf{x}_{batch}$):

$$
\begin{aligned}
&\underset{\substack{\mathbf{x}_p,\mathbf{x}_{prod},\\ \mathbf{x}_s^*,c_s^*,\mathbf{x}_b^*,\\ c_c^*,t^*,\mathbf{x}_c^*}}{\text{min.}} && f_p(\mathbf{x}_p,\mathbf{x}_{prod}) + c_s^* + c_c^* \\
&\text{s.t.} && \mathbf{h}_p(\mathbf{x}_p,\mathbf{x}_{prod}) = \mathbf{0} \\
& && \mathbf{g}_p(\mathbf{x}_p,\mathbf{x}_{prod}) \leq \mathbf{0} \\
& && \begin{matrix}\mathbf{x}_s^*,\mathbf{x}_b^*,c_s^*,\\ \mathbf{x}_c^*,t^*,c_c^*\end{matrix} \in \underset{\substack{\mathbf{x}_s,\mathbf{x}_{batch},c_s,\\ \mathbf{x}_{ctrl}^*,t_f^*,c_{ctrl}^*}}{\arg\min.} && f_s(\mathbf{x}_s,\mathbf{x}_{batch},t_f^*,\mathbf{x}_{prod}) \\
& && \text{s.t.} && \mathbf{h}_s(\mathbf{x}_s,\mathbf{x}_{batch},c_s,t_f^*,\mathbf{x}_{prod}) = \mathbf{0} \\
& && && \mathbf{g}_s(\mathbf{x}_s,\mathbf{x}_{batch},t_f^*,\mathbf{x}_{prod}) \leq \mathbf{0} \\
& && && x_{ctrl}^*,c_{ctrl}^*,t_f^* \in \underset{\mathbf{x}_c,c_{ctrl},t_f}{\arg\min.} && f_c(\mathbf{x}_c,t_f,\mathbf{x}_{batch}) \\
& && && && \text{s.t.} && \mathbf{h}_c(\mathbf{x}_c,t_f,c_{ctrl},\mathbf{x}_{batch}) = \mathbf{0} \\
& && && && && \mathbf{g}_c(\mathbf{x}_c,t_f,\mathbf{x}_{batch}) \leq \mathbf{0}.
\end{aligned}
\tag{5.7}
$$

$\mathbf{x}_p, \mathbf{x}_s, \mathbf{x}_c$ constitute the level-specific variables, and $\mathbf{x}_{prod}, \mathbf{x}_{batch}$ the complicating variables feeding as setpoints into lower levels, and $c_s^*, c_c^*, \mathbf{t}_f^*$ define the optimal responses of interest to upper levels. All other variables are level-specific versions of these quantities, introduced for the purpose of maintaining consistency and organization across different levels of the model. Before we de-

scribe how we can apply the techniques introduced on the general bi-level formulation (5.3.1), we first determine four metrics by which we can assess the quality of a planning-level solution with increasing levels of complexity and accuracy.

5.3.3 Metrics

To compare the solution quality of the proposed solution approaches, we define four different solution metrics. Figure 5.4 illustrates the differences between the four solution quality evaluation metrics accounting for different levels of lower-level information integration into the upper-level planning objective (5.8). In each case, we start by extracting planning-level variables $\mathbf{x}_{DFO}$ and $\mathbf{x}_{prod}$ from the DFO solution (Section 5.3.5) or from the surrogate-embedded optimization instance (Section 5.3.6). This information is sufficient in simulating the planning level and by extension hierarchical solutions by feeding down optimal setpoints sequentially.

The upper or *planning-only* evaluation (5.8a) only simulates the planning-level objective $f_p(\cdot)$ and any upper-level inequality constraint penalizations g_{viol}. The other metrics differ in how much detail is captured in the optimization as part of each evaluation: The *bi* evaluation (5.8b) calls all scheduling optimization problems in parallel to augment the planning-only objective by c_s^*; *approx tri* (5.8c) calls integrated scheduling-control optimization formulations instead to obtain the optimal scheduling and approximate optimal control costs $c_s^* + \hat{c}_c$; *tri* (5.8d) solves all scheduling problems first whose optimal solutions feed into the optimal control to obtain the 'real' tri-level objective using $c_s^* + c_c^*$. These evaluation metrics present increasingly accurate hierarchical decision-making modelling with the upper (5.8a) and tri (5.8d) evaluations emulating the sequential and hierarchical decision-making frameworks respectively as depicted in Figure 5.1.

$$\text{Upper/Planning-only:} \qquad f_p(\cdot) + g_{viol} \tag{5.8a}$$

$$\text{Bi:} \qquad f_p(\cdot) + g_{viol} + c_s^* \tag{5.8b}$$

$$\text{Approx tri:} \qquad f_p(\cdot) + g_{viol} + c_s^* + \hat{c}_c \tag{5.8c}$$

$$\text{Tri:} \qquad f_p(\cdot) + g_{viol} + c_s^* + c_c^* \tag{5.8d}$$

These metrics become crucial in all three methodologies as described in Figure 5.5: the DFO, surrogate, and combined approach. In principle, any solution quality metrics (5.8a)-(5.8d) can be

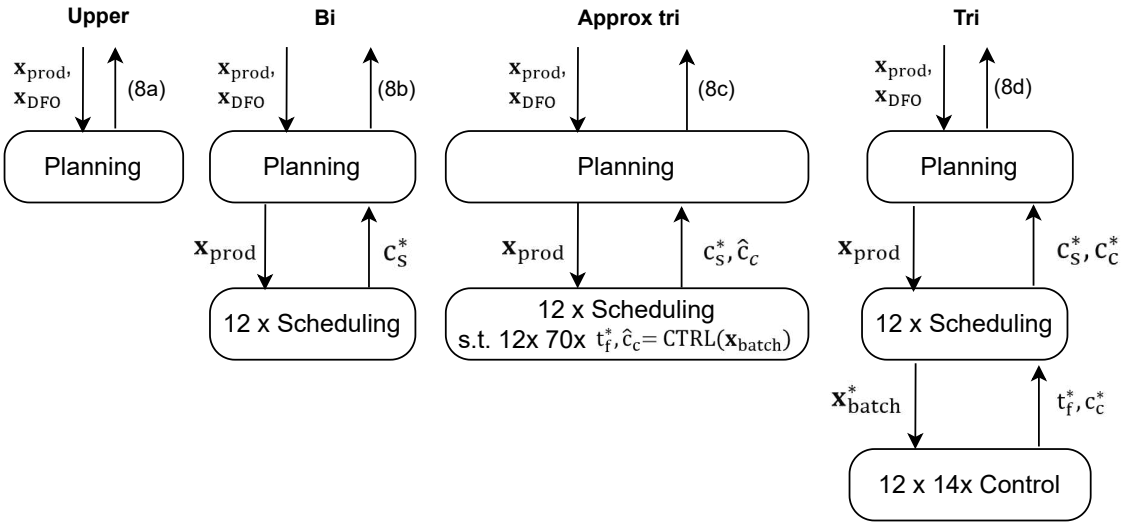


Figure 5.4: Increasing levels of solution accuracy evaluation. (8a)-(8d) refer to evaluation metrics (5.8b)-(5.8d). The upper or planning-only evaluation computes the planning objective and constraints in a simulation. The bi evaluation also solves 12 scheduling problems in parallel in each evaluation to account for the scheduling cost in the objective. The approximate tri evaluation additionally embeds control surrogates into the scheduling to account for approximate control cost in the objective. The tri evaluation solves scheduling and control levels in sequence to emulate hierarchical decision-making and adds the 'real' hierarchical scheduling and control costs to the planning objective.

used as the DFO objective; Similarly, we can train surrogates to map the outputs of scheduling-only (5.10), approximate scheduling-control (5.11), or accurate scheduling-control (5.12) optimization problems, called as part of the bi, approx tri, and tri evaluation metrics (5.8b)-(5.8d). A clear distinction between the increasing levels of solution accuracy becomes important in presenting the results. In the next section, we illustrate how we use DFO to find the planning-level variables that optimize any one of the solution metrics from (5.8).

5.3.4 Partitioning into DFO variables and lower-level scheduling-control problems

In order to use DFO for the integration of all three layers, we first reformulate the tri-level problem (5.7) into the canonical DFO formulation (5.6) for bi-level problems. We split the planning-specific variables $\mathbf{x}_p$ of the tri-level optimization problem (5.7) into its constituent storage $\mathbf{x}_{store}$, transport $\mathbf{x}_{transp}$, and sales $\mathbf{x}_{sales}$ variables. We observe that given $\mathbf{x}_{prod}$, $\mathbf{x}_{transp}$, and $\mathbf{x}_{sales}$, we can exploit the inventory mass balances $\mathbf{h}_p(\cdot)$ to obtain $\mathbf{x}_{store}$. We now have all the variables required to compute the quantities of interest that appear in the upper level: $f_p(\cdot)$, the penalty term g_{viol} via $\mathbf{g}_p(\cdot)$, and any (approximate) optimal costs c_s^* and c_c^* from simulating lower-level

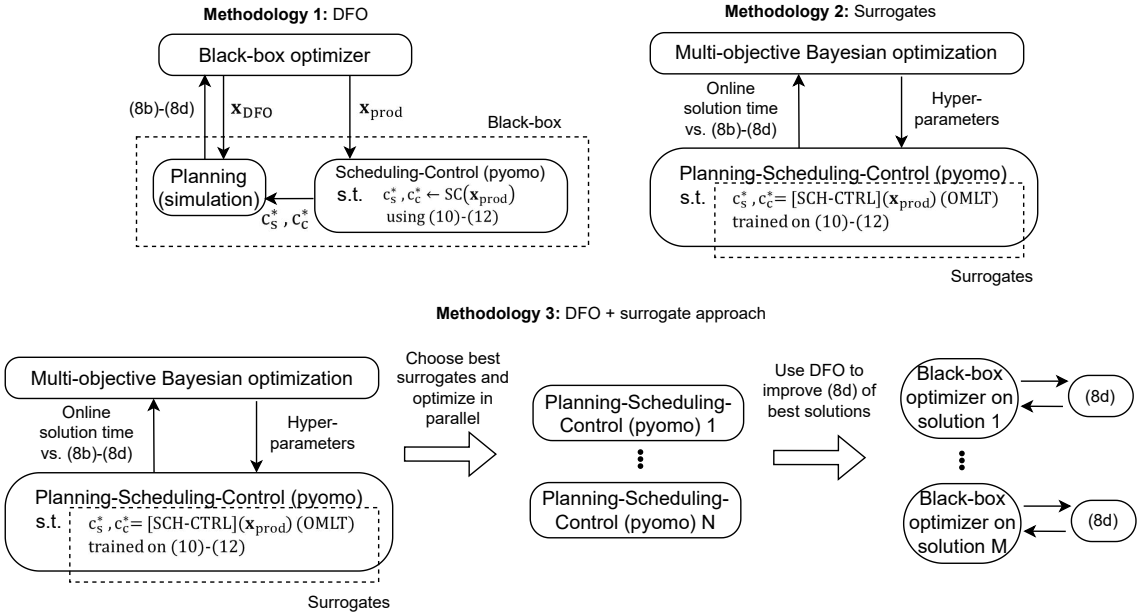


Figure 5.5: The three proposed methodologies: References to formulations as (X) refer to (5.X). The first methodology uses DFO to optimize any of the solution metrics in (5.8a)-(5.8d) directly by considering the planning simulation and scheduling-control optimization as a black-box. The second methodology uses hyperparameter optimization to find a surrogate architecture that, trained on one of the solution metrics in (5.8b)-(5.8d), leads to the best solution accuracy-time trade-off after embedding into the planning. The third methodology combines both workflows in using DFO to optimize the solutions obtained from the surrogate approach.

optimization problems via $\mathcal{SC}(\cdot)$. Problem (5.9) shows the formulations involved in using DFO to find $\mathbf{x}_{sales}, \mathbf{x}_{transp}, \mathbf{x}_{prod}$ that optimize the integrated planning-scheduling-control problem as a black-box objective $\mathcal{BB}(\cdot)$:

$$
\begin{aligned}
\min_{\mathbf{x}_{sales},\mathbf{x}_{transp},\mathbf{x}_{prod}} \quad & \mathcal{BB}(\mathbf{x}_{sales}, \mathbf{x}_{transp}, \mathbf{x}_{prod}) \\
\text{where} \quad & \mathbf{x}_{store} \leftarrow \mathbf{h}_p(\mathbf{x}_{store}, \mathbf{x}_{sales}, \mathbf{x}_{transp}, \mathbf{x}_{prod}) = \mathbf{0} \\
& \mathcal{BB}(\cdot) \leftarrow f_p(\mathbf{x}_{store}, \mathbf{x}_{sales}, \mathbf{x}_{transp}, \mathbf{x}_{prod}) + g_{viol} + c_s^* + c_c^* \\
& g_{viol} \leftarrow \mathbf{g}_p(\mathbf{x}_{store}, \mathbf{x}_{sales}, \mathbf{x}_{transp}, \mathbf{x}_{prod}) \leq \mathbf{0} \\
& c_s^*, c_c^* \leftarrow \mathcal{SC}(\mathbf{x}_{prod}).
\end{aligned}
\tag{5.9}
$$

Comparing (5.9) with the canonical DFO formulation (5.6), we see that $\mathbf{x}_{DFO} = [\mathbf{x}_{transp}, \mathbf{x}_{sales}]$, $\mathbf{x}_{sim} = \mathbf{x}_{store}$, $\mathbf{x}_{u\to l} = \mathbf{x}_{prod}$, and $\mathcal{LOW}(\cdot) = \mathcal{SC}(\cdot)$. In practice, this means that the DFO solver determines only the complicating planning targets, as well as a subset of the planning-specific variables (the transport and sales variables). This information is sufficient in determining all other relevant quantities in computing the black-box planning-scheduling-control objective. Different

formulations of the (integrated) scheduling-control optimization problem $\mathcal{SC}(\cdot)$ lead to different solution accuracy metrics (5.8) via $\mathbf{x}_{l\to u} = \emptyset, c_s^*, c_s^* + \hat{c}_c$, or $c_s^* + c_c^*$,. *Upper/Planning-only* does not use $\mathcal{SC}(\cdot)$. In the *bi* metric evaluations, $\mathcal{SC}(\cdot)$ solves the 12 scheduling optimization problems taking the form of (5.10) to extract c_s^* without lower-level restrictions on t_f:

$$\begin{aligned} \mathcal{SC}(\mathbf{x}_{prod}) = \quad & \min_{\mathbf{x}_s, \mathbf{x}_{batch}, c_s} \quad f_s(\mathbf{x}_s, \mathbf{x}_{batch}, t_f, \mathbf{x}_{prod}) \\ & \text{s.t.} \qquad\quad \mathbf{h}_s(\mathbf{x}_s, \mathbf{x}_{batch}, c_s, t_f, \mathbf{x}_{prod}) = \mathbf{0} \\ & \qquad\qquad\quad \mathbf{g}_s(\mathbf{x}_s, \mathbf{x}_{batch}, t_f, \mathbf{x}_{prod}) \leq \mathbf{0}. \end{aligned} \tag{5.10}$$

In the *approx tri* evaluations, $\mathcal{SC}(\cdot)$ takes the form of (5.11) and approximates the optimal control response in (5.10) for each scheduling problem by embedding the predicted optimal control responses as constraints into the scheduling via: $\hat{c}_{ctrl}^*, \hat{t}_f^* = \mathcal{CTRL}(\mathbf{x}_{batch})$. In each scheduling problem, we enumerate all 7 events by 2 machines by 5 product combinations of optimal control surrogates. The surrogates are trained as follows: we query the optimal costs and processing times corresponding to ten uniformly distributed samples between zero and the upper batch target limit for each equipment-production combination. We then train separate artificial neural networks with two hidden layers of five nodes each on the equipment-product datasets. No hyperparameter tuning is necessary at this point since this simple default architecture gives a good fit on the well-behaved optimal control response and remains tractable after embedding into the scheduling as discussed in Section C.2.1.

$$\begin{aligned} \mathcal{SC}(\mathbf{x}_{prod}) = \quad & \min_{\substack{\mathbf{x}_s, \mathbf{x}_{batch}, c_s, \\ \hat{c}_{ctrl}, \hat{t}_f^*}} \quad f_s(\mathbf{x}_s, \mathbf{x}_{batch}, \hat{t}_f^*, \mathbf{x}_{prod}) \\ & \text{s.t.} \qquad\quad \mathbf{h}_s(\mathbf{x}_s, \mathbf{x}_{batch}, c_s, \hat{t}_f^*, \mathbf{x}_{prod}) = \mathbf{0} \\ & \qquad\qquad\quad \mathbf{g}_s(\mathbf{x}_s, \mathbf{x}_{batch}, \hat{t}_f^*, \mathbf{x}_{prod}) \leq \mathbf{0} \\ & \qquad\qquad\quad \hat{c}_{ctrl}^*, \hat{t}_f^* = \mathcal{CTRL}(\mathbf{x}_{batch}) \end{aligned} \tag{5.11}$$

In the *tri* evaluations, two sequential optimization layers are solved within $\mathcal{SC}(\cdot)$: first, all 12 scheduling problems (5.10) are solved in parallel. Then, the optimal batch responses $\mathbf{x}_{batch}^*$ for each scheduling problem as determined in (5.10) feed into the 7 events by 2 machines combinations of optimal control problems as shown in (5.12). The control problem outputs are then returned as the 'true' optimal control costs and processing times to the scheduling and planning levels. In our case, the tri evaluations happen to be cheaper than the approx tri evaluations, as will be discussed in Section 5.4.2. However, this is rarely known *a priori* and is not expected to be the case when the

size of the control problems increases. As a result, we use the tri evaluation only to check 'ground truth' solution quality in the first two methodologies and for minimal fine-tuning in the third.

$$\begin{aligned} \min_{\mathbf{x}_c, c_{ctrl}, t_f} \quad & f_c(\mathbf{x}_c, t_f, \mathbf{x}^*_{batch}) \\ \text{s.t.} \quad & \mathbf{h}_c(\mathbf{x}_c, t_f, c_{ctrl}, \mathbf{x}^*_{batch}) = \mathbf{0} \\ & \mathbf{g}_c(\mathbf{x}_c, t_f, \mathbf{x}^*_{batch}) \leq \mathbf{0} \end{aligned} \tag{5.12}$$

Key to the tractable integration using DFO is parallelization in the scheduling layer. In our case, we see that the scheduling-level capacity for P1, P2, P3, and P4 production (Figure 5.2A) at any given time step is coupled to the planning level and other scheduling problems only via the inventory levels of I22. In the planning level, we are only interested in the changeover costs of the scheduling layer. Therefore, rather than solve the scheduling problems sequentially in time, we fix I22 inventory in the scheduling problems and solve all twelve scheduling problems in parallel leading to significant computational savings. In doing so, we rely on the planning-level safe storage constraints to ensure that we have enough I22 inventory for the production of any given planning target. Parallelization at the control level is in principle also possible, but in our case limited by the availability of compute nodes.

5.3.5 Methodology 1: Derivative-free optimization

The first methodology involves using DFO to find $\mathbf{x}_{prod}, \mathbf{x}_{transp},$ and $\mathbf{x}_{sales}$ that optimize the solution metrics in (5.8). However, all solution metrics apart from planning-only (5.8a) are considered expensive as they call at least one additional level of optimization problem in each evaluation. Consequently, the DFO budget is severely limited to a couple of hundred evaluations given the high dimensionality of 178 DFO variables. This prevents even the most exploitative solvers from making meaningful progress. To mitigate this issue, we hot-start the DFO search by using the Pyomo [394] solution of the planning-only optimization problem (5.13) as an initial guess:

$$\begin{aligned} \min_{\mathbf{x}_p, \mathbf{x}_{prod}} \quad & f_p(\mathbf{x}_p, \mathbf{x}_{prod}) \\ \text{s.t.} \quad & \mathbf{h}_p(\mathbf{x}_p, \mathbf{x}_{prod}) = \mathbf{0} \\ & \mathbf{g}_p(\mathbf{x}_p, \mathbf{x}_{prod}) \leq \mathbf{0}. \end{aligned} \tag{5.13}$$

Then, we sequentially optimize the bi (5.8b) and approx tri metrics (5.8c) using 500 evaluations each, where the solution of the former feeds into the latter as an initial guess. We assume *a priori* that the tri metric (5.8d) is intractable, and only use it to check solution quality. In principle, instead of this cascading approach, we could optimize the solution metric of interest from the start in the same solution time, which we discuss in Sections 5.4.1 and 5.4.4.

Any DFO solver (Bayesian optimization, evolutionary search, direct methods) can be used for this task. Informed by the benchmarking of data-driven coordination solvers in Chapter 4, we choose the exploitative trust region optimizer Py-BOBYQA [71]. In fact, we encounter a similar scenario, where the DFO evaluation budget only allows for a fine-tuning of the solution around the initial guess rather than an extensive exploration of the solution space. In Chapter 6, we revisit this issue by proposing a subspace optimization technique that can be used in conjunction with hot-starting to improve the scalability of the DFO methodology. Alternative high-dimensional DFO techniques also depend on optimizing over lower-dimensional embeddings, or otherwise exploiting structure in the original search space [70, 144, 231, 388].

5.3.6 Methodology 2: Optimality surrogates

In Section 5.3.4 we describe how we can use optimality surrogates to map the optimal control response as a function of the batch targets $\hat{c}^*_{ctrl}, \hat{t}^*_f = \mathcal{CTRL}(\mathbf{x}_{batch})$. Similarly, for each planning step, we can construct optimality surrogates that return the optimal scheduling and control costs at said step as a function of the four production planning targets $\hat{c}^*_s, \hat{c}^*_c = [\mathcal{SCH} - \mathcal{CTRL}](\mathbf{x}_{prod})$. The neural network $[\mathcal{SCH} - \mathcal{CTRL}](\cdot)$ is trained to map the output of the (integrated) scheduling-control optimization problem $\mathcal{SC}(\cdot)$, which can take the form of the scheduling-control optimization instances (5.10)-(5.12). Consequently, we can approximate the solution to the tri-level problem (5.7) by solving the single-level formulation (5.14):

$$
\begin{aligned}
\underset{\substack{\mathbf{x}_p, \mathbf{x}_{prod}, \\ \hat{c}^*_s \hat{c}^*_c}}{\min.} \quad & f_p(\mathbf{x}_p, \mathbf{x}_{prod}) + \hat{c}^*_s + \hat{c}^*_c \\
\text{s.t.} \quad & \mathbf{h}_p(\mathbf{x}_p, \mathbf{x}_{prod}) = \mathbf{0} \\
& \mathbf{g}_p(\mathbf{x}_p, \mathbf{x}_{prod}) \leq \mathbf{0} \\
& \hat{c}^*_{s,t}, \hat{c}^*_{c,t} = [\mathcal{SCH} - \mathcal{CTRL}]_t(\mathbf{x}_{prod}), \quad t = 1, \ldots, 12.
\end{aligned}
\tag{5.14}
$$

Before training, we sample 1,000 combinations of the planning targets $\mathbf{x}_{prod}$, where the upper bound is obtained from the planning-level resource limit for each product. Each sample involves

the solution of $\mathcal{SC}(\cdot)$ as formulated in the scheduling-only (5.10) or approximate scheduling-control (5.11) formulations associated with the bi (5.8b) and approx tri (5.8c) metrics respectively. In the results section, we also repeat this workflow with surrogates trained on the accurate scheduling-control optimization problems (5.12) associated with metric (5.8d) to investigate the effect of inaccurate control information. We add fixed penalty terms to the planning objective if the scheduling (or control) problems become infeasible.

Deciding on the architecture of the integrated scheduling-control surrogate is more difficult than mapping the optimal control surrogate. The bigger the neural network, the more capable it is (in principle) of accurately capturing the optimal response function, but the more computationally expensive the planning formulation becomes after embedding of the surrogate. To navigate this accuracy-tractability trade-off, we use multi-objective Bayesian optimization for 12 samples to find the number of nodes in the first and second hidden layer n_1, n_2 that best trade off accuracy and tractability. Each Bayesian optimization sample includes: the training of a network with hidden layers n_1, n_2; its embedding into the planning layer; optimization of the planning after embedding; and a solution quality check on all metrics of (5.8). The optimization times are returned with the approx tri (5.8c) or tri (5.8d) solution metrics and fed to the Bayesian optimization to suggest the next iteration of hyperparameters.

5.3.7 Methodology 3: Combining both approaches

The third methodology consists of combining the first two approaches to bring out the best of both worlds: We train a collection of the most promising network architectures identified in the offline hyperparameter tuning from Methodology 2. We then optimize separate planning problems in parallel, each with a different set of embedded surrogates, and evaluate their solution quality using the tri evaluation (5.8d) as the most accurate metric we can afford. Finally, we use DFO (in parallel) on the best solutions from the surrogates following Methodology 1 to fine-tune them towards the tri-level optimum using the tri metric (5.8d) in the remaining online evaluation budget.

5.3.8 Software and practical considerations

We use Py-BOBYQA [71] as our DFO solver of choice for the DFO approach (Methodology 1), and the OpenBOX implementation [234] for the multi-objective Bayesian optimization using Gaussian processes in the optimality surrogate hyperparameter tuning (Methodology 2). The rest of

the conventional optimization formulations are implemented in Pyomo [164]. While the control instances of Equation (5.12) are solved using Ipopt [394], the optimality surrogate formulation (5.14) and the integrated scheduling-control optimization instances $\mathcal{SC}(\cdot)$ in (5.10) and (5.11) are solved using Gurobi [157]. We use PyTorch [293] to train the neural networks and embed them as constraints into Pyomo blocks using OMLT [77]. We run the main DFO and surrogate scripts on the Imperial College Research Computing HPC service. Most scripts run on a maximum walltime of 8 hours, and we increase the number of compute nodes and memory as needed, usually within the range of 64-128 and 10-50gb respectively. Notable exceptions include the DFO approach on the approx evaluation using 18 hours of walltime, 256 nodes, and 25 gb memory, or the surrogate hyperparameter tuning on the tri evaluation using 8 hours of walltime, 128 nodes, and 100 gb.

In the next section, we present the solution time and accuracy results of Methodologies 1 and 2. In both approaches we are assuming that we have access to tri evaluations, but that they are expensive, meaning that we can only use them for checking the 'ground truth' solution quality. In Methodology 3, we then present the solution time and accuracy results and discuss the advantages of combining both approaches together with relevant practical considerations. Throughout this, we place special emphasis in our discussion on distinguishing between the different solution accuracy metrics, between on- and offline solution times, and the implications of model inaccuracies.

5.4 Results and Discussion

Table 5.1 gives an overview of what methods we benchmark on which criteria for each figure. We perform standard solution accuracy and solution time comparisons for each methodology, and include additional analysis in the surrogate results section: a study on performance variability of all surrogates queried in the hyperparameter tuning (Figure 5.8), and a comparison of on- and offline optimization times in the surrogate approach (Table 5.2). Furthermore, we link to supplementary analysis in the appendix where appropriate: an explanation of why optimality surrogates are particularly powerful for optimal control problems (Section C.2.1); an exploratory data analysis of sampled optimal scheduling costs for optimality surrogate training (Section C.2.2); a sensitivity analysis on surrogate performance metrics as a function of surrogate size (Section C.2.3); a qualitative discussion on the relative scaling of the DFO and surrogate approaches with the relative size of problems and linking variables (Section C.2.4); and a permutation study on the problem parameters (Section C.2.5).

Table 5.1: Summary of experiments performed

Section	Figure	Approach	What is evaluated	What is compared
5.4.1	Figure 5.6a	DFO	All 4 evaluation metrics (5.8)	Initialization, bi DFO, approx tri DFO
5.4.1	Figure 5.6b	DFO	Online solution time	Effect of parallelization on 3 configurations
5.4.2	Figure 5.7a	Surrogate	All 4 evaluation metrics (5.8)	3 best bi and approx tri surrogates
5.4.2	Figure 5.8	Surrogate	All 4 evaluation metrics (5.8)	All 12 bi, approx tri, and tri surrogates
5.4.2	Figure 5.7b	Surrogate	Online optimization time	3 best bi and approx tri surrogates
5.4.2	Table 5.2	Surrogate	On- and offline solution times	Bi, approx tri, and tri surrogates
5.4.4	Figure 5.9a	Combined	Tri evaluation metric (5.8d)	DFO tri on best solutions
5.4.4	Figure 5.9b	Combined	Online solution time	DFO tri on best solutions

5.4.1 Methodology 1: Integration into planning via DFO

Optimizing the scheduling-only problem or scheduling problem with embedded surrogates as presented in the previous section constitutes the bottleneck in the bi and approx tri evaluations. Figure 5.6 illustrates the solution accuracy obtained and time required to perform DFO on the bi and then on the approx tri metrics for 500 evaluations each.

Solution accuracy

Figure 5.6a shows the different solution metrics obtained in the DFO. We start with the bi DFO (DFO on the bi metric) using the solution obtained from the optimization of the planning-only problem in Pyomo. This planning-only solution is infeasible for all evaluation metrics accounting for lower-level information.

The planning-only solution then feeds into the bi DFO where the bi metric is improved from around 6 to -3 in 500 evaluations. Then, the bi DFO solution feeds into another round of 500 approx tri DFO evaluations, improving the approx metric from the bi DFO solution from about -2 to -3. This goes to show that DFO reliably improves the solution according to the evaluation metric used in the objective. Yet, there is no guarantee that an increase in cheaper bi/approx evaluations comes with an improvement in more accurate evaluations that are not used as the objective function of the DFO. This explains why the tri evaluation shows no improvement - all 3 tri evaluations remain infeasible in some scheduling and/or control problems. As a side note, performing the same experiment using an uninformed initial guess of zero prevents Py-BOBYQA from making any update whatsoever on the bi and approx tri evaluations, which stay around 5.

Realistically, given the expense of the scheduling-level optimization problems and the high dimensionality in the planning-level variables, we only have enough budget to fine-tune the solution

obtained in the initial guess. This is confirmed in the planning profiles associated with the DFO runs in the Appendix C.3. However, this fine-tuning can be crucial in rendering the planning-only solution feasible with respect to more accurate metrics, resulting in large increases in solution quality. While we use fixed costs to penalize infeasibilities in the scheduling and control level, DFO might make more consistent progress if soft penalty violations are used between the suggested and nearest feasible set of planning and batch targets, like in the augmented Lagrangian penalty term of (4.6) in the data-driven coordination framework.

5

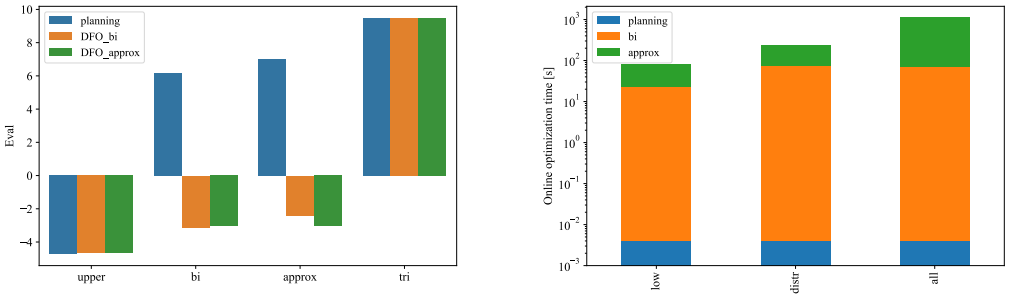
Solution time

Figure 5.6b illustrates the solution times for generating the planning-only initial guess and performing 500 DFO evaluations using the bi and approx tri metrics across three configurations of the case study. In *low*, a single average scheduling problem is solved for all planning steps, considering the optimal control of only one product; in *distr*, separate scheduling problems are solved in parallel for each planning step for one product; *all* encompasses the full case study, as described in Section 5.2, considering all five products in parallel at each planning step. Comparing the 'low' and 'distr' configurations allows us to examine the impact of parallelization, while comparing the 'distr' and 'all' configurations allows us to analyze the effect of increasing the number of control problems per scheduling instance.

We see that the planning-only Pyomo instance returns a solution in milliseconds. When going from the 'low' to the 'distr' configuration, the bi and approx DFO times only increase the solution time from around 20 to 80 and from around 60 to 200 seconds. This highlights the benefits of parallelization: rather than seeing an increase in computational time proportional to the 12 planning steps, we only see it increase by a factor of around 4. While in theory parallelization could keep the solution time constant, in practice, each evaluation call is limited by the time it takes for the slowest scheduling instance to be solved. Increasing the number of optimal control problems when moving from 'distr' to 'all' only increases the approx DFO time. Increasing the number of control problems by 5 increases the solution time by a factor of 10. This is to be expected given the discrete nature of the variables introduced in the surrogates. In Table C.1, we claim that the optimization instances in a single bi or approx tri evaluation take 10 and 250 seconds respectively. However, Figure 5.6b shows that 500 evaluations take around 100 and 1,000 seconds respectively. While this may seem contradictory, this means that most of the DFO samples are quickly determined to be infeasible.

Cascading approach

The question arises if rather than use a cascading approach of feeding the DFO solutions of less accurate metrics as initial guesses into the DFO of more accurate metrics, we can obtain better solutions by using the available budget to have more samples to optimize the expensive metric of interest directly. The answer to this is case-dependent and is influenced by the extent to which the metrics corresponding to the different levels of integration are correlated. In our case, the cascading approach is justified since the DFO on the bi metric is cheaper by an order of magnitude and still makes significant progress on the approx evaluation. However, if the ultimate goal is to optimize the tri evaluation, then neither the optimization of the bi nor the approx tri metric make significant progress. We continue this discussion in Section 5.4.4.



(a) Best evaluation on the upper, bi, approx tri, and tri solution metrics (from left to right)

(b) Cumulative online optimization time on the low, distributed, and full configurations

Figure 5.6: Solution accuracy and time for three methods: solution initialization using planning-only metric (blue), DFO on the bi metric (orange), and DFO on the approx tri metric (green). The solution accuracy bar plots denote the evaluation (y-axis) that the three methods (blue, orange, and green) achieve on the four solution quality metrics (x-axis) corresponding to (5.8a)-(5.8d). For example, using DFO with the bi solution metric as objective function (orange) outperforms DFO approx (green) on the bi solution metric (second bin). However, DFO approx (green) outperforms DFO bi (orange) on the approx solution metric (third bin). All three considered approaches are infeasible in the tri solution metric (fourth bin), and consequently heavily penalized, leading to a bad evaluation.

5.4.2 Methodology 2: Integration into planning via surrogates

In Methodology 2, rather than perform DFO on the bi or approx tri evaluations, we train optimality surrogates as a function of the planning targets to map the scheduling and control costs from the scheduling-only (5.10) and approximate scheduling-control problems (5.11). Since these instances constitute a major part of the bi and approx tri evaluations, we refer to the surrogates trained on them as bi or approx tri surrogates respectively. In Figure 5.7, for the bi and approx tri surrogates,

we present three architectures each that return a Pareto-optimal solution between solution time and best approx tri performance after optimization of the planning level with the embedded surrogate.

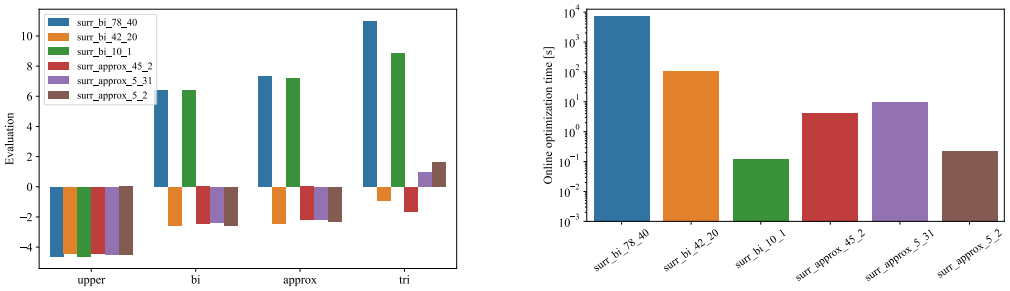
Solution accuracy

Figure 5.7a shows that the solutions obtained from the approx tri surrogates are more consistent and achieve better solution quality than those obtained from the bi surrogates. This suggests that using more informative metrics to build the surrogate translates into better solution accuracy. The bi surrogates, only trained on scheduling-only problems used in the bi evaluations, might miss valuable information that relates surrogate architecture, training accuracy, and optimization solution quality, leading to poorer, less consistent performance. As opposed to the DFO where using the bi evaluation metric as the objective significantly improves the bi evaluation, the bi surrogates even fail to consistently find a good bi evaluation.

However, it is surprising to see that the solution check on the tri metric evaluations corresponding to the approx tri surrogates are significantly better than the ones obtained in the DFO. This suggests a different explanation for why the surrogate approach performs surprisingly well. Rather than the surrogate approach performing well *despite* the inevitable small model inaccuracies, the surrogate approach might perform well *because* of them. In the DFO solutions, we are starting off from the planning-only solution that is optimal in the planning-level formulation only and active in some constraints that quickly become infeasible when more accurate lower-level information becomes available. Small inaccuracies in the surrogates might give a suboptimal solution that is further away from the constraints and as a result more 'robust' to information from the lower level. This hypothesis is supported by the planning solution profiles of the surrogate approaches being significantly different to the profiles observed for the centralized and DFO solutions as seen in C.3. It is in principle possible that model inaccuracies might be biased towards giving solutions in the infeasible space, but these models should be filtered out by the hyperparameter tuning process.

Consistency in solution accuracy

The major risk of the surrogate approach lies in selecting an architecture or training a specific model that leads to poor performance. To determine the effect of the model architecture on performance, Figure 5.8 displays the solution accuracy associated with all 12 model architectures queried in the hyperparameter tuning. Additionally, we investigate how much of the variability in bi and approx tri surrogate model performance is due to them not being trained on accurate control information,



(a) Best evaluation on the upper, bi, approx tri, and tri solution metrics (from left to right)

(b) Online optimization time after embedding of the surrogates

Figure 5.7: Solution accuracy and time of the best optimality surrogate solutions: the first three surrogate architectures are trained on the scheduling-only (bi: blue, orange, green), and the other three (approx: red, purple, brown) on the approximate scheduling-control subproblems. The two numbers in the label refer to the number of nodes in the two hidden layers. For example, all 6 surrogate approaches lead to a comparable solution quality at -4 in the planning-only solution metric (first bin). All three surrogates trained on the approx solution quality metric (red, purple, brown) achieve a similar performance around -2 on the approx solution quality metric (third bin), whereas the surrogate with 42 and 20 nodes (orange) is the only surrogate out of the three trained on the bi solution metric that achieves a good evaluation on the approx solution metric around -2 rather than 7. The surrogate with 45 and 2 nodes respectively (red) achieves the best performance on the tri solution quality (fourth bin).

by following the same workflow to train tri surrogates on accurate scheduling-control instances as used in the tri metric (5.8d).

Overall, the best of the 12 samples display similar performance among the three surrogate types on all solution metrics. In general, we see that the bi and tri surrogates display a similar median and variation in solution accuracy, with the bi surrogates being slightly more consistent in the tri metric evaluation. While the approx tri surrogates display a better median evaluation in the bi and tri metric evaluation, they also display more variability in solution performance. This is surprising to see given Figure 5.7a, where the best approx tri surrogate Pareto solutions seem to be more consistent than their bi surrogate counterparts. This can be explained however by 'bad' approx surrogates faring even worse because of compounding inaccuracies due to imperfect control surrogates.

The question still remains as to the tri surrogates' inability to clearly outperform the other surrogates. One possible explanation could be that accounting for the optimal control makes the solution space too ill-behaved to be accurately captured by the networks. Overall, the surrogates display too similar performance to argue that one type outperforms the other. This analysis calls for a sensitivity analysis of the surrogate performance as a function of surrogate size. In C.2.3, we discuss correlations between model size, test accuracy, integrated optimization time, integrated

optimization result, and tri metric evaluation. In short, we conclude that the best predictor of tri metric performance is test accuracy, which does not monotonically improve with surrogate size, prompting the need for careful hyperparameter selection.

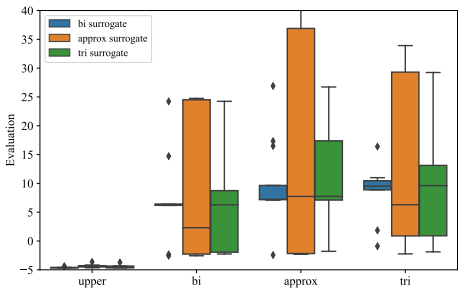


Figure 5.8: Box-and-whisker plot of solution accuracy variability for all queried surrogates in the hyperparameter tuning: Surrogates trained on scheduling-only (bi: blue), approximate scheduling-control (approx: orange), and accurate scheduling-control (tri: green) information are compared on upper, bi, approx tri, and tri solution metrics from left to right. The box represents the quartiles, while the whiskers demonstrate variability outside the box plots. Outliers are shown as isolated data points outside of the whiskers.

Sections C.2.1 and C.2.2 give interesting insights into the differences between the optimality surrogate approach for embedding control information into the scheduling layer, and for embedding scheduling-control information into the planning level. Mapping optimal scheduling cost appears to be a more difficult problem due to the following reasons. First, the optimal control surrogate is lower-dimensional in that it maps optimal processing times and control cost as a function of batch targets, while optimal scheduling surrogates take in four planning target variables. Additionally, while there are linear relationships in both optimality surrogates, the nonlinearities in the control problem only display a slight concave curvature at higher batch targets, whereas infeasibilities in the scheduling problems lead to a more ill-behaved mapping over multiple orders of magnitude. As discussed in Section 2.2.3, it is well-established in the literature that piecewise affine functions can present *exact* solutions for simpler, linear mp-MPC [314]. The mixed-integer nature of scheduling problems however can present additional modelling challenges that require careful handling.

Solution time

Figure 5.7b shows the online optimization time required to solve the embedded problem as shown in Formulation (5.14). The online optimization times of the bi and approx tri surrogates range widely from 100 milliseconds to 2 hours based on the size of the networks rather than the type of surrogate. For example, the bi surrogate with hidden layers of sizes 10 and 1 leads to a solution time comparable to that of the approx tri surrogate with sizes 5 and 2. The biggest surrogate (78

and 40 nodes) times out after 2 hours of solution time and displays the worst of the Pareto solutions in terms of accuracy. For the bi surrogates, the best solution is attributed to the medium-sized surrogate of 42 and 20 nodes as a trade-off between accuracy and tractability. However, for the approx tri surrogates, the best solution is found using the smallest network. This could suggest that the approx metric response surface is smoother given the composite approximations, and consequently more easily captured by smaller surrogates.

While Figure 5.7b shows the online optimization time as a function of surrogate size, Table 5.2 shows that the difference in solution time between the different types of surrogates is shifted offline to the sampling, training, and hyperparameter tuning steps. The training times are essentially negligible with each surrogate being trained in a maximum of 90 seconds. The hyperparameter tuning time can be adjusted based on the available budget. In our case, for each surrogate type, we limit the online optimization time per integrated problem to 2 hours and the hyperparameter tuning time involving 12 surrogates to 8 hours. Essentially, the bottleneck in moving from bi to approx tri surrogates is in the sampling step. We sample 1,000 bi, approx tri, and tri evaluation samples, which takes between 8 and 24 hours. These evaluation times are considerably less than what would be expected from Table C.1 given that half of the samples are infeasible.

It is surprising to see that the tri sampling time is shorter than the approx tri sampling time. It turns out that solving the planning problems, the scheduling problems, and then the 168 (2 machines by 7 events by 12 planning steps) optimal control problems in sequence is five times faster on average than solving the scheduling problems, each embedded with 70 optimal control surrogates (7 events by 2 machines by 5 productions). In other words, it is not trivial to see that embedding all possible optimal control combinations into the scheduling displays worse scaling than evaluating the 'ground truth' tri metric by solving the levels hierarchically. In C.2.4, we qualitatively discuss how we expect the DFO and surrogate approaches to scale with respect to the dimensionality of complicating variables and problem sizes. In hindsight, for the case study at hand, we conclude that since the tri evaluation is faster than the approx tri evaluation, the performance of using approx tri evaluations in the DFO or in their corresponding approximate scheduling-control problem (5.11) for surrogate training is not worth the compute.

5.4.3 Comparing the DFO and surrogate approaches

In the above two sections, we highlight the accuracy-tractability trade-offs associated with both approaches. DFO does not introduce additional inaccuracies if the lower levels are solved to global

Table 5.2: Online optimization, and offline hyperparameter tuning, training, and sampling times associated with the construction of surrogates in the integration of scheduling-only, approximate scheduling-control, and accurate scheduling-control lower-level information

		Time			
	Related evaluation	Online	Hyperparameter tuning	Training	Sampling (N=1,000)
Scheduling-only	Bi	0.1min-2hrs	8 hrs	up to 12x90 sec	8 hours
Approximate scheduling-control	Approx tri	0.1min-2hrs	8 hrs	up to 12x90 sec	24 hours
Scheduling-control	Tri	0.1min-2hrs	8 hrs	up to 12x90 sec	8 hours

optimality. As a result, DFO guarantees a solution at least as good as the initial guess on the evaluation metric used as its objective. The main drawback of DFO is its poor scaling in the number of planning-level variables and in the computational expense of the lower-level problems. Realistically, given these budget restrictions, we can only expect to fine-tune the solutions around the initial guess.

Generally, the surrogate approach scales better at the potential expense of solution quality. A major advantage of the surrogate approach is that tractability and accuracy can be easily adapted to the problem at hand by using Bayesian optimization (which ironically counts as DFO too). Empirically, we see that even the optima obtained with smaller, more tractable, and potentially less accurate models can display tri-level metric solutions that outperform any of the solutions obtained via DFO. However, by using surrogates, we forfeit any guarantee in the solution quality obtained. While surrogate techniques promise online computational savings, we have to invest our efforts offline into the model selection and validation process. In the next section, we synthesize these findings into a proposed method that leverages the advantages of both methods.

5.4.4 Methodology 3: Bringing out the best in both approaches

Figure 5.9 shows the solution quality-computational expense trade-off incurred by running DFO using the tri evaluation metric on the planning-only initialization, best DFO, and best surrogate solutions.

Solution accuracy

Figure 5.9a shows that running 500-1,000 DFO evaluations using the tri metric on prior DFO and surrogate solutions significantly improves their tri evaluation. The best tri evaluations are found by performing DFO on the bi and tri surrogate solutions for a standard budget of 500 samples,

improving the tri metric from about -1 and -2 to around -2.5 respectively. We use the same budget on the approx tri DFO solution, improving the solution from about 9.5 to 8.5. However, by running DFO on the planning-only and bi DFO solutions using 1,000 and 800 evaluations respectively, we find a solution that is just as good in less time than is required to perform the approx tri DFO in the first place. Performing tri DFO on the bi DFO solution even improves the solution to 7.5. This links back to the previous discussion in Section 5.4.1 of whether it is favourable to perform DFO on increasing levels of metric complexity in a cascading manner or if we should use the available budget to only optimize the targeted metric from the start. In our case, we find that we should use the cascading approach but avoid the more expensive approx tri evaluation in favour of the tri evaluation.

Solution time

Figure 5.9b compares the computational time required to fine-tune the solutions using tri DFO with the time it takes to obtain the initial guesses in the first place. 500 tri DFO evaluations on the approx tri DFO and surrogate solutions take about 240 minutes. To put this into context, the initial 500 approx tri DFO evaluations take about 1,000 minutes, and the optimality surrogates up to 120 minutes in the first place. Performing 1,000 and 800 tri DFO evaluations takes about 430 and 390 minutes to achieve a final tri evaluation of 8.5 and 7.5 starting from the planning-only and bi DFO solutions respectively. In both cases, the tri DFO constitutes the computational bottleneck compared with the 0.2 and 70 minutes needed to obtain the initial guesses in the first place.

While using the bi DFO solution as initial guess outperforms using other DFO initial guesses for a given time budget, better solutions can be obtained in less time when hot-starting the tri DFO with the surrogate approach. In our case, the 500 tri DFO evaluations take about 2-3 times longer than the pure surrogate approach. The relative solution time between the tri evaluations and surrogate-embedded optimization changes significantly with the number of infeasible samples in the DFO and the size of the surrogates. However, this ratio can be manipulated by the DFO budget and the maximum online integrated optimization runtime.

Sensitivity analysis

The relative performance of the DFO and surrogate approaches only holds for the current problem instance. Although we would ideally want to perform a comprehensive sensitivity analysis on problem topology and parameters, this endeavour is intractable. We qualitatively discuss how

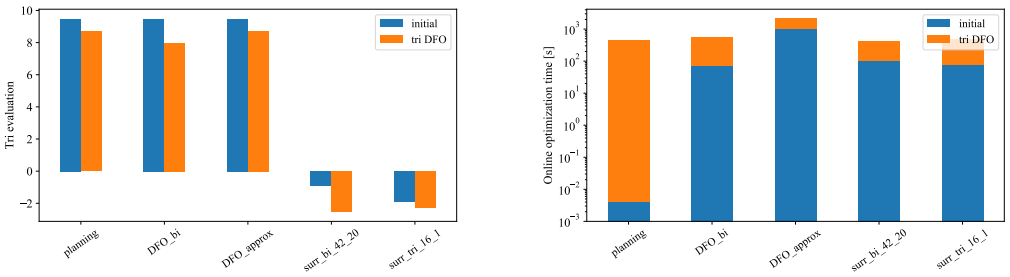
we expect the DFO and surrogate approaches to behave with changes in relative problem and linking variable sizes in Section C.2.4. In Figure C.7, we perform the same tri DFO analysis as in 5.9a in three perturbed instances, whose parameters (demand, forecatsts, changeover times, …) are changed by up to 1%. While this might seem like a small perturbation, it is hard enough to find a feasible solution to the default parameters. For all perturbed instances, the tri surrogates always outperform the DFO solutions and usually match the bi surrogates on the tri metric before fine-tuning. However, the best solution in all three instances is found by performing tri DFO on the bi surrogate solution.

Practical recommendations

Optimality surrogates elegantly trade-off tractability and accuracy. However, their variability in solution performance is concerning. In order to mitigate these drawbacks, we suggest finding safety in numbers and choosing multiple surrogates - either the same model architectures trained on slightly different datasets, or different architectures potentially even trained on different kinds of evaluations. We can then embed the different surrogates into separate planning instances and optimize the resulting formulations in parallel in the online optimization step.

All solutions are then checked using the most accurate evaluation metric available. Finally, we perform DFO on this evaluation metric in the hopes of finding a better solution in the remaining time budget. At this stage, the solution quality is guaranteed to be as good as the best surrogate solution for a chosen evaluation metric. Alternatively, we can still fall back on the much cheaper planning-only solution, which represents the traditional ‘sequential’ solution approach.

The combined approach fully exploits the available on- and offline optimization budget. By leveraging parallelization, we can mitigate the risks of model inaccuracy during online optimization, while still benefiting from the surrogate approach, which shifts much of the computational effort offline. The amount of effort dedicated to the offline sampling, training, hyperparameter tuning, and model validation step can be adjusted on a case-by-case basis to an acceptable tolerance. The remainder of the available online solution time can then be fully used to fine-tune the surrogate solutions via DFO.



(a) Tri evaluation improvement

(b) Cumulative online optimization time

Figure 5.9: Increase in solution quality and time when using DFO with the tri metric (orange) on the best solution (blue) of other methods. The solution approaches, included as initial guess, from left to right, are: planning-only initialization, DFO on the bi and approx tri metrics, and the best surrogates trained on the bi and approx tri metrics. For example, fine-tuning the solution obtained from the planning-only solution (first bin) increases the tri solution metric from about 9.5 to about 9.0. Fine-tuning the solution obtained from the surrogate trained on the bi solution metric improves the evaluation obtained on the tri solution metric (fourth bin) from about -0.5 to below -2.

Additional considerations for the DFO approach

In this work, we argue that the true potential of the DFO framework emerges when the case study exhibits at least one of the following properties: 1) The solution space contains exploitable structure at the DFO solver level, or 2) High-quality initial guesses can be obtained at low computational cost.

The first property is particularly relevant when the objective function is insensitive to certain planning-level variables or when strong correlations exist between these variables. For example, in our case, production of either A or B can be increased, but increasing both simultaneously leads to subproblem infeasibility due to resource constraints. Recognizing such patterns allows the DFO solver to focus on a reduced, more structured search space.

The second property is harder to anticipate in practice. Intuitively, initializing DFO with a good starting point—such as solving a single-level formulation first—can be beneficial, but only if incorporating lower-level considerations does not significantly alter the solution space. If the lower-level problem primarily adjusts objective costs, rather than introducing complex constraints that disconnect the previous solution from the new optimum, then warm-starting is likely to be effective.

While fine-tuning via DFO can be computationally expensive in realistic scenarios, we argue that multi-level solution quality checks should be considered best practice. As problem scale

increases, the dimensionality of complicating variables can grow rapidly, often exceeding thousands. However, a one-off hierarchical validation using multi-level metrics — such as the tri-metric - could still be enabled via efficient evaluation through parallelization as a much more reasonable and practical approach. Additionally, a multi-level solution metric could serve as an arbiter to compare solutions obtained from different hierarchical methods, such as planning-only models or alternative optimality surrogates.

In hindsight, an alternative formulation could have been explored: instead of modeling the entire upper-level problem as a simulation within DFO, we could solve only for $\mathbf{x}_{u\to l}$ in the DFO, then extract $\mathbf{x}^*_{l\to u}$ by solving all lower-level problems $\mathcal{LOW}$, and finally solve f_{up} as an *additional* level. The trade-off here is evident: we can significantly cut down on the number of DFO variables at the expense of solving an additional optimization problem. Given that planning problems are typically much smaller than scheduling problems in hierarchical decision-making, this trade-off could be worthwhile. However, the dimensionality of the complicating variables would still grow linearly with time and the number of products, raising concerns about the scalability of this modified DFO approach for industrial-scale applications.

5.5 Conclusion

In this chapter, we present how derivative-free optimization (DFO) and optimality surrogates can be used to optimize general bi-level formulations. We show how these techniques can increase the solution quality of a multi-level, multi-site hierarchical planning-scheduling-control problem with respect to the conventional sequential solution approach. When tackling a task as ambitious as solving large-scale tri-level formulations, it is crucial to select tools that achieve this in minimal time. At the same time, we need to mitigate the risk of the solution becoming unusable due to model inaccuracies. We synthesize our findings into a methodology that leverages the advantages of both techniques: We use derivative-free optimization (Bayesian optimization) in the offline model selection process to efficiently trade-off solution accuracy and tractability. We then mitigate the risk of model inaccuracies in the surrogates by employing the surrogate approach in parallel on multiple planning instances embedded with different trained models. After verifying the solution quality on the most accurate affordable solution metric, we can finally use DFO to fine-tune the best solution(s) in the remaining online optimization budget on the metric of interest.

While we can make general observations about the trade-offs involved in both methods, these

are limited to the case study at hand. While the DFO approach is agnostic to the integrated scheduling-control formulation, its efficacy depends on the relative size of the three layers. Similarly, the surrogate approach is agnostic to the lower-level problems as long as the bottom-up responses can be mapped as a function of the top-down complicating variables. Future work could investigate and stress-test different planning network configurations, different types of scheduling formulations, and different surrogate formulations in the control. The surrogate technique, in particular, opens up numerous avenues of research, all centered around ensuring robust model performance when selecting and training model architectures.

Even though the relative performance of both approaches depends on the case study at hand, we believe that the cornerstone of DFO, emulating the hierarchical multi-level objective via black-box evaluations, deserves increased attention and consideration. When information about lower levels is available, practitioners should endeavour to check solution quality on more accurate hierarchical metrics, even if they opt for single-level or surrogate-embedded formulations for the sake of tractability.

Ultimately, we demonstrate the maturity of data-driven techniques and their ease of integration into conventional optimization frameworks. This prompts us to reconsider their potential in tackling more accurate hierarchical formulations, even in larger-scale industrial case studies.

5.5.1 Looking back

The data-driven hierarchical integration was the most fun out of the four chapters. Starting from a challenging problem formulation, we explored which data-driven techniques would best balance solution accuracy and tractability. This required integrating machine learning, optimization, and data science pipelines. While the focus of this work is primarily on data-driven approaches, future improvements could include investigating the modified DFO approach as presented in Section 5.4.4, and comparing against mathematical decomposition techniques for bi- and multi-level problems [125, 297], similar to how we benchmark against distributed optimization in Chapter 4.

At this stage, I had thoroughly benchmarked DFO solvers across typical chemical engineering applications and investigated their potential in coordinating geographically or hierarchically distributed agents. In the context of data-driven coordination and hierarchical integration, we discovered that CUATRO was not the most suitable DFO solver for budget-constrained, high-dimensional problems. Instead, DIRECT-L and Py-BOBYQA generally demonstrate superior

performance in the higher-dimensional data-driven coordination case studies, and in the current chapter, we only consider Py-BOBYQA.

This led me to realize that, when attempting to use DFO for coordinating integrated EWO problems, there is often a way to cheaply obtain a good starting point that is already close to an optimum, for example through sequential decomposition. Consequently, this compelled us to investigate whether we can improve the exploitation capabilities of the CUATRO solver from Chapter 3 to outperform Py-BOBYQA in high-dimensional case studies.

6

Scaling to higher dimensions via trust region optimization in informative linear embeddings

Contents

6.1 Sampling heuristics for improving trust-region DFO 180
6.1.1 Exploration 180
6.1.2 Constraint satisfaction 183
6.2 Extension to high dimensions 184
6.2.1 Introduction 185
6.2.2 Methodology 186
6.2.3 Case studies 191
6.2.4 Results and discussion 194
6.3 Conclusion 196
6.3.1 Looking back and ahead 196

In this chapter, we first discuss some of the improvements made to CUATRO since its introduction in Chapter 3. Then, we revisit the value chain coordination and hierarchical decision-making settings as introduced in Section 1.2 that inspired the methodologies in Chapters 4 and 5. We finally come full circle by proposing CUATRO_PLS as an extension to CUATRO focused on high-dimensional derivative-free optimization (DFO) problems, where intrinsic structure is suspected in the solution space. Section 6.2 builds on the conference proceedings published in [388]. Permission to reuse Figures 6.5, 6.6, and 6.7 is granted by Elsevier via the Copyright Clearance Center. The newest version of the CUATRO solver is available on Github:

(`https://github.com/OptiMaL-PSE-Lab/cuatro`).

6.1 Sampling heuristics for improving trust-region DFO

While the performance of CUATRO in Chapter 3 was promising, its performance was competitive with other solvers only in specific niche areas of chemical engineering. Here, we discuss sampling heuristics to improve exploration and constraint satisfaction.

6.1.1 Exploration

Heuristics

Before improving global exploration, we first focus on improving sampling within trust regions. Instead of random sampling, we implement maximum closest distance sampling (MCD_sampling), which could be formulated as such:

$$\max_{\mathbf{x}\in\mathcal{B}\cap\mathcal{T}(x^k,\delta^k)} \min_{i=1..n_{\mathcal{D}}} \quad ||\mathbf{x}-\mathbf{x}_i||_p \tag{6.1}$$

The max-min formulation (6.1) looks for the candidate $\mathbf{x}$ that is furtherst away from the closest sample $\mathbf{x}_i$ in the dataset X according to the L_p norm, making it a highly nonconvex optimization problem. We initially tried reformulating (6.1) with the Manhattan (L_1) distance into an integer linear programming formulation similar to Thebelt et al. [370], but the optimality guarantee to get the global optimum to said formulation was not worth the increase in computational overhead. We found that the following heuristic works well in practice: we uniformly sample points throughout the trust region, and query the point that is furthest away from the closest evaluated data sample.

Inspired by Thebelt et al. [370], rather than navigate exploitation and exploration by alternating between the exploitative surrogate update step and an explorative sampling step, we also considered adding mixed-integer linear formulation (6.1) to the surrogate minimization objective (3.14). This would, loosely speaking, correspond to an acquisition function with a distance-based uncertainty measure. However, we did not see any performance improvements despite considerable degradation in speed. We would argue that the reasons for this are two-fold: sacrificing computational time for more informed surrogates might only be justified when surrogates are constructed over the whole solution space rather than small trust regions. Additionally, one of the reasons popular acquisition

functions in BO are so successful is because they can be derived rigorously from first principles and appropriate choices of utility functions using Bayesian statistical learning and information theory [143]. Merely adding an 'exploration term' to the objective muddles this theoretical foundation.

At this stage, we can focus on further exploitation or exploration. In sampling_region, we follow a separate sampling region strategy. Mathematical details and rules on how to update the sampling region can be found in the work that we adopted our strategy from [121]. Having a separate sampling region contained within the trust region should in principle allow for larger steps near optima.

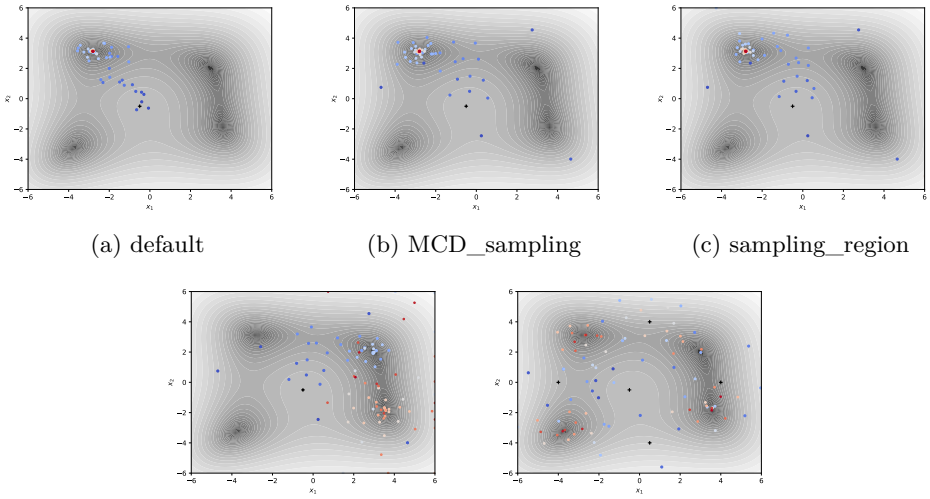
Alternatively, we can opt for more global exploration heuristics. In TIS (trajectories in sequence), we reset the center and radius of the trust region after a minimum size is reached or after a certain number of unsuccessful steps, which is similar to the restart heuristic that Py-BOBYQA uses. In TIP (trajectories in parallel), we simultaneously set up separate trust regions from multiple starting points. At each iteration, we update each trajectory, unless a trajectory is cancelled due to overlap with a more promising trajectory.

Results

Figure 6.1 compares the convergence of the different CUATRO variations in the two-dimensional solution space of the Himmelblau function. We use the Himmelblau function as a relatively well-behaved toy problem with four optima of the same quality. Figure 6.2 explores their average exploitation-exploration trade-off over five runs and includes the SciPy [393] implementations of COBYLA [306] and Nelder-Mead [276] as exploitative, and latin hypercube sampling (LHS) as explorative references. We quantify the extent of exploration as the average distance of each sample to its five nearest neighbors. Other methods like grid coverage could be used if we were purely interested in 'global exploration'.

We can see that both MCD_sampling and sampling_region heuristics visibly increase exploration in the solution space. This is confirmed by a doubling of their average nearest neighbor distance from 0.2 to 0.4, whereas said metrics are around 0.15 and 0.5 for the exploitative and explorative references respectively. Introducing a separate sampling region improves convergence close to the optimum as expected, and manifests as an improvement in the optimum found by almost an order of magnitude. Qualitatively, both trajectories look similar, with the separate sampling region run seemingly being more tightly centered around the optimum.

In Figure 6.1d, the TIS trajectory first converges to the optimum on the upper right before converging to the one on the lower right after a restart. In the TIP run in Figure 6.1e, three out of the five initial trust region instances converge to separate optima with the other two getting annulled due to overlap. The TIS and TIP variations display the most exploration, with a higher exploration metric than LHS.



(a) default (b) MCD_sampling (c) sampling_region

(d) TIS (e) TIP

Figure 6.1: Two-dimensional solution space convergence plots of CUATRO variations on the Himmelblau function. Black crosses denote starting points. Early samples are blue, later samples red.

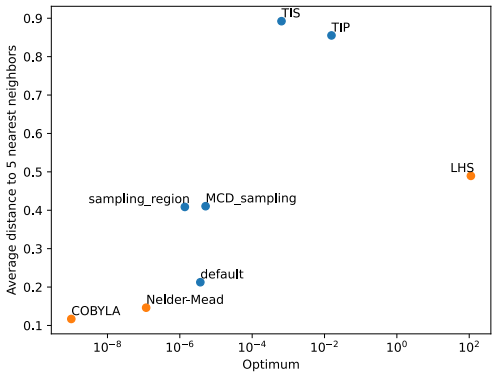


Figure 6.2: Exploration (y-axis: Average distance to nearest neighbours) versus exploitation (x-axis: Optimum) on the Himmelblau function: The higher the distance to the nearest neighbors, the more explorative. Blue samples represent CUATRO variations, and orange samples represent reference solvers

6.1.2 Constraint satisfaction

Heuristic

One of the selling points of CUATRO is its explicit constraint handling capacity, i.e. the fitting and optimization of constraint evaluation surrogates for safe exploration. While this helps with constraint satisfaction in the update step, the random sampling step is still agnostic to constraints. We propose the feasible_sampling routine, which builds on (6.1) but constrains the solution on the n_g constraint surrogates $\hat{g}_l$ being feasible at the most recent iteration k:

$$\begin{aligned} \max_{\mathbf{x}\in\mathcal{B}\cap\mathcal{T}(x^k,\delta^k)} \min_{i=1..n_\mathcal{D}} \quad & ||\mathbf{x}-\mathbf{x}_i||_p \\ \text{s.t.} \quad & \hat{g}_l^k(\mathbf{x}) \leq 0, \quad l=1..n_g \end{aligned} \tag{6.2}$$

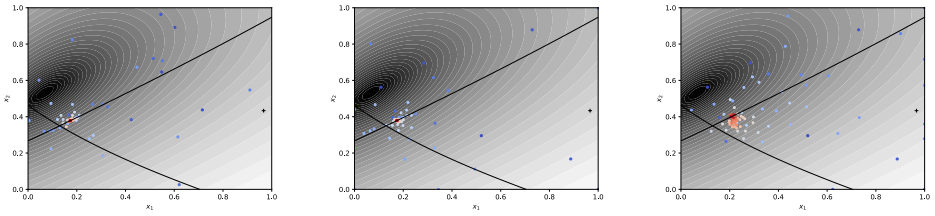
We solve this using the same heuristic as in MCD sampling, but first filter out the points whose constraint surrogate evaluations are infeasible.

Results

In Figure 6.3, we again plot the two-dimensional convergence of the default or local CUATRO variation, MCD_sampling, and feasible_sampling. In Figure 6.4, we benchmark the average performance of the three CUATRO variations over five runs on the constrained, deterministic real-time optimization case study from Chapter 3. We also include the best-performing algorithm from Table 3.3 for each variation of exploitation/exploration and model-based/direct DFO: Py-BOBYQA [96], an in-house implementation of constrained Bayesian optimization (BO), Nelder-Mead [276], and DIRECT-L [137].

In going from the default local CUATRO to the MCD variation, we see an increase in solution quality accompanied by a drop in constraint satisfaction of ten percentage points. While all three CUATRO variations are competitive with BO and Py-BOBYQA, the improved sampling strategy of MCD makes it the best-performing algorithm with an optimum at -75.8. However, it also demonstrates poor constraint satisfaction at 30%, only slightly better than the exploitative penalty methods Nelder-Mead and Py-BOBYQA. The feasible_sampling routine on the other hand improves constraint satisfaction by almost 20 percentage points while retaining a good optimum close to that of the default variation.

Figure 6.3c illustrates how despite its many initial constraint violations, feasible_sampling approaches the optimum more cautiously. Using initial trust region sampling instead of LHS and decreasing the initial radius would also improve the constraint satisfaction of all variations.



(a) default/local (b) MCD_sampling (c) feasible_sampling

Figure 6.3: Two-dimensional solution space convergence plots of CUATRO variations on the constrained, deterministic real-time optimization case study from Section 3.4.3

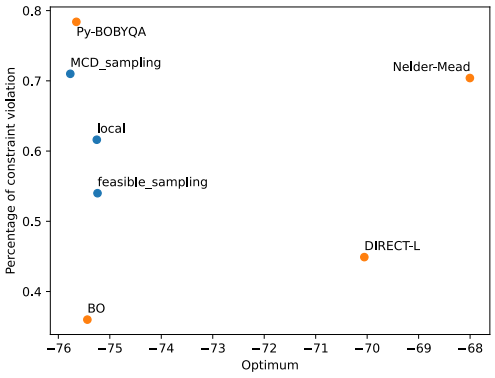


Figure 6.4: Average constraint violation versus average best optimum found for each algorithm on the deterministic real-time optimization case study from Section 3.4.3. Blue samples represent CUATRO variations, and orange samples reference solvers

We refactor the above findings into an updated version of CUATRO local called 'CUATRO_base', where we use maximum closest distance as the default sampling method within trust regions. The algorithm is described in detail in 9, and is compatible with any of the previously described exploration strategies.

6.2 Extension to high dimensions

Maintaining the benefits of derivative-free optimization (DFO) in higher-dimensional decision spaces presents challenges for existing optimization methods. We introduce CUATRO_PLS - an extension of the CUATRO quadratic trust region optimizer that leverages intrinsic structure across high-dimensional black-box variables.

6.2.1 Introduction

In Chapter 3, we review the use of DFO as an effective tool for optimizing chemical engineering systems without explicit gradient expressions. We discuss how DFO solvers usually scale poorly in the number of decision variables and their evaluation budget. Existing methods for high-dimensional black-box optimization (BBO) usually rely on exploiting structure in the solution space and optimizing over more informative subspaces. This can take the form of automatic relevance determination [33], exploiting axis-aligned sparsity in the decision variables [126], exploiting additive structure via decomposition schemes [141], deep kernel learning [401], neural embeddings [354], or linear embeddings [399].

Despite their simplicity, sketching techniques (Section 2.2.3) and linear embeddings have shown great potential in the high-dimensional BBO literature. This often involves random (Gaussian) projections to find embeddings over which to construct and optimize surrogates. These surrogates can take the form of Gaussian processes in Bayesian optimization (BO) [231] or linear-quadratic trust region models [70]. While end-to-end learning of linear [144] or nonlinear embeddings [267] with Gaussian processes is possible, limited evaluations can compromise performance.

In the remainder of this chapter, we introduce CUATRO_PLS, which integrates end-to-end learning concepts into the scalable Convex qUAdratic Trust Region Optimizer (CUATRO), first proposed in Section 3.3. At each iteration, Partial Least Squares regression (PLS) [300] identifies the linear embedding from which to best predict the output of all evaluations within a trust region. Surrogate fitting and minimization are then performed within this subspace and the minimization candidate is evaluated after reconstruction to the original space. This approach maintains the tractability of working with convex quadratic surrogates in low dimensions, while enhancing the quality of the subspaces as compared with Gaussian projections by incorporating response information.

We benchmark CUATRO_PLS against a random baseline and six State-Of-The-Art (SOTA) DFO solvers. We first use a synthetic toy problem with up to 1,000 dimensions and a known linear latent space to show that CUATRO_PLS shows consistent convergence to a high-accuracy solution and is multiple orders of magnitude faster than other model-based methods. We then assess all algorithms on four high-dimensional chemical engineering problems exhibiting intrinsic structure.

In Section 6.2.2, we discuss latent embeddings in DFO and introduce our algorithm CUATRO_PLS, before describing our case studies in Section 6.2.3. In Section 6.2.4, we then show that

CUATRO_PLS ranks among the top three algorithms on all problems and outperforms competitors in high-dimensional scenarios favoring exploitation over exploration.

6.2.2 Methodology

Derivative-free optimization and latent structure

We start again by considering a generic DFO problem formulation of the following form:

$$\min_{\mathbf{x}\in\mathcal{B}} f(\mathbf{x}),$$

where $\mathbf{x} \in \mathcal{B} \subseteq \mathbb{R}^{n_x}$ denotes the variables within box bounds $\mathcal{B}$ used to optimize the expensive black-box objective $f(\cdot)$. For simplicity, we assume that any constraints are handled implicitly via the penalty method. Subspace optimization relies on the premise that there exists a pair of encoder-decoder transformations $\mathcal{E} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_z}$ and $\mathcal{R} : \mathbb{R}^{n_z} \to \mathbb{R}^{n_x}$ that allow a projection from an ambient solution $\mathbf{x}$ to a lower-dimensional embedding $\mathbf{z}$ and a reconstruction back. The use of $\mathbf{z}$ in this chapter to refer to subspace variables should not be confused with its use in Chapter 4 to denote complicating, or shared variables. This allows us to query the black-box objective:

$$\min_{\mathbf{x}\in\mathcal{B}} f(\mathbf{x}) \approx \min_{\mathbf{z}} f(\mathcal{R}(\mathbf{z})) \text{ s.t. } \mathcal{R}(\mathbf{z}) \in \mathcal{B}.$$

By constructing and optimizing surrogate models $m(\cdot)$ in the lower-dimensional space, we can thus 'break' the curse of dimensionality when $n_z \ll n_x$. In finding adequate encoder-decoder transformations, we need to balance their accuracy in reconstructing original inputs from latent embeddings ($X \approx \mathcal{R}(Z) = \mathcal{R}(\mathcal{E}(X))$) with the accuracy of surrogates $m(\cdot)$ in predicting outputs from the latent space ($\mathbf{y} \approx m(Z)$). A lot of high-dimensional DFO and BO techniques differ only in the projection and reconstruction techniques used, the choice of surrogates in the latent space, and how these are updated.

Cartis and Roberts [70] combine sketching techniques with linear-quadratic trust-region DFO solvers. While random embeddings preserve theoretical guarantees useful for convergence proofs, we believe that we can integrate response information to improve the sample efficiency of high-dimensional trust region methods.

Partial Least Squares (PLS)

PLS regression is a dimensionality reduction and regression technique that projects the input features $X \in \mathbb{R}^{n_{\mathcal{D}} \times n_x}$ into lower-dimensional embeddings $Z \in \mathbb{R}^{n_{\mathcal{D}} \times n_z}$, such that the latter explain as much variance as possible in X, and maximize the covariance between the latent variables Z and the response $Y \in \mathbb{R}^{n_{\mathcal{D}} \times 1}$. Formally, this corresponds to the following optimization problem:

$$\begin{aligned} \max_{Z,M,P,Q,E,F} \quad & \mathrm{Cov}(Z, Y) \\ \text{s.t.} \quad & X = ZP^\intercal + E, \quad Y = ZQ^\intercal + F, \quad Z = XM, \end{aligned} \tag{6.3}$$

where $P \in \mathbb{R}^{n_x \times n_z}$ and $Q \in \mathbb{R}^{n_z \times 1}$ are the loading matrices for X and Y, respectively, $M \in \mathbb{R}^{n_x \times n_z}$ the projection, also called weight, matrix that defines how X is projected onto the latent space, and $E \in \mathbb{R}^{n_{\mathcal{D}} \times n_x}$ and $F \in \mathbb{R}^{n_{\mathcal{D}} \times n_x}$ the residual matrices assumed to be independent and identically distributed random normal variables. The Scikit-Learn [295] implementation uses the Nonlinear Iterative Partial Least Squares (NIPALS) algorithm under the hood to solve this formulation. Instead of performing a full singular value decomposition (SVD) on X and Y, NIPALS extracts one latent component at a time, using the following steps:

1. Selecting an initial estimate for the latent scores.
2. Iteratively refining the latent variables by alternating between updating Z, M, and P.
3. Deflating X and Y by subtracting the contribution of the extracted latent component before proceeding to the next component.
4. Repeating until a specified number of components is reached or until convergence is achieved.

CUATRO_PLS

CUATRO_PLS is an extension of CUATRO that performs dimensionality reduction to find a lower-dimensional subspace over which to perform surrogate updates. Algorithm 10 describes the full CUATRO_PLS logic built on top of CUATRO_base. While the framework in principle allows for black-box constraints, their handling requires special consideration. Additionally, while CUATRO_PLS can in principle be integrated into any more global exploration heuristic, we currently only consider the 'base' strategy.

Algorithm 9: CUATRO_base

Input: Black-box objective f and constraints $g_l, l = 1..n_g$, Initial guess $\mathbf{x}^0$, Initial radius δ^0, β_{red}, β_{inc}, η_1, η_2, Minimum number of samples for fitting $N_{s,min}$, Maximum number of function evaluations $N_{f,max}$, sampling strategy $S \in$ [MCD, feasible_sampling], exploration strategy $E \in$ [base, sampling_region, TIS, TIP], Optional: minimum restart radius δ_{max}

Initialisation: Initial data sets $X^0 := [\mathbf{x}^0]$ and $Y^0 := [\{(f(\mathbf{x}^0), g_l(\mathbf{x}^0)), l = 1..n_g\}]$, sampling region radius $\delta_s^0 = \delta^0$ or $\delta_s^0 < \delta^0$ for $S =$sampling_region, Sampling region $\mathcal{T}_s(\mathbf{x}^0, \delta_s^0)$, Trust region $\mathcal{T}(\mathbf{x}^0, \delta^0)$, Bound-constrained region $\mathcal{B}$, (Optional: initialize above quantities for $t = 1..T$ trust regions); Function evaluation counter $n_f := 1$, Iteration counter $k := 0$

while $(N_{f,max} \geq n_f)$ **do**

- MCD sampling using (6.1) or (6.2) based on S until region contains $N_{s,min}$ samples;
- Compute $(P_l^k, \mathbf{q}_l^k, r_l^k)$ from (3.12) for $l = 0..n_g$ using all evaluations in $\mathcal{B} \cap \mathcal{T}_s(\mathbf{x}^k, \delta^k)$;
- Find $\mathbf{x}_{new}$ by minimizing (3.14) within $\mathcal{B} \cap \mathcal{T}(\mathbf{x}^k, \delta^k)$ using $(P_l^k, \mathbf{q}_l^k, r_l^k)$ for $l = 0..n_g$;
- Update datasets with n_s new evaluations X_{n_s}: $X^{k+1} \leftarrow [X^k, X_{n_s}], Y^{k+1} \leftarrow [Y^k, \{(f(X_{n_s}), g_l(X_{n_s})), l = 1..n_g\}]$;
- $n_f \leftarrow n_f + n_s$; $k \leftarrow k + 1$;
- **if** $E =$*base* **then**
  - Update $\mathbf{x}^k, \delta^k = \delta_s^k$ based on CUATRO local trust region scheme (5 steps 8-16) ;
- **else if** $E =$*sampling_region* **then**
  - Update $\mathbf{x}^k, \delta^k,$ and δ_s^k following [121] ;
- **else if** $E =$*TIS* **then**
  - Update $\mathbf{x}^k, \delta^k = \delta_s^k$ based on CUATRO local trust region scheme (5 steps 8-16) ;
  - If $\delta^k \leq \delta_{max}$: reset $\mathbf{x}^k, \delta^k$;
- **else if** $E =$*TIP* **then**
  - For current trust region t: Update $\mathbf{x}_t^k, \delta_t^k$ based on algorithm 5 steps 8-16 ;
  - Fathom $\mathcal{T}_t$ if for any $t' \in [1..T] \setminus \{t\}$: $\mathcal{T}_t \cap \mathcal{T}_{t'} \wedge \min_{y \in Y_t} > \min_{y \in Y_{t'}}$, ;
  - Pick next trust region $(t \leftarrow t')$ among remaining available t';

end

Algorithm 10: CUATRO_PLS

Input: Black-box objective f and constraints $g_l, l = 1..n_g$, Initial guess $\mathbf{x}^0$, Initial radius δ^0, β_{red}, β_{inc}, η_1, η_2, Minimum number of samples for fitting $N_{s,min}$, Maximum number of function evaluations $N_{f,max}$, dimensionality of latent embedding n_{PLS}

Initialisation: Initial data sets $X^0 := [\mathbf{x}^0]$ and $Y^0 := [\{(f(\mathbf{x}^0), g_l(\mathbf{x}^0)), l = 1..n_g\}]$, sampling region radius $\delta_s^0 = \delta^0$, Sampling region $\mathcal{T}_s(\mathbf{x}^0, \delta_s^0)$, Trust region $\mathcal{T}(\mathbf{x}^0, \delta^0)$, Bound-constrained region $\mathcal{B}$, Function evaluation counter $n_f := 1$, Iteration counter $k := 0$

while $(N_{f,max} \geq n_f)$ **do**

- MCD sampling using (6.1) until $\mathcal{T}_s(\mathbf{x}^k, \delta_s^k)$ contains $N_{s,min}$ samples (X_s^k, Y_s^k);
- Perform PLS using SciPy implementation. Get M, $R = M^\intercal (MM^\intercal)^{-1}$, and $Z = X_s^k M$;
- Compute $(P_0^k, \mathbf{q}_0^k, r_0^k)$ from (6.4) using (Z, Y_s^k);
- Find $\mathbf{z}_{new}$ by minimizing (6.5) using $(P_0^k, \mathbf{q}_0^k, r_0^k)$;
- Reconstruct $\mathbf{x}_{new} = \mathbf{z}_{new} R$ and update datasets with n_s new evaluations X_{n_s}: $X^{k+1} \leftarrow [X^k, X_{n_s}], Y^{k+1} \leftarrow [Y^k, \{(f(X_{n_s}), g_l(X_{n_s})), l = 1..n_g\}]$;
- $n_f \leftarrow n_f + n_s$; $k \leftarrow k + 1$;
- Update $\mathbf{x}^k, \delta^k = \delta_s^k$ based on CUATRO local trust region scheme (5 steps 8-16) ;

end

Surrogate fitting in the latent space At each iteration, PLS identifies the linear projection matrix $M \in \mathbb{R}^{n_x \times n_z}$ and reconstruction matrix $R = M^\intercal(MM^\intercal)^{-1}$ that best predict the outputs $\mathbf{y}$ of all n_D samples in Y within the original trust region $\mathcal{T}(\cdot)$ after projection to their linear embedding Z such that $Z \approx XM$ and $X \approx ZR$. The surrogates are then fitted using the embedded scores Z:

$$\begin{aligned} \min_{P \succeq 0, \mathbf{q}, r} \quad & \sum_{j=1}^{n_D} \left(m(\mathbf{z}_j) - y_j\right)^2 \\ \text{s.t.} \quad & m(\mathbf{z}_j) = \mathbf{z}_j^\intercal P \mathbf{z}_j + \mathbf{q}^\intercal \mathbf{z}_j + r, \quad \forall j = 1..n_D \text{ where } \mathbf{z}_j \subset Z, \end{aligned} \tag{6.4}$$

Note that (6.4) is convex like its counterpart in CUATRO.

Surrogate minimization in the latent space To conserve convexity in the subspace surrogate minimization, the exact reconstructed trust region constraint in the original space (of the form $(R\mathbf{z} - \mathbf{x}^k)^\intercal(R\mathbf{z} - \mathbf{x}^k) \leq \delta^2$) is replaced with a heuristic trust region in latent space centered around $\mathbf{z}^k = M\mathbf{x}^k$, where an effective radius is defined as the distance from $\mathbf{z}^k$ to the furthest projected sample as $\hat{\delta}^2 = \max_{\mathbf{z}_j \in Z} \ (\mathbf{z}_j - \mathbf{z}^k)^\intercal(\mathbf{z}_j - \mathbf{z}^k)$:

$$\begin{aligned} \min_{\mathbf{z} \in \mathbb{R}^{n_z}} \quad & m(\mathbf{z}) \\ \text{s.t.} \quad & R\mathbf{z} \in \mathcal{B}, \\ & (\mathbf{z} - \mathbf{z}^k)^\intercal(\mathbf{z} - \mathbf{z}^k) \leq \hat{\delta}^{k^2}, \end{aligned} \tag{6.5}$$

where the constraints denote the box bounds and trust region respectively. The minimization candidate of (6.5) is then evaluated and the trust region is updated after reconstruction to the original space following the original CUATRO local or CUATRO_base update schemes, see algorithm 10.

Handling of constraints In principle, this framework also allows explicit handling of black-box constraints by fitting surrogates on the constraint evaluations in the subspace and adding them to (6.5) as: $m_c(\cdot) \leq 0$. However, traditional PLS does not explicitly account for constraint evaluations in the linear subspace selection. Instead, we could use PLS2, which extends PLS to multi-target evaluations, where the constraint evaluations represent the additional outputs. This approach would have to carefully balance fitting the subspace selection on the objective versus constraint function topologies, by potential pre-processing of the input-output samples.

Dimensionality reduction considerations This methodology introduces the latent dimensionality n_z as an important hyperparameter. At fixed n_z, PLS would give the optimal embedding if we were to use linear instead of quadratic surrogates. We find that PLS works well in practice even in combination with quadratic surrogates. The intuition behind this is that the black-box function displays increasingly linear behaviour as the trust region shrinks.

More generally speaking, the idea of combining dimensionality reduction techniques with DFO is not unique to CUATRO and could be applied to any trust-region or even more generic framework and has even been considered in the PSE community [211]. Our update scheme differs from most previous works, which usually either only perform initial subspace selection using PLS, or use random embeddings adaptively.

A fundamental challenge with this approach is that we lose the theoretical convergence properties of the Johnson-Lindenstrauss lemma without random embeddings. More broadly, this highlights another shortcoming of this approach: by using PLS to identify our linear subspace, we adopt a purely exploitative strategy for subspace selection, making it highly dependent on the quality of the trust region samples. In other words, if our trust region samples fail to capture the necessary latent information, the embedding quality might suffer. Ideally, we would balance exploitation and exploration over the trust region or the whole solution space. Possible strategies might involve the incorporation of uncertainty-aware terms in the 'acquisition function' objectives akin to Bayesian optimization, or an alternating or ϵ-greedy approach, where a small probability of random embeddings is maintained to preserve some of the theoretical guarantees of the randomized approach.

Benchmarked solvers

We benchmark CUATRO against a wide range of SOTA DFO solvers and search techniques.

Py-BOBQYQA [96] is a trust-region optimizer constructing linear-quadratic surrogates using interpolation and regression techniques. The algorithm exhibits strong exploitative properties by making quick progress at the expense of exploration across Chapters 4 and 5.

DIRECT-L [137] is a search space partitioning direct DFO solver that elegantly trades off exploitation and exploration. The method is more robust to ill-conditioning in the objective as it does not rely on surrogates.

Latin hypercube sampling (LHS) is a space-filling design method. Any proposed high-

dimensional DFO should at least outperform random search in the form of LHS.

CMA-ES [161], short for covariance matrix adaptation evolution strategy, is a common benchmarking solver in the high-dimensional BBO literature. CMA-ES shines in highly nonlinear, stochastic applications with a high evaluation budget.

HEBO [100], short for Heteroscedastic and Evolutionary Bayesian Optimization solver, is the winning submission to the NeurIPS 2020 Black-Box Optimisation challenge.

TuRBO [127] is a high-dimensional Bayesian optimization solver that leverages Thompson sampling within trust regions for scalability.

ALEBO [231] revisits some design considerations of previous high-dimensional Bayesian optimization solvers relying on random embeddings like Rembo [399].

While it would be interesting to benchmark against other trust-region or more general DFO solvers in combination with PLS, we limit the scope of the benchmarking to CUATRO_PLS as it would take a considerable amount of low-level coding in other open-source frameworks to implement.

6.2.3 Case studies

We first define a synthetic high-dimensional Rosenbrock function with a known linear embedding dimensionality n_e to control the original n_h and effective n_e dimensionalities in studying the effect of the embedding dimensionality hyperparameter n_{PLS}. n_h essentially takes the place of n_x, but we insist on keeping separate notation, specifically to highlight the contrast to n_e and n_{PLS}. Then, we benchmark all algorithms on four high-dimensional problems spanning controller tuning and EWO. While we believe these to exhibit intrinsic structure, the dimensionality of the true latent embedding is unknown, and does not have to be linear.

High-dimensional Rosenbrock with low effective dimensionality

We start with a generalized Rosenbrock (RB) formulation of effective dimensionality n_e:

$$RB(\mathbf{z}; n_e) = \sum_{i=1}^{n_e-1} [100(z_{i+1} - z_i^2)^2 + (1 - z_i)^2].$$

We can create a high-dimensional RB function of ambient dimensionality n_h with a true effective

linear embedding of dimensionality n_e by adding $n_h - n_e$ 'fake' dimensions to the effective vector $\mathbf{z}$ and mixing up the informative and uninformative inputs using an orthogonal Gaussian matrix $G \in \mathbb{R}^{n_h \times n_h}$. Conversely, we define our 'fake' high-dimensional Rosenbrock function as follows:

$$RB_{fake}(\mathbf{x}; n_h, n_e) = RB(\mathbf{z}; n_e), \quad \mathbf{z} = [\tilde{x}_1, \ldots, \tilde{x}_{n_e}], \quad \tilde{\mathbf{x}} = G\mathbf{x}.$$

PID Controller Tuning

This case study presents the classical dynamic control of a Continuous Stirred Tank Reactor (CSTR) equipped with a cooling jacket. In this system, the inlet flow rate $F_{in}(t)$ and the cooling jacket temperature $T_c(t)$ are the manipulated variables, and the reactor temperature $T(t)$ is the controlled variable. The CSTR facilitates the following reactions where A is the reactant, B is the intermediate, and C is the product: $A \to B \to C$.

The objective is to maintain the control variables at their set point using a Proportional-Integral-Derivative (PID) controller. The PID controller $\mathcal{C}(\cdot)$ adjusts the manipulated variables $\mathbf{u}(t) = [F_{in}(t), T_c(t)]^\intercal$ based on the deviation $\mathbf{e}(t) = \mathbf{x}(t) - \mathbf{x}_{sp}$ of the five state variables $\mathbf{x}(t)$ (reactor temperature, volume, and mass of reactant, intermediate, and product) from their setpoints $\mathbf{x}_{sp}$. As opposed to PI controllers as introduced in Section 3.4.6, PID controllers also account for instantaneous changes in the error $\mathbf{e}'(t)$:

$$\begin{aligned} i \in \{F_{in}, T_c\}: \quad u_i(t) \quad &= C(\mathbf{e}(t); \mathbf{K}_{P,i}, \mathbf{K}_{I,i}, \mathbf{K}_{D,i}, K_i) \\ &= K_i + \sum_{i'=1}^{5} [K_{P,i,i'} e_{i,i'}(t) + K_{I,i,i'} \int_{\tau=t_0}^{t} e_{i,i'}(t)\, d\tau + K_{D,i,i'} e'_{i,i'}(t)], \end{aligned} \tag{6.6}$$

where $\mathbf{e}'(t) = \frac{\partial \mathbf{e}(t)}{\partial t}$. The optimization problem then consists of finding the $n_h = 32$ controller gains that optimize the weighted sum of the reactor temperature squared error in the state variables with a term penalizing magnitude and changes in the controls. The mathematical description of the system can be found in Appendix D.1.1.

Neural network control policy tuning

The neural network control policy tuning problem investigates the same reactor control setup as in the previous PID controller tuning case study with some key distinctions: The inlet flowrate is fixed so that the jacket cooling temperature $T_c(t)$ becomes the only manipulated variable $u(t)$.

In addition to the reactor temperature $T(t)$, we also control the concentration of reactant $c_A(t)$. Instead of implementing PID control, we define a parameterized explicit control law $\mathcal{C}_\theta(\cdot)$ to return the manipulated variable $T_c(t)$ as a function of $T(t)$ and $c_A(t)$:

$$T_c(t) = \mathcal{C}_\theta(c_A(t), T(t)). \tag{6.7}$$

We then want to find the $n_h = 61$ neural network weights and biases θ that result in the optimal control of $T(t)$ and $c_A(t)$ in pursuing a similar objective to the previous case study. We suspect the parameters of the neural network to exhibit low effective dimensionality.

Value chain coordination

This case study applies the data-driven coordination methodology from Chapter 4 on the planning formulation from Appendix C.1.1 under the privacy- and agency-preserving considerations as described in the value chain coordination motivating example (Section 1.2). By inspection of the mathematical formulation, we discover that the problem becomes separable into three site-specific subproblems after fixing the material transport variables connecting site S1 with sites S2 and S3. We introduce local copies of the material transport variables into each subproblem, and penalize deviations from the suggested coordinated variables in each subproblem objective, as described in the data-driven coordination methodology (Section 4.2.4).

The goal then becomes finding the 72 material transport variables that optimize the sum of the sites' planning profits, which can be obtained as the result of privacy-preserving subproblem solutions. Following Chapter 4, we also include the distributed optimization algorithm ADMM as described in Section 4.2.3. We believe that effective embeddings are caused by temporal and spatial correlations between material flows.

Hierarchical planning-scheduling

The hierarchical planning-scheduling case study corresponds to the bi evaluation from Chapter 5 as described in Section 5.3.3. We solve for a set of $n_h = 172$ material production, transport, and sales variables that optimize a bi-level planning-scheduling objective. If only a single level was considered, the optimization formulation would be equivalent to the planning model considered in the value chain coordination. However, in data-driven coordination, we fix only the complicating variables necessary to extract all other quantities from the subproblem optimization solutions; in

the hierarchical planning-scheduling, we fix all the variables required to turn the planning problem into a pure simulation. Intrinsic subspaces are again believed to arise from temporal correlations.

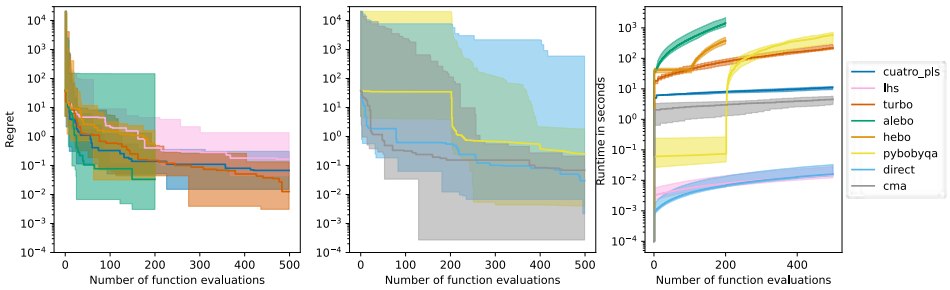
6.2.4 Results and discussion

Synthetic low-dimensional case study

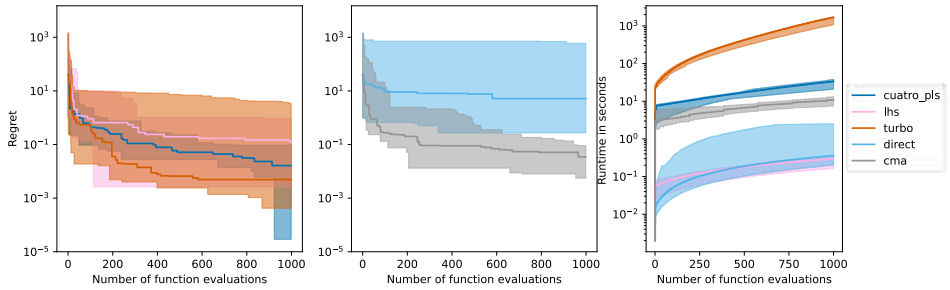
On the 100-dimensional Rosenbrock instance in Figure 6.5a, CMA-ES, ALEBO, and TuRBO display the best convergence. CUATRO_PLS seems to be competitive with other SOTA solvers, with a median and worst-case final convergence comparable to that of CMA-ES. As expected, CUATRO_PLS has a runtime at least an order of magnitude lower than that of other model-based DFO solvers while displaying comparable performance. HEBO and ALEBO are not investigated further given their exploding runtime on high-dimensional, high-budget problems. When the dimensionality and budget are then increased to 1,000 in Figure 6.5b, we see that CUATRO_PLS, despite displaying a slightly worse median run than TuRBO, displays the best final run of all samples and seems to be the most consistent. Additionally, while TuRBO's runtime increases by an order of magnitude when doubling the budget, CUATRO_PLS's runtime only doubles.

Sensitivity analysis on the embedding dimensionality n_{PLS}

Figure 6.6 investigates the effect that the embedding dimensionality n_{PLS} has on CUATRO_PLS performance when other hyperparameters are fixed to their default. We keep the RB original dimensionality at 1,000, and the true effective dimensionality at 10. We see no major difference in performance when perfectly or overestimating the true latent dimensionality at six or higher. To the contrary, the best performance is found when underestimating the effective dimensionality at two and six. This suggests that CUATRO_PLS stands as a powerful high-dimensional DFO solver, even if no knowledge about the intrinsic dimensionality is suspected. It might even be encouraged to use CUATRO_PLS in fewer effective dimensions. In other words, there is merit to the idea of performing surrogate optimization in the most informative linear combination of dimensions, rather than capturing all information.



(a) $n_h = 100$ and $n_e = 2$



(b) $n_h = 1000$ and $n_e = 2$

Figure 6.5: Regret versus evaluation budget convergence plot for the synthetic high-dimensional Rosenbrock case study at constant effective dimensionality $n_e = 2$, and ambient dimensionality $n_h = 100$ (top) and $1,000$ (bottom) for model-based (left) and direct search (center) DFO methods. The corresponding solution times are displayed on the right. Reprinted from van de Berg et al. [388] with permission from Elsevier.

Chemical engineering applications

Throughout all case studies in Figure 6.7, the CUATRO_PLS runs (on the left) generally display consistency across different hyperparameters and embedding dimensionalities. This supports our hypothesis that CUATRO_PLS performs well even when underestimating the effective dimensionality in the absence of problem-specific knowledge about embedding structure. In the PID controller tuning in Figure 6.7a, TuRBO outshines all other methods, but CUATRO_PLS remains competitive with Py-BOBYQA and CMA-ES. In the policy training in Figure 6.7b, CUATRO_PLS outperforms most methods and is only beat in consistency and in the median run by CMA-ES and TuRBO respectively. In the value chain coordination in Figure 6.7c, most CUATRO_PLS runs outperform the best DFO solvers and even the distributed optimization benchmark ADMM. In the hierarchical integration case study in Figure 6.7d, CUATRO_PLS and Py-BOBYQA are the only methods that make any progress. While CUATRO_PLS makes significant initial progress, Py-BOBYQA manages to better fine-tune the solution. Overall, the above findings are not surprising: CUATRO_PLS, as a high-dimensional quadratic trust region optimizer, excels in finding good nearby optima quickly. However, CUATRO_PLS remains a competitive optimizer even in

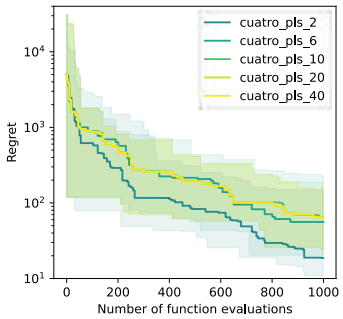


Figure 6.6: Regret versus evaluation budget convergence plot on the 1000-dimensional Rosenbrock with an effective dimensionality of ten ($n_e = 10$), when CUATRO_PLS is used with different embedding dimensionalities ($n_{PLS} = 2, 6, 10, 20, 40$ in the labels) and other hyperparameters are fixed to their default. Reprinted from van de Berg et al. [388] with permission from Elsevier.

the policy tuning, where more explorative methods like CMA-ES are encouraged.

6

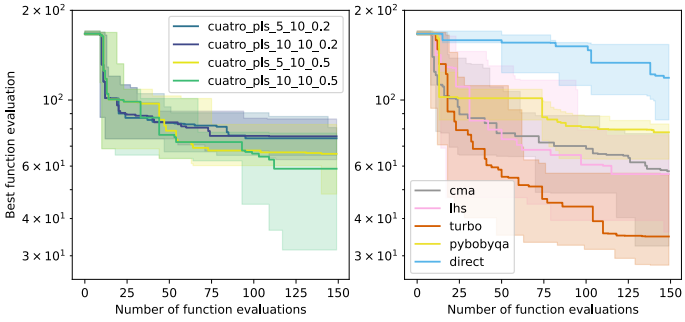
6.3 Conclusion

We improve the exploration and constraint satisfaction capabilities of the CUATRO solver as introduced in Chapter 3 by implementing heuristic sampling strategies: maximum closest distance sampling, separating the sampling region from the trust region, trajectory restarts, trajectories in parallel, and leveraging constraint surrogates for feasible sampling.

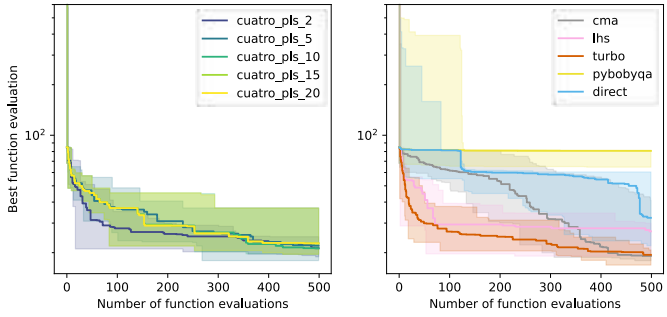
We then introduce CUATRO_PLS as a scalable high-dimensional extension to the DFO solver. We use learned linear structures in the decision variables to display competitive performance with SOTA alternatives at significantly lower runtime compared with other model-based methods. We benchmark our solver on four high-dimensional chemical engineering DFO problems where intrinsic dimensionality is suspected, and highlight its strengths in fine-tuning solutions and finding good optima quickly. CUATRO_PLS, by performing surrogate optimization in the most informative linear embedding dimensions, can even be used if no intrinsic dimensionality is known or suspected.

6.3.1 Looking back and ahead

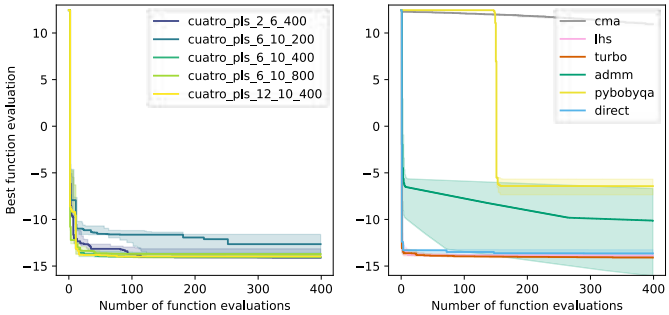
This chapter naturally concludes the thesis by building on the solver we introduced in Chapter 3 and enhancing its performance on EWO case studies from Chapters 4 and 5. In these chapters, we established that DFO solvers are in principle viable alternatives to distributed optimization and optimality surrogates if their sample efficiency and scaling can be improved. To address these issues, the sampling heuristics and subspace optimization techniques introduced here make



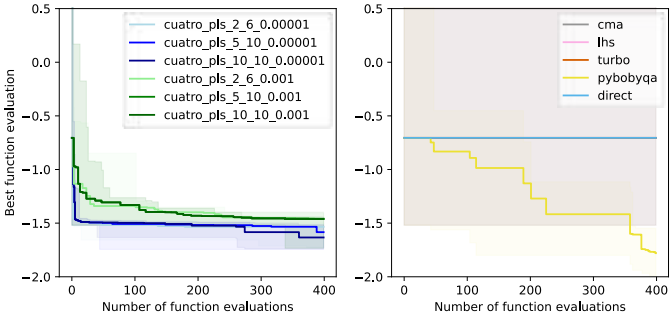
(a) 32-dimensional PID controller tuning



(b) 61-dimensional neural network control policy tuning



(c) 72-dimensional value chain coordination



(d) 172-dimensional hierarchical planning-scheduling

Figure 6.7: Best function evaluation versus evaluation budget convergence plots on high-dimensional chemical engineering case studies. Left: CUATRO with different hyperparameters (embedding dimensionality, number of trust region samples, and trust region initial radius). Right: Other competitive solvers. Reprinted from van de Berg et al. [388] with permission from Elsevier.

CUATRO competitive with the best DFO solvers for these problems. Furthermore, the results indicate that CUATRO_PLS is a powerful solver beyond these case studies, excelling in high-dimensional spaces with a thousands variables when exploitation is prioritized over exploration.

Current work seeks to improve the exploitation-exploration trade-off of CUATRO_PLS by incorporating subspace optimization into a 'trajectories in parallel' framework similar to that of TuRBO, where trajectories are chosen based on a multi-armed bandit strategy. Additionally, it should be investigated if 'quadratic PLS' [185], as an 'exact' end-to-end dimensionality reduction technique, could improve current performance.

Beyond safe exploration and high-dimensional continuous domains, many problems in chemical engineering involve discrete decisions [424] or allow for asynchronous or parallel batch evaluations [131]. Extending CUATRO to these contexts could present another opportunity for tailoring CUATRO to the peculiarities of chemical engineering applications.

Conclusions

This thesis investigates how data-driven methods - black-box optimization to be precise - can be used to integrate manufacturing and supply chain operations under organizational considerations. We first highlight the key contributions and observations of each chapter. Then, we provide a condensed retrospective of the key contributions in Section 6.3.2, before discussing some potential future directions in Section 6.3.3.

In Chapter 1, we trace the evolution of major developments in the Process Systems Engineering (PSE) literature, from early systems thinking to Industry 4.0 and the latest advances in artificial intelligence (AI). We introduce the concept of Enterprise-Wide Optimization (EWO) with the overarching goal of integrating all decision-making within chemical companies. A recurring value chain case study illustrates two business scenarios where traditional EWO tools fall short: distributed planning, where organizational constraints limit information sharing and restrict the choice of efficient numerical solvers; and hierarchical planning-scheduling-control, where accurately modeling information flow between hierarchical agents leads to a tri-level formulation that remains intractable, even for our small motivating example.

In Chapter 2, we examine why mathematical programming has emerged as the dominant decision-making framework in EWO. We explore canonical problem formulations from the supply chain management, operations research (OR), and control literature, and present various decision-making frameworks, including reordering policies, simulation optimization, local search, constraint programming, explicit control laws, reinforcement learning, and model predictive control. We highlight the advantages of mathematical programming in terms of modeling flexibility, theoretical convergence guarantees, and specialized solvers, for the integration of planning, scheduling, and control .

Next, we introduce essential mathematical optimization theory, emphasizing decomposition algorithms and their relation to mathematical structure. We also provide high-level explanations of how optimization solvers function. We follow up by exploring various Machine Learning (ML) paradigms and their application to optimization. These include ML optimality or feasibility surrogates, subspace optimization over learned manifolds, and derivative-free optimization (DFO) of black-box systems. Although not the focus of this thesis, we also touch on other paradigms in

the broader field of 'Learning to Optimize', and offer personal perspectives on the potential and limitations of recent generative AI (GenAI) advances.

We argue that pure ML approaches struggle to consistently outperform traditional tools across different problem instances, particularly in combinatorial optimization (CO), where established methods exploit problem structure for efficiency. However, identifying 'pockets of computational reducibility' within mathematical optimization workflows that ML tools can exploit presents significant opportunities for improving tractability. Lastly, we argue that the current potential of the latest GenAI advancements lies in their integration into decision-making pipelines to complement, rather than replace, human workflows. This is especially true given the unique challenges of organizational decision-making in chemical companies. Ensuring value alignment is crucial to making AI tools safe and compatible with human and business values as their capabilities expand.

In Chapter 3, we conduct a comprehensive benchmarking study comparing state-of-the-art derivative-free optimization (DFO) algorithms with finite difference-based methods across various chemical engineering applications, including flowsheet optimization, real-time optimization, and controller tuning. Central to this chapter is CUATRO, a convex quadratic trust region optimizer designed for expensive, noisy process systems engineering (PSE) problems. CUATRO's explicit constraint-handling capabilities make it particularly suitable for safety-critical applications. We also explore sample average approximation with conservative constraint handling for safer exploration in noisy environments.

Our benchmarking reveals that DFO performance varies significantly with problem characteristics, such as dimensionality, black-box function conditioning, and the presence of constraints. No single algorithm excels across all scenarios, emphasizing the importance of selecting methods tailored to specific problem demands. Finite difference-based approaches generally break down in noisy or high-dimensional settings. Model-based methods like BO, CUATRO, and Py-BOBYQA excel in smoother spaces, though each has unique strengths: BO is especially sample-efficient in low-data, low-dimensional regimes, and has built-in routines for navigating exploitation and exploration; Py-BOBYQA displays fast convergence to local optima, but is sensitive to noise; CUATRO can be configured to display predominantly exploitative (local) or explorative (global) behaviour. Direct search methods like DIRECT-L offer a robust alternative when no prior problem knowledge is available, balancing locally biased exploitation and partition-based exploration of the solution space at minimal computational overhead. While model-based methods were initially believed to display better sample efficiency, direct methods prove more resilient to ill-conditioning.

We equip practitioners with practical insights into Python-based DFO implementations to inform

their choice of solver based on problem and desired solution characteristics. The lessons learned here, particularly in handling failing subproblem optimizations and selecting solvers under tight evaluation budgets, directly inform our experiments using DFO to coordinate 'black-box' optimal subproblem solutions in EWO applications.

In Chapter 4, we introduce a data-driven coordination framework for EWO subproblems, addressing organizational constraints that make centralized formulations impractical. We use DFO to optimize the shared linking variables that minimize the sum of optimal subproblem objectives, allowing agents to maintain autonomy, privacy, and flexibility in moving away from greedy Nash equilibria towards Pareto-optimal solutions. We benchmark our approach against ADMM across three case studies: collaborative learning, facility location, and multi-objective blending. Each case study includes a sensitivity analysis focused on case-specific considerations, like the number of agents, number of complicating variables, subproblem size, and subproblem convexity.
Our results show that DFO methods, particularly CUATRO, Py-BOBYQA, and DIRECT-L, can outperform ADMM in two different scenarios: in using explorative DFO solvers for low-dimensional, multi-modal solution spaces where ADMM struggles to overcome local optima; or in fine-tuning good initial solutions with exploitative solvers to improve solution accuracy. We also argue that the relative performance of DFO and ADMM solvers is less dependent on subproblem size, provided other factors like solution space topology and dimensionality are controlled.
Our framework does not rely on the numerical optimization of known subproblem model expressions, and is consequently particularly useful when subproblems are subject to information sharing constraints, or not solved to global optimality. While identifying industrially relevant cases with a manageable number of complicating variables can be challenging, our framework demonstrates a creative integration of data-driven and traditional optimization pipelines to address real business needs.

In Chapter 5, we tackle the integration of planning, scheduling, and control within a hierarchical framework, which often leads to computationally intractable multi-level formulations. We propose a novel approach that combines DFO with optimality surrogates to create a tractable solution that outperforms traditional sequential methods in a multi-site, multi-product value chain.
We present a step-by-step workflow comparing DFO and surrogate approaches, highlighting differences in how they leverage integrated scheduling-control data. We explore their trade-offs between solution accuracy and time, the impact of parallelization in DFO, and the sensitivity of optimality surrogate performance to model architectures. We also propose a hybrid approach that combines the advantages of the DFO and surrogate techniques in terms of accuracy and tractability respec-

tively, leveraging both on- and offline solution time to its fullest. Our method is flexible, retains the status quo heuristic sequential solution as a fallback, and is agnostic to specific scheduling and control formulations once linking variables are identified.
We innovate by combining DFO with data-driven techniques to address tractability concerns in accurately modelling hierarchical information flows. We demonstrate the maturity and potential of these methods and prompt their re-evaluation on complex, large-scale industrial problems.

In Chapter 6, we enhance the CUATRO solver from Chapter 3 for high-dimensional DFO applications, specifically the data-driven coordination and hierarchical integration problems from Chapters 4 and 5. CUATRO_PLS leverages Partial Least Squares (PLS) regression to include response information in identifying more informative linear subspaces over which to optimize, reducing computational complexity during trust region surrogate updates.
Our benchmarking shows that CUATRO_PLS excels in high-dimensional controller tuning and EWO problems, especially in budget-constrained scenarios, where fast convergence to any optimum is prioritized over extensive solution space exploration. Sensitivity analysis reveals robustness to embedding dimensionality when no exact knowledge is available about the true embedding. Sometimes, underestimation of embedding dimensionality is even favourable, and could be interpreted as a search in the most informative linear combination of dimensions.
Ultimately, this chapter brings full circle the methodologies introduced in earlier chapters, and establishes CUATRO as a competitive solver alongside state-of-the-art DFO, BO, and direct search methods in high-dimensional spaces.

6.3.2 Retrospective

Throughout my PhD journey, I have developed expansive expertise that bridges traditional optimization, PSE, and modern ML techniques. My focus however, has been on the role that DFO can play in the application of ML to the optimization of interconnected process systems within the broader context of organizational decision-making frameworks.

My first work consisted of a comprehensive benchmarking of Python-based DFO solvers on PSE applications, with an emphasis on constraint satisfaction and safe exploration. This benchmarking lay the foundation for my subsequent works. In my next project, I explored whether DFO could replace traditional mathematical decomposition techniques in EWO, particularly in scenarios that involve coordinating subproblems with minimal information sharing. While the initial proof-of-concept showed promise, it became clear that pure data- or learning-based methods struggled to

outperform established mathematical techniques, which are designed to exploit problem structure and subgradient information for numerical efficiency.

The workflow in the data-driven coordination project consisted of proposing a novel methodology, vaguely informed by value chain considerations, and trying to find additional use cases for it. Equipped with our new findings, we tried a different approach: We started from an intractable tri-level planning-scheduling-control formulation, integrated optimization, ML, and data science pipelines, and tailored our solution to the problem at hand. Although DFO had already been applied to bi-level planning-scheduling problems, we extended this approach by incorporating control as another level. Our contribution consisted of exploring how we can combine the DFO and surrogate approaches to render the problem tractable, and find the best solution on the most accurate solution metrics in the available on- and offline time budget.

Finally, our last contribution would neatly wrap up the PhD journey. We enhanced the performance of the CUATRO solver on the data-driven coordination and hierarchical integration case studies. Initially showing moderate to good performance across a wide range of problems, CUATRO evolved to compete with the best methods in high-dimensional scenarios. Informed by problem-specific insights, we strategically combined relatively simple techniques — such as quadratic trust region methods and partial least squares regression - and demonstrated that they result in competitive performance in less time compared with more involved methods from the BO and other ML literature.

6.3.3 Future work

Finally, we briefly summarize potential future research directions that we describe in more detail in the corresponding concluding and 'looking back' sections of each chapter.

Direct extensions to this thesis

In Chapter 3, we do not include several powerful DFO solvers and BO implementations, such as HEBO, ALEBO, NOMAD, and CMA-ES. In principle, the benchmarking should be regularly updated to include the best methods to-date. Additionally, we could incorporate more case studies, particularly those involving mixed decision variables or batch evaluations common in chemical engineering. Safe exploration under uncertainty is another promising area that warrants deeper investigation and could even be the focus of a dedicated thesis. Building on the success of CUA-

TRO_PLS, future work could explore the use of quadratic PLS to identify exact optimal linear embeddings, or integrate exploitative subspace optimization techniques with a multi-armed bandit strategy to manage trust regions in parallel. These advances could position CUATRO_PLS as a leading DFO solver for high-dimensional applications.

In Chapter 4, a more comprehensive benchmarking effort could include nonconvex distributed optimization algorithms to provide a fairer comparison with DFO methods, some of which are designed to escape local minima. DFO algorithms can struggle to match the numerical efficiency of ADMM, which relies on subgradient information in their coordination step. A detailed comparison of the privacy aspects of data-driven coordination with those of distributed optimization frameworks may highlight areas where data-driven approaches offer superior safeguards. Additionally, further research is needed to (dis)confirm our intuition that data-driven coordination can be applied in scenarios where agents use heuristic decision-making, or where optimality is not guaranteed in the subproblems.

In the hierarchical integration case study discussed in Chapter 6, it would be interesting to also include hierarchical decomposition techniques in our benchmarking. We could stress-test the relative scaling of the data-driven techniques on different planning, scheduling, and control formulations of increasing size. Ensuring accurate surrogate models is another critical area. Additionally, expanding beyond three levels — such as adding a design level above planning, splitting the control into real-time optimization and setpoint tracking, or combining the data-driven coordination and hierarchical integration settings — could represent the next step in pursuing the ultimate goal of EWO in our motivating case study.

Perspectives on DFO for PSE

DFO is emerging as a powerful tool in PSE and EWO. DFO provides a pragmatic approach when insufficient physical, domain, or model knowledge is available, by abstracting away the complexities of mathematical formulations or organizational considerations. This simplicity makes it attractive to practitioners who may not have the expertise to implement gradient-based optimization. For instance, it may be easier to wrap a simulation model into an input-output black box than to derive analytical gradients. However, the user-friendliness of DFO often comes at the cost of theoretical guarantees and numerical efficiency.

Similar trade-offs pervade throughout PSE: in the face of intractability, do we care more about finding *the* best possible optimization solution at the expense of model accuracy, or do we care

more about finding *a* good solution to an accurate model? When faced with parametric uncertainty in a non-linear mathematical programming formulation, do we relax the model formulations and employ robust optimization techniques, or do we keep the model formulations and relax the way we consider uncertainty? Ultimately, this choice boils down to what is useful to the practitioner. Many coordination problems in EWO remain intractable. While developing tailored solution approaches is essential, finding good, feasible solutions with 'easy-to-use' tools can provide substantial value while we wait for experts to develop specialized tools for the problem at hand or for the organizational landscape to change.

Since our data-driven EWO papers were first published, the validity of using DFO for EWO applications has been bolstered by parallel and follow-up work. Nikkhah et al. [280] solve bi-level planning-scheduling problems with mixed-integer nonlinear scheduling formulations via a hybrid DOMINO-NOMAD DFO framework. Additionally, Krishnamoorthy and Paulson [221] consider an alternative way of using DFO for coordination problems by applying Bayesian optimization of standard acquisition terms in the subproblems, which are augmented with coordinating terms as derived from ADMM. In follow-up work, they generalize this approach to coordinating terms derived from other decomposition techniques and perform regret analysis [220]. I would expect the application of DFO to multi-level, or distributed coordination problems to become more common.

For DFO to gain widespread adoption in PSE, I believe future research efforts should focus on two fronts. First, we can improve general-purpose solvers to demonstrate strong empirical performance out-of-the-box on traditional chemical engineering applications. While we aim to address some of these considerations with CUATRO_PLS in terms of high dimensionality and safe exploration, other works focus on mixed-integer decision spaces and batch evaluations [131, 370]. These improvements can be achieved through a combination of adopting effective techniques from rapidly evolving fields like machine learning and reviving well-known methods from the numerical optimization and operations research literature, as suggested in the future work section for CUATRO_PLS.

Secondly, we should consider which theoretical convergence proofs would be most useful in advancing DFO for PSE. A significant portion of PSE research on DFO has focused, for example, on establishing convergence guarantees to global optima, mirroring past successes in deterministic global optimization. However, I would argue that when guarantees on global optimality are desired, efforts would be better spent on the model-building process. Instead, a key challenge of DFO lies in safety-critical applications. Small steps toward establishing theoretical bounds on constraint violations in the presence of noisy, black-box constraint evaluations, for example, could go a long

way in increasing trust in and adoption of DFO methods.

Just as advances in Bayesian statistical learning have enabled sample-efficient black-box optimization, exploring additional ML-optimization synergies could further enhance DFO's practical applicability.

Perspectives on ML for optimization

Reflecting on the broader landscape, DFO represents just one approach within the realm of ML for optimization. The successful deployment of graph neural networks or the latest recent GenAI models to real-world CO problems could yield substantial online computational savings. Generative flow networks [418, 188] could be applied to navigate conflicting black-box objectives of coordinating value chain agents. Neural algorithmic reasoning [147] could offer new insights into combining mathematical optimization with deep learning in interconnected EWO problems. Finally, there are many other creative opportunities for combining AI tools with mathematical optimization, for example by using LLMs to troubleshoot optimization model infeasibilities [162]. Similarly, we could investigate if we can use LLMs to facilitate coordination between cross-silo value chain agents via workflow orchestration. Given the peculiarities of the chemical industries, we highlight the importance of integrating advanced AI tools thoughtfully and responsibly into existing frameworks to complement human decision-makers rather than replace them.

Bibliography

[1] J. Abramson, J. Adler, J. Dunger, R. Evans, T. Green, A. Pritzel, O. Ronneberger, L. Willmore, A. J. Ballard, J. Bambrick, S. W. Bodenstein, D. A. Evans, C.-C. Hung, M. O'Neill, D. Reiman, K. Tunyasuvunakool, Z. Wu, A. Žemgulytė, E. Arvaniti, C. Beattie, O. Bertolli, A. Bridgland, A. Cherepanov, M. Congreve, A. I. Cowen-Rivers, A. Cowie, M. Figurnov, F. B. Fuchs, H. Gladman, R. Jain, Y. A. Khan, C. M. R. Low, K. Perlin, A. Potapenko, P. Savy, S. Singh, A. Stecula, A. Thillaisundaram, C. Tong, S. Yakneen, E. D. Zhong, M. Zielinski, A. Žídek, V. Bapst, P. Kohli, M. Jaderberg, D. Hassabis, and J. M. Jumper. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630 (8016):493–500, Jun 2024. ISSN 1476-4687. doi: 10.1038/s41586-024-07487-w. URL `https://doi.org/10.1038/s41586-024-07487-w`.

[2] J. Achiam. Spinning Up in Deep Reinforcement Learning, 2018.

[3] T. Achterberg, R. E. Bixby, Z. Gu, E. Rothberg, and D. Weninger. Presolve reductions in mixed integer programming. *INFORMS Journal on Computing*, 32(2):473–506, 2020. doi: 10.1287/ijoc.2018.0857. URL `https://doi.org/10.1287/ijoc.2018.0857`.

[4] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd. A Rewriting System for Convex Optimization Problems. *Journal of Control and Decision*, 5(1):42–60, 9 2017. URL `http://arxiv.org/abs/1709.04494`.

[5] M. Aharon, M. Elad, A. M. Bruckstein, and Y. Katz. K-svd : An algorithm for designing of overcomplete dictionaries for sparse representation. 2005. URL `https://api.semanticscholar.org/CorpusID:1912619`.

[6] R. Ahuja and C. Cunha. Very large-scale neighborhood search for the k-constraint multiple knapsack problem. *Journal of Heuristics*, 11:465–481, 01 2005. doi: 10.1007/s10732-005-2634-9.

[7] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: theory, algorithms, and applications*. Prentice-Hall, Inc., USA, 1993. ISBN 013617549X.

[8] M. H. Akyüz. The capacitated multi-facility weber problem with polyhedral barriers: Efficient heuristic methods. *Computers & Industrial Engineering*, 113:221–240, 2017. ISSN 0360-8352. doi: https://doi.org/10.1016/j.cie.2017.09.011. URL https://www.sciencedirect.com/science/article/pii/S0360835217304217.

[9] A. Allman and Q. Zhang. Distributed fairness-guided optimization for coordinated demand response in multi-stakeholder process networks. *Computers & Chemical Engineering*, 161: 107777, 2022. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2022.107777. URL https://www.sciencedirect.com/science/article/pii/S0098135422001181.

[10] F. Almeida-Trasvina and R. Smith. Design and optimisation of novel cascade refrigeration cycles for LNG production. In *Computer Aided Chemical Engineering*, pages 621–626. Elsevier, 2018. doi: 10.1016/b978-0-444-64235-6.50111-x. URL https://doi.org/10.1016/b978-0-444-64235-6.50111-x.

[11] M. A. Alvarez, L. Rosasco, and N. D. Lawrence. Kernels for vector-valued functions: a review, 2012.

[12] S. Amaran, N. V. Sahinidis, B. Sharda, and S. J. Bury. Simulation optimization: a review of algorithms and applications. *Annals of Operations Research*, 240(1):351–380, 5 2016. ISSN 15729338. doi: 10.1007/s10479-015-2019-x. URL https://link.springer.com/article/10.1007/s10479-015-2019-x.

[13] B. Amos. Tutorial on amortized optimization. *Found. Trends Mach. Learn.*, 16:592–732, 2022. URL https://api.semanticscholar.org/CorpusID:258298859.

[14] O. Andres-Martinez and L. A. Ricardez-Sandoval. Integration of planning, scheduling, and control: A review and new perspectives. *The Canadian Journal of Chemical Engineering*, 100(9):2057–2070, 2022. doi: https://doi.org/10.1002/cjce.24501. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cjce.24501.

[15] M. ApS. *MOSEK Optimizer API for Python 9.2.43*, 2019. URL https://docs.mosek.com/9.2/pythonapi/index.html.

[16] T. Arauz Pison, P. Chanfreut, and J. Maestre. Cyber-security in networked and distributed model predictive control. *Annual Reviews in Control*, 11 2021. doi: 10.1016/j.arcontrol.2021.10.005.

[17] A. C. Arne Drud and Development. *GAMS Documentation 47: Solver manuals: CONOPT*, 2023. URL https://www.gams.com/latest/docs/S_CONOPT4.html.

[18] S. P. Asprey and S. Macchietto. Statistical tools for optimal dynamic model building. In *Computers and Chemical Engineering*, volume 24, pages 1261–1267. Elsevier Science Ltd, 7 2000. doi: 10.1016/S0098-1354(00)00328-8.

[19] M. Athans. The role and use of the stochastic linear-quadratic-gaussian problem in control system design. *IEEE Transactions on Automatic Control*, 16(6):529–552, 1971. doi: 10.1109/TAC.1971.1099818.

[20] T. G. authors. Gpyopt: A bayesian optimization framework in python. `http://github.com/SheffieldML/GPyOpt`, 2016.

[21] S. Avraamidou and E. Pistikopoulos. *Multi-level Mixed-Integer Optimization: Parametric Programming Approach*. De Gruyter, Berlin, Boston, 2022. ISBN 9783110760316. doi: doi:10.1515/9783110760316. URL `https://doi.org/10.1515/9783110760316`.

[22] S. Avraamidou and E. N. Pistikopoulos. A bi-level formulation and solution method for the integration of process design and scheduling. In S. G. Muñoz, C. D. Laird, and M. J. Realff, editors, *Proceedings of the 9th International Conference on Foundations of Computer-Aided Process Design*, volume 47 of *Computer Aided Chemical Engineering*, pages 17–22. Elsevier, 2019. doi: https://doi.org/10.1016/B978-0-12-818597-1.50003-5. URL `https://www.sciencedirect.com/science/article/pii/B9780128185971500035`.

[23] S. Avraamidou and E. N. Pistikopoulos. Adjustable robust optimization through multi-parametric programming. *Optimization Letters*, 14(4):873–887, Jun 2020. ISSN 1862-4480. doi: 10.1007/s11590-019-01438-5. URL `https://doi.org/10.1007/s11590-019-01438-5`.

[24] O. Badejo and M. Ierapetritou. Integrating tactical planning, operational planning and scheduling using data-driven feasibility analysis. *Computers & Chemical Engineering*, 161: 107759, 2022. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2022.107759. URL `https://www.sciencedirect.com/science/article/pii/S0098135422001004`.

[25] M.-F. Balcan, T. Dick, T. Sandholm, and E. Vitercik. Learning to branch. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 344–353. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/balcan18a.html`.

[26] M. Baldea and I. Harjunkoski. Integrated production scheduling and process control: A systematic review. *Computers & Chemical Engineering*, 71:377–390, 2014. ISSN 0098-1354. doi:

https://doi.org/10.1016/j.compchemeng.2014.09.002. URL https://www.sciencedirect.com/science/article/pii/S0098135414002543.

[27] J. R. Beaumont. Connectivity, Complexity, and Catastrophe in Large-Scale Systems. *Journal of the Operational Research Society*, 31(9):863–864, 9 1980. ISSN 0160-5682. doi: 10.1057/jors.1980.158. URL https://www.tandfonline.com/doi/abs/10.1057/jors.1980.158.

[28] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio. Neural combinatorial optimization with reinforcement learning. *ArXiv*, abs/1611.09940, 2016. URL https://api.semanticscholar.org/CorpusID:3649804.

[29] A. Bemporad, N. A. Bozinis, V. Dua, M. Morari, and E. N. Pistikopoulos. Model predictive control: A multi-parametric programming approach. In S. Pierucci, editor, *European Symposium on Computer Aided Process Engineering-10*, volume 8 of *Computer Aided Chemical Engineering*, pages 301–306. Elsevier, 2000. doi: https://doi.org/10.1016/S1570-7946(00)80052-8. URL https://Fwww.sciencedirect.com/science/article/pii/S1570794600800528.

[30] S. Bengesi, H. El-Sayed, M. K. Sarker, Y. Houkpati, J. Irungu, and T. Oladunni. Advancements in generative ai: A comprehensive review of gans, gpt, autoencoders, diffusion model, and transformers. *IEEE Access*, 12:69812–69837, 2024. doi: 10.1109/ACCESS.2024.3397775.

[31] Y. Bengio, A. Lodi, and A. Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2020.07.063. URL https://www.sciencedirect.com/science/article/pii/S0377221720306895.

[32] A. S. Berahas, L. Cao, K. Choromanski, and K. Scheinberg. A theoretical and empirical comparison of gradient approximations in derivative-free optimization. *Found. Comput. Math.*, 22(2):507–560, apr 2022. ISSN 1615-3375. doi: 10.1007/s10208-021-09513-z. URL https://doi.org/10.1007/s10208-021-09513-z.

[33] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13(null):281–305, feb 2012. ISSN 1532-4435.

[34] B. Beykal, F. Boukouvala, C. A. Floudas, N. Sorek, H. Zalavadia, and E. Gildin. Global optimization of grey-box computational systems using surrogate functions and application to highly constrained oil-field operations. *Computers and Chemical Engineering*, 114:99–110, 6 2018. ISSN 00981354. doi: 10.1016/j.compchemeng.2018.01.005.

[35] B. Beykal, S. Avraamidou, I. P. Pistikopoulos, M. Onel, and E. N. Pistikopoulos. DOMINO: Data-driven Optimization of bi-level Mixed-Integer NOnlinear Problems. *Journal of Global Optimization*, 78(1), 9 2020. ISSN 15732916. doi: 10.1007/s10898-020-00890-3.

[36] B. Beykal, S. Avraamidou, and E. Pistikopoulos. *Bi-level Mixed-Integer Data-Driven Optimization of Integrated Planning and Scheduling Problems*, volume 50, pages 1707–1713. 01 2021. ISBN 9780323885065. doi: 10.1016/B978-0-323-88506-5.50265-5.

[37] B. Beykal, S. Avraamidou, and E. N. Pistikopoulos. Data-driven optimization of mixed-integer bi-level multi-follower integrated planning and scheduling problems under demand uncertainty. *Computers & Chemical Engineering*, 156:107551, 2022. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2021.107551. URL `https://www.sciencedirect.com/science/article/pii/S009813542100329X`.

[38] A. Bhosekar and M. Ierapetritou. Advances in surrogate based modeling, feasibility analysis, and optimization: A review. *Computers & Chemical Engineering*, 108:250–267, 2018. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2017.09.017. URL `https://www.sciencedirect.com/science/article/pii/S0098135417303228`.

[39] L. T. Biegler. New nonlinear programming paradigms for the future of process optimization. *AIChE Journal*, 63(4):1178–1193, 4 2017. ISSN 00011541. doi: 10.1002/aic.15674. URL `http://doi.wiley.com/10.1002/aic.15674`.

[40] L. T. Biegler, Y. d. Lang, and W. Lin. Multi-scale optimization for process systems engineering. *Computers and Chemical Engineering*, 60:17–30, 1 2014. ISSN 00981354. doi: 10.1016/j.compchemeng.2013.07.009.

[41] A. Birhane, A. Kasirzadeh, D. Leslie, and S. Wachter. Science in the age of large language models. *Nature Reviews Physics*, 5(5):277–280, May 2023. ISSN 2522-5820. doi: 10.1038/s42254-023-00581-4. URL `https://doi.org/10.1038/s42254-023-00581-4`.

[42] R. G. Bland, D. Goldfarb, and M. J. Todd. Feature article - the ellipsoid method: A survey. *Oper. Res.*, 29:1039–1091, 1981. URL `https://api.semanticscholar.org/CorpusID:12281291`.

[43] P. T. Boggs and J. W. Tolle. Sequential quadratic programming. *Acta Numerica*, 4:1 – 51, 1995. URL `https://api.semanticscholar.org/CorpusID:16697278`.

[44] I. D. L. Bogle. A Perspective on Smart Process Manufacturing Research Challenges for Process Systems Engineers. *Engineering*, 3(2):161–165, 4 2017. ISSN 20958099. doi: 10.1016/J.ENG.2017.02.003.

[45] P. Bonami and J. Lee. Bonmin users' manual. 01 2011.

[46] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization Methods for Large-Scale Machine Learning. *SIAM Review*, 60(2):223–311, 6 2016. URL http://arxiv.org/abs/1606.04838.

[47] F. Boukouvala and C. Floudas. Argonaut: Algorithms for global optimization of constrained grey-box computational problems. *Optimization Letters*, 11, 06 2017. doi: 10.1007/s11590-016-1028-2.

[48] F. Boukouvala and M. G. Ierapetritou. Derivative-free optimization for expensive constrained problems using a novel expected improvement objective function. *AIChE Journal*, 60(7): 2462–2474, 7 2014. ISSN 00011541. doi: 10.1002/aic.14442. URL http://doi.wiley.com/10.1002/aic.14442.

[49] F. Boukouvala, R. Misener, and C. A. Floudas. Global optimization advances in Mixed-Integer Nonlinear Programming, MINLP, and Constrained Derivative-Free Optimization, CDFO, 2016. ISSN 03772217.

[50] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, mar 2004. doi: 10.1017/cbo9780511804441. URL https://www.cambridge.org/core/books/convex-optimization/17D2FAA54F641A2F62C7CCD01DFA97C4.

[51] S. Boyd, L. Xiao, and A. Mutapcic. Notes on Decomposition Methods. Technical report, 2003.

[52] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers, 2010. ISSN 19358237. URL http://www.nowpublishers.com/article/Details/MAL-016.

[53] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson. The industrial internet of things (iiot): An analysis framework. *Computers in Industry*, 101:1–12, 2018. ISSN 0166-3615. doi: https://doi.org/10.1016/j.compind.2018.04.015. URL https://www.sciencedirect.com/science/article/pii/S0166361517307285.

[54] C. C. Bozarth, D. P. Warsing, B. B. Flynn, and E. J. Flynn. The impact of supply chain complexity on manufacturing plant performance. *Journal of Operations*

Management, 27(1):78–93, 1 2009. ISSN 02726963. doi: 10.1016/j.jom.2008.07.003. URL https://onlinelibrary.wiley.com/doi/full/10.1016/j.jom.2008.07.003https://onlinelibrary.wiley.com/doi/abs/10.1016/j.jom.2008.07.003https://onlinelibrary.wiley.com/doi/10.1016/j.jom.2008.07.003.

[55] E. Bradford, L. Imsland, D. Zhang, and E. A. del Rio Chanona. Stochastic data-driven model predictive control using gaussian processes. *Computers & Chemical Engineering*, 139: 106844, aug 2020. ISSN 00981354. doi: 10.1016/j.compchemeng.2020.106844.

[56] S. Bradtke. Reinforcement learning applied to linear quadratic regulation. In S. Hanson, J. Cowan, and C. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5. Morgan-Kaufmann, 1992. URL https://proceedings.neurips.cc/paper_files/paper/1992/file/19bc916108fc6938f52cb96f7e087941-Paper.pdf.

[57] D. G. Brown and J. Boström. Analysis of Past and Present Synthetic Methodologies on Medicinal Chemistry: Where Have All the New Reactions Gone? *Journal of Medicinal Chemistry*, 59(10):4443–4458, 2016. doi: 10.1021/acs.jmedchem.5b01409.

[58] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[59] S. Bubeck, V. Chandrasekaran, R. Eldan, J. A. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y.-F. Li, S. M. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro, and Y. Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4. *ArXiv*, abs/2303.12712, 2023. URL https://api.semanticscholar.org/CorpusID:257663729.

[60] A. Buccini, P. Dell'Acqua, and M. Donatelli. A general framework for admm acceleration. *Numerical Algorithms*, 85, 11 2020. doi: 10.1007/s11075-019-00839-y.

[61] B. Burnak, N. A. Diangelakis, J. Katz, and E. N. Pistikopoulos. Integrated process design, scheduling, and control using multiparametric programming. *Computers & Chemical Engineering*, 125:164–184, 2019. ISSN 0098-1354. doi: https://doi.org/10.1016/

j.compchemeng.2019.03.004. URL https://www.sciencedirect.com/science/article/pii/S0098135418312857.

[62] R.-A. Burtea and C. Tsay. Safe deployment of reinforcement learning using deterministic optimization over neural networks. In A. C. Kokossis, M. C. Georgiadis, and E. Pistikopoulos, editors, *33rd European Symposium on Computer Aided Process Engineering*, volume 52 of *Computer Aided Chemical Engineering*, pages 1643–1648. Elsevier, 2023. doi: https://doi.org/10.1016/B978-0-443-15274-0.50261-4. URL https://www.sciencedirect.com/science/article/pii/B9780443152740502614.

[63] M. L. Bynum, G. A. Hackebeil, W. E. Hart, C. D. Laird, B. L. Nicholson, J. D. Siirola, J.-P. Watson, and D. L. Woodruff. *Pyomo–optimization modeling in python*, volume 67. Springer Science & Business Media, third edition, 2021.

[64] R. Byrd, G. Chin, J. Nocedal, and Y. Wu. Sample size selection in optimization methods for machine learning. *Mathematical Programming*, 134, 08 2012. doi: 10.1007/s10107-012-0572-5.

[65] J. A. Caballero and I. E. Grossmann. An algorithm for the use of surrogate models in modular flowsheet optimization. *AIChE Journal*, 54(10):2633–2650, 10 2008. ISSN 00011541. doi: 10.1002/aic.11579. URL https://aiche.onlinelibrary.wiley.com/doi/full/10.1002/aic.11579https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.11579https://aiche.onlinelibrary.wiley.com/doi/10.1002/aic.11579.

[66] J. M. Calvin and J. Y.-T. Leung. Average-case analysis of a greedy algorithm for the 0/1 knapsack problem. *Operations Research Letters*, 31(3):202–210, 2003. ISSN 0167-6377. doi: https://doi.org/10.1016/S0167-6377(02)00222-5. URL https://www.sciencedirect.com/science/article/pii/S0167637702002225.

[67] M. M. Câmara, A. D. Quelhas, and J. C. Pinto. Performance evaluation of real industrial RTO systems. *Processes*, 4(4):44, 12 2016. ISSN 22279717. doi: 10.3390/pr4040044. URL www.mdpi.com/journal/processes.

[68] Q. Cappart, D. Chételat, E. B. Khalil, A. Lodi, C. Morris, and P. Veličković. Combinatorial optimization and reasoning with graph neural networks. In Z.-H. Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4348–4355. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/595. URL https://doi.org/10.24963/ijcai.2021/595. Survey Track.

[69] R. Carbonneau, K. Laframboise, and R. Vahidov. Application of machine learning techniques for supply chain demand forecasting. *European Journal of Operational Research*, 184(3): 1140–1154, 2008. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2006.12.004. URL https://www.sciencedirect.com/science/article/pii/S0377221706012057.

[70] C. Cartis and L. Roberts. Scalable subspace methods for derivative-free nonlinear least-squares optimization. *Mathematical Programming*, 199, 06 2022. doi: 10.1007/s10107-022-01836-1.

[71] C. Cartis, J. Fiala, B. Marteau, and L. Roberts. Improving the flexibility and robustness of model-based derivative-free optimization solvers. *ACM Trans. Math. Softw.*, 45(3), aug 2019. ISSN 0098-3500. doi: 10.1145/3338517. URL https://doi.org/10.1145/3338517.

[72] C. Cartis, L. Roberts, and O. Sheridan-Methven. Escaping local minima with local derivative-free methods: a numerical investigation. *https://doi.org/10.1080/02331934.2021.1883015*, 2021. ISSN 10294945. doi: 10.1080/02331934.2021.1883015. URL https://www.tandfonline.com/doi/abs/10.1080/02331934.2021.1883015.

[73] C. Cartis, J. Fowkes, and Z. Shao. Randomised subspace methods for non-convex optimization, with applications to nonlinear least-squares, 2022. URL https://arxiv.org/abs/2211.09873.

[74] A. Caspari, C. Tsay, A. Mhamdi, M. Baldea, and A. Mitsos. The integration of scheduling and control: Top-down vs. bottom-up. *Journal of Process Control*, 91:50–62, 2020. ISSN 0959-1524. doi: https://doi.org/10.1016/j.jprocont.2020.05.008. URL https://www.sciencedirect.com/science/article/pii/S0959152420302158.

[75] P. M. Castro, I. E. Grossmann, and Q. Zhang. Expanding scope and computational challenges in process scheduling. *Computers and Chemical Engineering*, 114:14–42, 6 2018. ISSN 00981354. doi: 10.1016/j.compchemeng.2018.01.020.

[76] F. Ceccon, J. Jalving, J. Haddad, A. Thebelt, C. Tsay, C. D. Laird, and R. Misener. Presentation abstract: Optimization formulations for machine learning surrogates. In Y. Yamashita and M. Kano, editors, *14th International Symposium on Process Systems Engineering*, volume 49 of *Computer Aided Chemical Engineering*, pages 57–58. Elsevier, 2022. doi: https://doi.org/10.1016/B978-0-323-85159-6.50008-7. URL https://www.sciencedirect.com/science/article/pii/B9780323851596500087.

[77] F. Ceccon, J. Jalving, J. Haddad, A. Thebelt, C. Tsay, C. D. Laird, and R. Misener. Omlt: Optimization & machine learning toolkit. *J. Mach. Learn. Res.*, 23(1), jan 2022. ISSN 1532-4435.

[78] B. Chachuat, B. Srinivasan, and D. Bonvin. Adaptation strategies for real-time optimization. *Computers and Chemical Engineering*, 33(10):1557–1567, 10 2009. ISSN 00981354. doi: 10.1016/j.compchemeng.2009.04.014.

[79] K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 10:273–304, 1995. URL `https://api.semanticscholar.org/CorpusID:13676847`.

[80] E. A. R. Chanona, P. Petsagkourakis, E. Bradford, J. E. Graciano, and B. Chachuat. Real-time optimization meets Bayesian optimization and derivative-free optimization: A tale of modifier adaptation. *Computers and Chemical Engineering*, 147:107249, 4 2021. ISSN 00981354. doi: 10.1016/j.compchemeng.2021.107249.

[81] C.-L. Chen and W.-C. Lee. Multi-objective optimization of multi-echelon supply chain networks with uncertain product demands and prices. *computers & Chemical Engineering*, 28: 1131–1144, 06 2004. doi: 10.1016/j.compchemeng.2003.09.014.

[82] T. Chen, X. Chen, W. Chen, Z. Wang, H. Heaton, J. Liu, and W. Yin. Learning to optimize: a primer and a benchmark. *J. Mach. Learn. Res.*, 23(1), jan 2022. ISSN 1532-4435.

[83] X. Chen, J. Liu, and W. Yin. Learning to optimize: A tutorial for continuous and mixed-integer optimization, 2024. URL `https://arxiv.org/abs/2405.15251`.

[84] Y. Chu and F. You. Integrated scheduling and dynamic optimization by stackelberg game: Bilevel model formulation and efficient solution algorithm. *Industrial & Engineering Chemistry Research*, 53(13):5564–5581, 2014. doi: 10.1021/ie404272t. URL `https://doi.org/10.1021/ie404272t`.

[85] Y. Chu and F. You. Integrated planning, scheduling, and dynamic optimization for batch processes: Minlp model formulation and efficient solution methods via surrogate modeling. *Industrial & Engineering Chemistry Research*, 53(34):13391–13411, 2014. doi: 10.1021/ie501986d. URL `https://doi.org/10.1021/ie501986d`.

[86] Y. Chu and F. You. Integrated planning, scheduling, and dynamic optimization for sequential batch processes. In *2015 American Control Conference (ACC)*, pages 5635–5640, 2015. doi: 10.1109/ACC.2015.7172222.

[87] Y. Chu and F. You. Model-based integration of control and operations: Overview, challenges, advances, and opportunities. *Computers & Chemical Engineering*, 83:2–20, 2015. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2015.04.011. URL https://www.sciencedirect.com/science/article/pii/S009813541500112X. 24th European Symposium on Computer Aided Process Engineering(ESCAPE).

[88] G. Ciaburro. Machine fault detection methods based on machine learning algorithms: A review. *Mathematical Biosciences and Engineering*, 19(11):11453–11490, 2022. ISSN 1551-0018. doi: 10.3934/mbe.2022534. URL https://www.aimspress.com/article/doi/10.3934/mbe.2022534.

[89] A. D. Clayton, J. A. Manson, C. J. Taylor, T. W. Chamberlain, B. A. Taylor, G. Clemens, and R. A. Bourne. Algorithms for the self-optimisation of chemical reactions, 9 2019. ISSN 20589883. URL https://pubs.rsc.org/en/content/articlehtml/2019/re/c9re00209jhttps://pubs.rsc.org/en/content/articlelanding/2019/re/c9re00209j.

[90] A. D. Clayton, A. M. Schweidtmann, G. Clemens, J. A. Manson, C. J. Taylor, C. G. Niño, T. W. Chamberlain, N. Kapur, A. J. Blacker, A. A. Lapkin, and R. A. Bourne. Automated self-optimisation of multi-step reaction and separation processes using machine learning. *Chemical Engineering Journal*, 384:123340, 3 2020. ISSN 13858947. doi: 10.1016/j.cej.2019.123340.

[91] G. H. Cohen, G. Coon, and N. Nichols. Theoretical consideration of retarded control. *Transactions of the ASME*, 75:827–834, 1952.

[92] B. Colson, P. Marcotte, and G. Savard. Bilevel programming: A survey. *4OR*, 3(2):87–107, Jun 2005. ISSN 1614-2411. doi: 10.1007/s10288-005-0071-0. URL https://doi.org/10.1007/s10288-005-0071-0.

[93] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of Operations Research*, 153:235–256, 2007.

[94] A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust Region Methods*. Society for Industrial and Applied Mathematics, 1 2000. doi: 10.1137/1.9780898719857. URL https://epubs.siam.org/page/terms.

[95] V. Cooner. Solving knapsack problem using constraint programming. 2014. URL https://api.semanticscholar.org/CorpusID:124021469.

[96] L. R. Coralia Cartis and O. Sheridan-Methven. Escaping local minima with local derivative-free methods: a numerical investigation. *Optimization*, 71(8):2343–2373, 2022. doi: 10.1080/02331934.2021.1883015. URL https://doi.org/10.1080/02331934.2021.1883015.

[97] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009. ISBN 0262033844.

[98] G. D. Corporation. General Algebraic Modeling System (GAMS) Release 24.2.1. Washington, DC, USA, 2013. URL http://www.gams.com/.

[99] A. Costa and G. Nannicini. RBFOpt: an open-source library for black-box optimization with costly function evaluations. *Mathematical Programming Computation*, 10(4):597–629, 12 2018. ISSN 18672957. doi: 10.1007/s12532-018-0144-7. URL https://doi.org/10.1007/s12532-018-0144-7.

[100] A. I. Cowen-Rivers, W. Lyu, Z. Wang, R. Tutunov, J. Hao, J. Wang, and H. Bou-Ammar. Hebo: Heteroscedastic evolutionary bayesian optimisation. *ArXiv*, abs/2012.03826, 2020. URL https://api.semanticscholar.org/CorpusID:227338279.

[101] A. Damianou and N. D. Lawrence. Deep Gaussian processes. In C. M. Carvalho and P. Ravikumar, editors, *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pages 207–215, Scottsdale, Arizona, USA, 29 Apr–01 May 2013. PMLR.

[102] P. Daoutidis, J. H. Lee, I. Harjunkoski, S. Skogestad, M. Baldea, and C. Georgakis. Integrating operations and control: A perspective and roadmap for future research. *Computers & Chemical Engineering*, 115:179–184, 2018. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2018.04.011. URL https://www.sciencedirect.com/science/article/pii/S0098135418303132.

[103] P. Daoutidis, J. H. Lee, S. Rangarajan, L. Chiang, B. Gopaluni, A. M. Schweidtmann, I. Harjunkoski, M. Mercangöz, A. Mesbah, F. Boukouvala, F. V. Lima, A. del Rio Chanona, and C. Georgakis. Machine learning in process systems engineering: Challenges and opportunities. *computers & Chemical Engineering*, 181:108523, 2024. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2023.108523. URL https://www.sciencedirect.com/science/article/pii/S0098135423003939.

[104] S. Dara, S. Dhamercherla, S. S. Jadav, C. M. Babu, and M. J. Ahsan. Machine learning in

drug discovery: A review. *Artif. Intell. Rev.*, 55(3):1947–1999, mar 2022. ISSN 0269-2821. doi: 10.1007/s10462-021-10058-4. URL https://doi.org/10.1007/s10462-021-10058-4.

[105] M. L. Darby, M. Nikolaou, J. Jones, and D. Nicholson. RTO: An overview and assessment of current practice, 7 2011. ISSN 09591524.

[106] B. Decardi-Nelson, A. S. Alshehri, A. Ajagekar, and F. You. Generative ai and process systems engineering: The next frontier. *computers & Chemical Engineering*, 187:108723, 2024. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2024.108723. URL https://www.sciencedirect.com/science/article/pii/S0098135424001418.

[107] E. A. del Rio-Chanona, F. Fiorelli, and V. S. Vassiliadis. Automated structure detection for distributed process optimization. *Computers & Chemical Engineering*, 89:135–148, 2016. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2016.03.014. URL https://www.sciencedirect.com/science/article/pii/S0098135416300758.

[108] E. A. Del Rio Chanona, J. E. Alves Graciano, E. Bradford, and B. Chachuat. Modifier-adaptation schemes employing Gaussian processes and trust regions for real-time optimization. In *IFAC-PapersOnLine*, volume 52, pages 52–57. Elsevier B.V., 1 2019. doi: 10.1016/j.ifacol.2019.06.036.

[109] J. Devriendt, A. Gleixner, and J. Nordström. Learn to relax: Integrating 0-1 integer linear programming with pseudo-boolean conflict-driven search. *Constraints*, 26(1–4):26–55, oct 2021. ISSN 1383-7133. doi: 10.1007/s10601-020-09318-x. URL https://doi.org/10.1007/s10601-020-09318-x.

[110] F. Di Fiore, M. Nardelli, and L. Mainini. Active learning and bayesian optimization: A unified perspective to learn with a goal. *Archives of Computational Methods in Engineering*, 04 2024. doi: 10.1007/s11831-024-10064-z.

[111] S. Diamond and S. Boyd. CVXPY: A Python-Embedded Modeling Language for Convex Optimization. *Journal of Machine Learning Research*, 17:1–5, 3 2016. URL http://arxiv.org/abs/1603.00943.

[112] L. S. Dias and M. G. Ierapetritou. Integration of scheduling and control under uncertainties: Review and challenges. *Chemical Engineering Research and Design*, 116:98–113, 2016. ISSN 0263-8762. doi: https://doi.org/10.1016/j.cherd.2016.10.047. URL https://www.sciencedirect.com/science/article/pii/S0263876216304130. Process Systems Engineering - A Celebration in Professor Roger Sargent's 90th Year.

[113] L. S. Dias, R. C. Pattison, C. Tsay, M. Baldea, and M. G. Ierapetritou. A simulation-based optimization framework for integrating scheduling and model predictive control, and its application to air separation units. *Computers & Chemical Engineering*, 113:139–151, 2018. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2018.03.009. URL https://www.sciencedirect.com/science/article/pii/S0098135418301716.

[114] M. Dillon, F. Oliveira, and B. Abbasi. A two-stage stochastic programming model for inventory management in the blood supply chain. *International Journal of Production Economics*, 187:27–41, 2017. ISSN 0925-5273. doi: https://doi.org/10.1016/j.ijpe.2017.02.006. URL https://www.sciencedirect.com/science/article/pii/S0925527317300361.

[115] E. A. Dinic. Algorithm for solution of a problem of maximal flow in a network with power estimation. 1970. URL https://api.semanticscholar.org/CorpusID:115824074.

[116] M. E. Dogan and I. E. Grossmann. A decomposition method for the simultaneous planning and scheduling of single-stage continuous multiproduct plants. *Industrial & Engineering Chemistry Research*, 45:299–315, 2006.

[117] R. Dominguez and S. Cannella. Insights on multi-agent systems applications for supply chain management. *Sustainability (Switzerland)*, 12(5):1–13, 3 2020. ISSN 20711050. doi: 10.3390/su12051935. URL www.mdpi.com/journal/sustainability.

[118] J. Du, J. Park, I. Harjunkoski, and M. Baldea. A time scale-bridging approach for integrating production scheduling and process control. *Computers & Chemical Engineering*, 79:59–69, 2015. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2015.04.026. URL https://www.sciencedirect.com/science/article/pii/S0098135415001271.

[119] H. Duan, P. Vaezipoor, M. B. Paulus, Y. Ruan, and C. J. Maddison. Augment with care: Contrastive learning for combinatorial problems. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 5627–5642. PMLR, 2022. URL https://proceedings.mlr.press/v162/duan22b.html.

[120] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Mach. Learn.*, 110(9):2419–2468, sep 2021. ISSN 0885-6125. doi: 10.1007/s10994-021-05961-4. URL https://doi.org/10.1007/s10994-021-05961-4.

[121] J. P. Eason and L. T. Biegler. Advanced trust region optimization strategies for glass box / black box models. *AIChE Journal*, 64(11):3934–3943, 11 2018. ISSN 00011541. doi: 10.1002/aic.16364. URL http://doi.wiley.com/10.1002/aic.16364.

[122] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, apr 1972. ISSN 0004-5411. doi: 10.1145/321694.321699. URL https://doi.org/10.1145/321694.321699.

[123] L. N. Egidio, A. Hansson, and B. Wahlberg. Learning the step-size policy for the limited-memory broyden-fletcher-goldfarb-shanno algorithm. *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020. URL https://api.semanticscholar.org/CorpusID:222133313.

[124] A. Engelmann, Y. Jiang, B. Houska, and T. Faulwasser. Decomposition of Nonconvex Optimization via Bi-Level Distributed ALADIN. *IEEE Transactions on Control of Network Systems*, 7(4):1848–1858, dec 2020. doi: 10.1109/TCNS.2020.3005079.

[125] M. Erdirik-Dogan, I. E. Grossmann, and J. Wassick. A bi-level decomposition scheme for the integration of planning and scheduling in parallel multi-product batch reactors. In V. Pleşu and P. Şerban Agachi, editors, *17th European Symposium on Computer Aided Process Engineering*, volume 24 of *Computer Aided Chemical Engineering*, pages 625–630. Elsevier, 2007. doi: https://doi.org/10.1016/S1570-7946(07)80127-1. URL https://www.sciencedirect.com/science/article/pii/S1570794607801271.

[126] D. Eriksson and M. Jankowiak. High-dimensional bayesian optimization with sparse axis-aligned subspaces. *ArXiv*, abs/2103.00349, 2021. URL https://api.semanticscholar.org/CorpusID:232076367.

[127] D. Eriksson, M. Pearce, J. R. Gardner, R. Turner, and M. Poloczek. *Scalable global optimization via local Bayesian optimization*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[128] K. C. Felton, J. G. Rittig, and A. A. Lapkin. Summit: Benchmarking Machine Learning Methods for Reaction Optimisation. *Chemistry–Methods*, 1(2):116–122, 2 2021. ISSN 2628-9725. doi: 10.1002/cmtd.202000051. URL https://onlinelibrary.wiley.com/doi/10.1002/cmtd.202000051.

[129] T. D. A. Ferreira, H. A. Shukla, T. Faulwasser, C. N. Jones, and D. Bonvin. Real-Time optimization of Uncertain Process Systems via Modifier Adaptation and Gaussian Processes.

In *2018 European Control Conference, ECC 2018*, pages 465–470. Institute of Electrical and Electronics Engineers Inc., 11 2018. ISBN 9783952426982. doi: 10.23919/ECC.2018.8550397.

[130] FICO. Fico xpress optimizer reference manual, 2024. URL `https://www.fico.com/fico-xpress-optimization/docs/latest/solver/optimizer/HTML/GUID-3BEAAE64-B07F-302C-B880-A11C2C4AF4F6.html`.

[131] J. P. Folch, S. Zhang, R. M. Lee, B. Shafei, D. Walz, C. Tsay, M. van der Wilk, and R. Misener. Snake: Bayesian optimization via pathwise exploration. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9781713871088.

[132] R. Fourer, J. Ma, and K. Martin. Optimization services: A framework for distributed optimization. *Operations Research*, 58(6):1624–1636, 2010. ISSN 0030364X, 15265463. URL `http://www.jstor.org/stable/40984032`.

[133] G. Franceschini and S. Macchietto. Model-based design of experiments for parameter precision: State of the art. *Chemical Engineering Science*, 63(19):4846–4872, 2008.

[134] P. Frazier. A tutorial on bayesian optimization. *ArXiv*, abs/1807.02811, 2018.

[135] P. Frazier, W. Powell, and S. Dayanik. The knowledge-gradient policy for correlated normal beliefs. *INFORMS Journal on Computing*, 21(4):599–613, 2009. doi: 10.1287/ijoc.1080.0314.

[136] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, Cambridge, MA, 1991. Translated into Chinesse by Renin University Press, Bejing: China.

[137] J. M. Gablonsky and C. T. Kelley. A Locally-Biased form of the DIRECT Algorithm. *Journal of Global Optimization*, 21(1):27–37, 2001. ISSN 09255001. doi: 10.1023/A:1017930332101. URL `https://link.springer.com/article/10.1023/A:1017930332101`.

[138] I. Gabriel, A. Manzini, G. Keeling, L. A. Hendricks, V. Rieser, H. Iqbal, N. Tomašev, I. Ktena, Z. Kenton, M. Rodriguez, S. El-Sayed, S. Brown, C. Akbulut, A. Trask, E. Hughes, A. S. Bergman, R. Shelby, N. Marchal, C. Griffin, J. Mateos-Garcia, L. Weidinger, W. Street, B. Lange, A. Ingerman, A. Lentz, R. Enger, A. Barakat, V. Krakovna, J. O. Siy, Z. Kurth-Nelson, A. McCroskery, V. Bolina, H. Law, M. Shanahan, L. Alberts, B. Balle, S. de Haas, Y. Ibitoye, A. Dafoe, B. Goldberg, S. Krier, A. Reese, S. Witherspoon, W. Hawkins, M. Rauh, D. Wallace, M. Franklin, J. A. Goldstein, J. Lehman, M. Klenk, S. Vallor, C. Biles, M. R. Morris, H. King, B. A. y Arcas, W. Isaac, and J. Manyika. The ethics of advanced ai assistants, 2024. URL `https://arxiv.org/abs/2404.16244`.

[139] D. J. Garcia and F. You. The water-energy-food nexus and process systems engineering: A new focus. *computers & Chemical Engineering*, 91:49–67, 2016. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2016.03.003. URL https://www.sciencedirect.com/science/article/pii/S0098135416300552. 12th International Symposium on Process Systems Engineering & 25th European Symposium of Computer Aided Process Engineering (PSE-2015/ESCAPE-25), 31 May - 4 June 2015, Copenhagen, Denmark.

[140] J. García and F. Fernández. A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.*, 16(1):1437–1480, jan 2015. ISSN 1532-4435.

[141] J. Gardner, C. Guo, K. Weinberger, R. Garnett, and R. Grosse. Discovering and Exploiting Additive Structure for Bayesian Optimization. In A. Singh and J. Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1311–1319. PMLR, 20–22 Apr 2017. URL https://proceedings.mlr.press/v54/gardner17a.html.

[142] A. I. Garmendia, J. Ceberio, and A. Mendiburu. Neural combinatorial optimization: a new player in the field. *ArXiv*, abs/2205.01356, 2022. URL https://api.semanticscholar.org/CorpusID:248505773.

[143] R. Garnett. *Bayesian Optimization*. Cambridge University Press, 2023.

[144] R. Garnett, M. A. Osborne, and P. Hennig. Active learning of linear embeddings for gaussian processes. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, page 230–239, Arlington, Virginia, USA, 2014. AUAI Press. ISBN 9780974903910.

[145] A. M. Geoffrion. Elements of Large Scale Mathematical Programming Part II: Synthesis of Algorithms and Bibliography. *Management Science*, 16(11):676–691, 7 1970. ISSN 0025-1909. doi: 10.1287/mnsc.16.11.676. URL https://pubsonline.informs.org/doi/abs/10.1287/mnsc.16.11.676.

[146] A. M. Geoffrion. Elements of Large-Scale Mathematical Programming Part I: Concepts. *Management Science*, 16(11):652–675, 7 1970. ISSN 0025-1909. doi: 10.1287/mnsc.16.11.652. URL https://pubsonline.informs.org/doi/abs/10.1287/mnsc.16.11.652.

[147] D. Georgiev, D. Numeroso, D. Bacciu, and P. Lio'. Neural algorithmic reasoning for combinatorial optimisation. *ArXiv*, abs/2306.06064, 2023. URL https://api.semanticscholar.org/CorpusID:259129830.

[148] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson. Optimal parameter selection for the alternating direction method of multipliers (admm): Quadratic problems. *IEEE Transactions on Automatic Control*, 09 2014. doi: 10.1109/TAC.2014.2354892.

[149] J. González, Z. Dai, P. Hennig, and N. Lawrence. Batch Bayesian Optimization via Local Penalization. Technical report. URL `http://sheffieldml.github.io/GPyOpt/.`

[150] C. E. Gounaris and I. E. Grossmann. A preface to the special issue on enterprise-wide optimization, 12 2019. ISSN 15732924. URL `https://doi.org/10.1007/s11081-019-09468-9`.

[151] A. Grancharova and T. A. Johansen. *Multi-parametric Programming*, pages 1–37. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-28780-0. doi: 10.1007/978-3-642-28780-0_1. URL `https://doi.org/10.1007/978-3-642-28780-0_1`.

[152] I. Grossmann. Enterprise-wide optimization: A new frontier in process systems engineering. *AIChE Journal*, 51(7):1846–1857, 7 2005. ISSN 0001-1541. doi: 10.1002/aic.10617. URL `http://doi.wiley.com/10.1002/aic.10617`.

[153] I. E. Grossmann. Advances in mathematical programming models for enterprise-wide optimization. *Computers & Chemical Engineering*, 47:2–18, 12 2012. ISSN 0098-1354. doi: 10.1016/J.COMPCHEMENG.2012.06.038. URL `https://www.sciencedirect.com/science/article/abs/pii/S0098135412002220`.

[154] I. E. Grossmann and I. Harjunkoski. Process systems Engineering: Academic and industrial perspectives. *Computers and Chemical Engineering*, 126:474–484, 7 2019. ISSN 00981354. doi: 10.1016/j.compchemeng.2019.04.028.

[155] I. E. Grossmann and J. P. Ruiz. Generalized disjunctive programming: A framework for formulation and alternative algorithms for minlp optimization. In J. Lee and S. Leyffer, editors, *Mixed Integer Nonlinear Programming*, pages 93–115, New York, NY, 2012. Springer New York. ISBN 978-1-4614-1927-3.

[156] M. Guignard and S. Kim. Lagrangean decomposition: A model yielding stronger lagrangean bounds. *Mathematical Programming*, 39(2):215–228, 6 1987. ISSN 00255610. doi: 10.1007/BF02592954.

[157] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL `https://www.gurobi.com`.

[158] M. A. Gutiérrez-Limón, A. Flores-Tlacuahuac, and I. E. Grossmann. Minlp formulation for simultaneous planning, scheduling, and control of short-period single-unit processing systems. *Industrial & Engineering Chemistry Research*, 53(38):14679–14694, 2014. doi: 10.1021/ie402563j. URL https://doi.org/10.1021/ie402563j.

[159] M. A. Gutiérrez-Limón, A. Flores-Tlacuahuac, and I. E. Grossmann. A reactive optimization strategy for the simultaneous planning, scheduling and control of short-period continuous reactors. *Computers & Chemical Engineering*, 84:507–515, 2016. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2015.09.017. URL https://www.sciencedirect.com/science/article/pii/S0098135415003129.

[160] B. Hambly, R. Xu, and H. Yang. Policy gradient methods for the noisy linear quadratic regulator over a finite horizon. *SIAM Journal on Control and Optimization*, 59(5):3359–3391, 2021. doi: 10.1137/20M1382386. URL https://doi.org/10.1137/20M1382386.

[161] N. Hansen, Y. Akimoto, and P. Baudis. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, Feb. 2019. URL https://doi.org/10.5281/zenodo.2559634.

[162] G. E. C.-F. Hao Chen and C. Li. Diagnosing infeasible optimization problems using large language models. *INFOR: Information Systems and Operational Research*, 0(0):1–15, 2024. doi: 10.1080/03155986.2024.2385189. URL https://doi.org/10.1080/03155986.2024.2385189.

[163] F. O. Hao-Jun Michael Shi, Melody Qiming Xuan and J. Nocedal. On the numerical performance of finite-difference-based methods for derivative-free optimization. *Optimization Methods and Software*, 38(2):289–311, 2023. doi: 10.1080/10556788.2022.2121832. URL https://doi.org/10.1080/10556788.2022.2121832.

[164] W. E. Hart, J.-P. Watson, and D. L. Woodruff. Pyomo: modeling and solving mathematical programs in python. *Mathematical Programming Computation*, 3(3):219–260, 2011.

[165] F. Hastedt, R. Bailey, K. Hellgardt, S. Yaliraki, E. del Rio-Chanona, and D. Zhang. Investigating the reliability and interpretability of machine learning frameworks for chemical retrosynthesis. *Digital Discovery*, 3, 05 2024. doi: 10.1039/D4DD00007B.

[166] T. Hastie, R. Tibshirani, and J. Friedman. *Overview of Supervised Learning*, pages 9–41. Springer New York, New York, NY, 2009. ISBN 978-0-387-84858-7. doi: 10.1007/978-0-387-84858-7_2. URL https://doi.org/10.1007/978-0-387-84858-7_2.

[167] T. Hastie, R. Tibshirani, and J. Friedman. *Unsupervised Learning*, pages 485–585. Springer New York, New York, NY, 2009. ISBN 978-0-387-84858-7. doi: 10.1007/978-0-387-84858-7_14. URL https://doi.org/10.1007/978-0-387-84858-7_14.

[168] H. He, H. Daume III, and J. M. Eisner. Learning to search in branch and bound algorithms. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper_files/paper/2014/file/757f843a169cc678064d9530d12a1881-Paper.pdf.

[169] J. D. Hedengren, R. A. Shishavan, K. M. Powell, and T. F. Edgar. Nonlinear modeling, estimation and predictive control in APMonitor. *Computers & Chemical Engineering*, 70: 133 – 148, 2014. ISSN 0098-1354. doi: 10.1016/j.compchemeng.2014.04.013.

[170] J. E. Hein. Machine learning made easy for optimizing chemical reactions. *Nature*, 590 (7844):40–41, 2 2021. ISSN 0028-0836. doi: 10.1038/d41586-021-00209-6. URL https://www.nature.com/articles/d41586-021-00209-6.

[171] C. A. Henao and C. T. Maravelias. Surrogate-based superstructure optimization framework. *AIChE Journal*, 2011. ISSN 00011541. doi: 10.1002/aic.12341.

[172] P. Hennig and C. J. Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(1):1809–1837, 2012. doi: 10.5555/2188385.2343701.

[173] M. Herrera, M. Pérez-Hernández, A. Kumar Parlikad, and J. Izquierdo. Multi-Agent Systems and Complex Networks: Review and Applications in Systems Engineering. *Processes*, 8(3): 312, 3 2020. ISSN 2227-9717. doi: 10.3390/pr8030312. URL https://www.mdpi.com/2227-9717/8/3/312.

[174] L. Hewing, J. Kabzan, and M. N. Zeilinger. Cautious Model Predictive Control using Gaussian Process Regression. *IEEE Transactions on Control Systems Technology*, 28(6): 2736–2743, 5 2017. doi: 10.1109/TCST.2019.2949757. URL http://arxiv.org/abs/1705.10702http://dx.doi.org/10.1109/TCST.2019.2949757.

[175] J.-B. Hiriart-Urruty and C. Lemaréchal. Convex analysis and minimization algorithms. 1993. URL https://api.semanticscholar.org/CorpusID:118755909.

[176] W. Hite and P. A. Per. Digitalization and the Chemical Plant of the Future. Technical report.

[177] N. Hollmann, S. Müller, K. Eggensperger, and F. Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *NeurIPS 2022 First Table Representation Workshop*, 2022. URL https://openreview.net/forum?id=eu9fVjVasr4.

[178] T. Homem-De-Mello and G. Bayraksan. Monte Carlo Sampling-Based Methods for Stochastic Optimization. Technical report, 2014.

[179] C. A. Hone, N. Holmes, G. R. Akien, R. A. Bourne, and F. L. Muller. Rapid multistep kinetic model generation from transient flow data. *Reaction Chemistry and Engineering*, 2 (2):103–108, 2017.

[180] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 44(09):5149–5169, sep 2022. ISSN 1939-3539. doi: 10.1109/TPAMI.2021.3079209.

[181] B. Houska, J. Frasch, and M. Diehl. An augmented lagrangian based algorithm for distributed nonconvex optimization. *SIAM Journal on Optimization*, 26(2):1101–1127, 2016. doi: 10.1137/140975991. URL https://doi.org/10.1137/140975991.

[182] J. Huchette, G. Muñoz, T. Serra, and C. Tsay. When deep learning meets polyhedral theory: A survey, 2023.

[183] G. Hüllen, J. Zhai, S. H. Kim, A. Sinha, M. J. Realff, and F. Boukouvala. Managing uncertainty in data-driven simulation-based optimization. *Computers & Chemical Engineering*, 136:106519, may 2020. ISSN 00981354. doi: 10.1016/j.compchemeng.2019.106519.

[184] W. Huyer and A. Neumaier. Snobfit-Stable Noisy Optimization by Branch and Fit. Technical report.

[185] A. Höskuldsson. Quadratic pls regression. *Journal of Chemometrics*, 6(6):307–334, 1992. doi: https://doi.org/10.1002/cem.1180060603. URL https://analyticalsciencejournals.onlinelibrary.wiley.com/doi/abs/10.1002/cem.1180060603.

[186] R. R. Iyer and I. E. Grossmann. A bilevel decomposition algorithm for long-range planning of process networks. *Industrial & Engineering Chemistry Research*, 37:474–481, 1998.

[187] J. R. Jackson and I. E. Grossmann. Temporal decomposition scheme for nonlinear multisite production planning and distribution models. *Industrial & Engineering Chemistry Research*, 42(13):3045–3055, 2003. doi: 10.1021/ie030070p. URL https://doi.org/10.1021/ie030070p.

[188] M. Jain, T. Deleu, J. S. Hartford, C.-H. Liu, A. Hernández-García, and Y. Bengio. Gflownets for ai-driven scientific discovery. *ArXiv*, abs/2302.00615, 2023. URL https://api.semanticscholar.org/CorpusID:256459319.

[189] G. K. Janssens and K. M. Ramaekers. A linear programming formulation for an inventory management decision problem with a service constraint. *Expert Systems with Applications*, 38(7):7929–7934, 2011. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2010.12.009. URL https://www.sciencedirect.com/science/article/pii/S0957417410013722.

[190] N. Ji and X. Gu. Integration of scheduling and control for batch process based on generalized benders decomposition approach with genetic algorithm. *Computers & Chemical Engineering*, 145:107166, 2021. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2020.107166. URL https://www.sciencedirect.com/science/article/pii/S0098135420303306.

[191] H. Jiang, L. Ge, Y. Gao, J. Wang, and R. Song. Large language model for causal decision making. *ArXiv*, abs/2312.17122, 2023. URL https://api.semanticscholar.org/CorpusID:266573774.

[192] T. Jiralerspong, X. Chen, Y. More, V. Shah, and Y. Bengio. Efficient causal graph discovery using large language models. *ArXiv*, abs/2402.01207, 2024. URL https://api.semanticscholar.org/CorpusID:267406688.

[193] D. S. Johnson and L. A. McGeoch. The traveling salesman problem: A case study in local optimization. 2008. URL https://api.semanticscholar.org/CorpusID:208903251.

[194] S. G. Johnson. *The NLopt nonlinear-optimization package*, 2020. URL http://github.com/stevengj/nlopt.

[195] W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into hilbert space. *Contemporary mathematics*, 26:189–206, 1984. URL https://api.semanticscholar.org/CorpusID:117819162.

[196] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, 10 1993. ISSN 00223239. doi: 10.1007/BF00941892. URL https://link.springer.com/article/10.1007/BF00941892.

[197] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998. doi: 10.1023/A:1008306431147.

[198] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konecný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1–2):1–210, 2021. ISSN 1935-8237. doi: 10.1561/2200000083. URL `http://dx.doi.org/10.1561/2200000083`.

[199] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960. ISSN 0021-9223. doi: 10.1115/1.3662552. URL `https://doi.org/10.1115/1.3662552`.

[200] M. I. Kamien and N. L. Schwartz. Sufficient conditions in optimal control theory. *Journal of Economic Theory*, 3(2):207–214, 1971. ISSN 0022-0531. doi: https://doi.org/10.1016/0022-0531(71)90018-4. URL `https://www.sciencedirect.com/science/article/pii/0022053171900184`.

[201] T. Kandanaarachchi and H. N. Perera. Gamified learning of supply chain optimization through the beer distribution game. In *2021 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 503–507, 2021. doi: 10.1109/IEEM50564.2021.9672844.

[202] M. Karimi-Mamaghan, M. Mohammadi, P. Meyer, A. M. Karimi-Mamaghan, and E.-G. Talbi. Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *European Journal of Operational Research*, 296 (2):393–422, 2022. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2021.04.032. URL `https://www.sciencedirect.com/science/article/pii/S0377221721003623`.

[203] K. S. Kaswan, J. S. Dhatterwal, K. Malik, and A. Baliyan. Generative ai: A review on models and applications. In *2023 International Conference on Communication, Security*

and Artificial Intelligence (ICCSAI), pages 699–704, 2023. doi: 10.1109/ICCSAI59793.2023.10421601.

[204] N. Kazantzis and C. Kravaris. Synthesis of state feedback regulators for nonlinear processes. *Chemical Engineering Science*, 55(17):3437–3449, 9 2000. ISSN 00092509. doi: 10.1016/S0009-2509(00)00006-3.

[205] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. 01 2004. ISBN 978-3-540-40286-2. doi: 10.1007/978-3-540-24777-7.

[206] J. E. Kelley, Jr. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712, 1960. doi: 10.1137/0108053. URL https://doi.org/10.1137/0108053.

[207] M. T. Kelley, R. C. Pattison, R. Baldick, and M. Baldea. An milp framework for optimizing demand response operation of air separation units. *Applied Energy*, 222:951–966, 2018. ISSN 0306-2619. doi: https://doi.org/10.1016/j.apenergy.2017.12.127. URL https://www.sciencedirect.com/science/article/pii/S0306261917318524.

[208] M. T. Kelley, C. Tsay, Y. Cao, Y. Wang, J. Flores-Cerrillo, and M. Baldea. A data-driven linear formulation of the optimal demand response scheduling problem for an industrial air separation unit. *Chemical Engineering Science*, 252:117468, 2022. ISSN 0009-2509. doi: https://doi.org/10.1016/j.ces.2022.117468. URL https://www.sciencedirect.com/science/article/pii/S0009250922000525.

[209] M. Khamies, G. Magdy, S. Kamel, and B. Khan. Optimal model predictive and linear quadratic gaussian control for frequency stability of power systems considering wind energy. *IEEE Access*, 9:116453–116474, 2021. doi: 10.1109/ACCESS.2021.3106448.

[210] M. Kim, S. Choi, H. Kim, J. Son, J. Park, and Y. Bengio. Ant colony sampling with gflownets for combinatorial optimization, 2024. URL https://arxiv.org/abs/2403.07041.

[211] S. H. Kim and F. Boukouvala. Machine learning-based surrogate modeling for data-driven optimization: a comparison of subset selection for regression techniques. *Optimization Letters*, 14:989 – 1010, 2019. URL https://api.semanticscholar.org/CorpusID:164710554.

[212] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL https://api.semanticscholar.org/CorpusID:6628106.

[213] T. Kleinert and M. Schmidt. Why there is no need to use a big-m in linear bilevel optimization: A computational study of two ready-to-use approaches. *Computational Management Science*, 20(1), 2023. doi: 10.1007/s10287-023-00435-5.

[214] T. Kleinert, M. Labbé, I. Ljubić, and M. Schmidt. A survey on mixed-integer programming techniques in bilevel optimization. *EURO Journal on Computational Optimization*, 9:100007, 2021. ISSN 2192-4406. doi: https://doi.org/10.1016/j.ejco.2021.100007. URL `https://www.sciencedirect.com/science/article/pii/S2192440621001349`.

[215] P. Koch, O. Golovidov, S. Gardner, B. Wujek, J. Griffin, and Y. Xu. Autotune: A Derivative-free Optimization Framework for Hyperparameter Tuning. *KDD*, 18. doi: 10.1145/3219819.3219837. URL `https://doi.org/10.1145/3219819.3219837`.

[216] B. S. Korkmaz and A. D. R. Chanona. Integrating table representations into large language models for improved scholarly document comprehension. In *SDP*, 2024. URL `https://api.semanticscholar.org/CorpusID:271769638`.

[217] D. Kouzoupis, R. Quirynen, B. Houska, and M. Diehl. A block based ALADIN scheme for highly parallelizable direct Optimal Control. In *Proceedings of the American Control Conference*, 2016. ISBN 9781467386821. doi: 10.1109/ACC.2016.7525066.

[218] M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991. doi: https://doi.org/10.1002/aic.690370209. URL `https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.690370209`.

[219] D. Krishnamoorthy. A distributed feedback-based online process optimization framework for optimal resource sharing. *Journal of Process Control*, 97:72–83, 2021.

[220] D. Krishnamoorthy. A general-purpose approach to multi-agent bayesian optimization across decomposition methods. *Optimization and Engineering*, pages 1–25, 01 2025. doi: 10.1007/s11081-024-09953-w.

[221] D. Krishnamoorthy and J. A. Paulson. Multi-agent black-box optimization using a bayesian approach to alternating direction method of multipliers*. *IFAC-PapersOnLine*, 56(2):2232–2237, 2023. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2023.10.1155. URL `https://www.sciencedirect.com/science/article/pii/S2405896323015586`. 22nd IFAC World Congress.

[222] H. J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106, 1964. doi: 10.1115/1.3653121.

[223] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960. ISSN 00129682, 14680262. URL `http://www.jstor.org/stable/1910129`.

[224] R. Lange, T. Schaul, Y. Chen, C. Lu, T. Zahavy, V. Dalibard, and S. Flennerhag. Discovering attention-based genetic algorithms via meta-black-box optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, page 929–937, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701191. doi: 10.1145/3583131.3590496. URL `https://doi.org/10.1145/3583131.3590496`.

[225] R. Lange, T. Schaul, Y. Chen, T. Zahavy, V. Dalibard, C. Lu, S. Singh, and S. Flennerhag. Discovering evolution strategies via meta-black-box optimization. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, GECCO '23 Companion, page 29–30, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701207. doi: 10.1145/3583133.3595822. URL `https://doi.org/10.1145/3583133.3595822`.

[226] R. T. Lange, Y. Tian, and Y. Tang. Large language models as evolution strategies, 2024. URL `https://arxiv.org/abs/2402.18381`.

[227] C. L. Lara, F. Trespalacios, and I. E. Grossmann. Global optimization algorithm for capacitated multi-facility continuous location-allocation problems. *undefined*, 71(4):871–889, 8 2018. ISSN 15732916. doi: 10.1007/S10898-018-0621-6.

[228] J. Larson, M. Menickelly, and S. M. Wild. Derivative-free optimization methods. *Acta Numerica*, 2019. ISSN 14740508. doi: 10.1017/S0962492919000060.

[229] S. Le Digabel. Algorithm 909: Nomad: Nonlinear optimization with the mads algorithm. *ACM Trans. Math. Softw.*, 37(4), feb 2011. ISSN 0098-3500. doi: 10.1145/1916461.1916468. URL `https://doi.org/10.1145/1916461.1916468`.

[230] J. H. Lee, J. Shin, and M. J. Realff. Machine learning: Overview of the recent progresses and implications for the process systems engineering field. *Computers & Chemical Engineering*, 114:111–121, 6 2018. ISSN 0098-1354. doi: 10.1016/J.COMPCHEMENG.2017.10.008. URL `https://www.sciencedirect.com/science/article/abs/pii/S0098135417303538`.

[231] B. Letham, R. Calandra, A. Rai, and E. Bakshy. Re-examining linear embeddings for high-dimensional bayesian optimization. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

[232] O. Levenspiel. Modeling in chemical engineering. *Chemical Engineering Science*, 57(22):4691–4696, 2002. ISSN 0009-2509. doi: https://doi.org/10.1016/S0009-2509(02)00280-4. URL https://www.sciencedirect.com/science/article/pii/S0009250902002804. Festschrift in Honour of Dr Winn van Swaaij.

[233] Y. Li, K. H. Ang, G. C. Y. Chong, W. Feng, K. C. Tan, and H. Kashiwagi. CAutoCSD-evolutionary search and optimisation enabled computer automated control system design. *International Journal of Automation and Computing*, 1(1):76–88, 10 2004. ISSN 1476-8186. doi: 10.1007/s11633-004-0076-8.

[234] Y. Li, Y. Shen, W. Zhang, Y. Chen, H. Jiang, M. Liu, J. Jiang, J. Gao, W. Wu, Z. Yang, C. Zhang, and B. Cui. Openbox: A generalized black-box optimization service. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 3209–3219, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383325. doi: 10.1145/3447548.3467061. URL https://doi.org/10.1145/3447548.3467061.

[235] Z. Li, Z. Dong, Z. Liang, and Z. Ding. Surrogate-based distributed optimisation for expensive black-box functions. *Automatica*, 125:109407, 2021. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2020.109407. URL https://www.sciencedirect.com/science/article/pii/S0005109820306099.

[236] L. Liberti. Introduction to global optimization. 2006. URL https://api.semanticscholar.org/CorpusID:14115596.

[237] R. Lima and I. Grossmann. Computational advances in solving mixed integer linear programming problems. *Chemical Engineering Greetings to Prof. Sauro Pierucci*, pages 151–160, 01 2011.

[238] G. Lin. A multistart local search heuristic for knapsack problem. In E. Qi, J. Shen, and R. Dou, editors, *The 19th International Conference on Industrial Engineering and Engineering Management*, pages 957–962, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-38391-5.

[239] A. Lindquist. On feedback control of linear stochastic systems. *SIAM Journal on Control*, 11(2):323–343, 1973. doi: 10.1137/0311025. URL https://doi.org/10.1137/0311025.

[240] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989. URL https://api.semanticscholar.org/CorpusID:5681609.

[241] Q. Liu, X. Li, L. Gao, and Y. Li. A modified genetic algorithm with new encoding and decoding methods for integrated process planning and scheduling problem. *IEEE Transactions on Cybernetics*, 51(9):4429–4438, 2021. doi: 10.1109/TCYB.2020.3026651.

[242] X. Liu, Z. Wen, and Y.-X. Yuan. Subspace methods for nonlinear optimization. *CSIAM Transactions on Applied Mathematics*, 2(4):585–651, 2021. ISSN 2708-0579. doi: https://doi.org/10.4208/csiam-am.SO-2021-0016. URL http://global-sci.org/intro/article_detail/csiam-am/19986.html.

[243] Y.-C. Lo, S. E. Rensi, W. Torng, and R. B. Altman. Machine learning in chemoinformatics and drug discovery. *Drug Discovery Today*, 23(8):1538–1546, 2018. ISSN 1359-6446. doi: https://doi.org/10.1016/j.drudis.2018.05.010. URL https://www.sciencedirect.com/science/article/pii/S1359644617304695.

[244] J. Luo, C. Paduraru, O. Voicu, Y. Chervonyi, S. Munns, J. Li, C. Qian, P. Dutta, J. Q. Davis, N. Wu, X. Yang, C.-M. Chang, T. Li, R. Rose, M. Fan, H. Nakhost, T. Liu, B. Kirkman, F. Altamura, L. Cline, P. Tonker, J. Gouker, D. Uden, W. B. Bryan, J. Law, D. Fatiha, N. Satra, J. Rothenberg, M. Waraich, M. Carlin, S. Tallapaka, S. Witherspoon, D. Parish, P. Dolan, C. Zhao, and D. J. Mankowitz. Controlling commercial cooling systems using reinforcement learning, 2022. URL https://arxiv.org/abs/2211.07357.

[245] J. Maestre, D. Pena, and E. Camacho. Distributed model predictive control based on a cooperative game. *Optimal Control Applications and Methods*, 32:153 – 176, 03 2011. doi: 10.1002/oca.940.

[246] M. W. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224, 2011. ISSN 1935-8237. doi: 10.1561/2200000035. URL http://dx.doi.org/10.1561/2200000035.

[247] G. Makrigiorgos, A. D. Bonzanini, V. Miller, and A. Mesbah. Performance-oriented model learning for control via multi-objective bayesian optimization. *Computers & Chemical Engineering*, page 107770, 03 2022. doi: 10.1016/j.compchemeng.2022.107770.

[248] F. Manenti and G. V. Reklaitis. Preface. In F. Manenti and G. V. Reklaitis, editors, *34th European Symposium on Computer Aided Process Engineering / 15th International Symposium on Process Systems Engineering*, volume 53 of *Computer Aided Chemical Engineering*, pages xvii–xviii. Elsevier, 2024. doi: https://doi.org/10.1016/B978-0-443-28824-1.50593-7. URL https://www.sciencedirect.com/science/article/pii/B9780443288241505937.

[249] C. T. Maravelias and I. E. Grossmann. New general continuous-time state−task network formulation for short-term scheduling of multipurpose batch plants. *Industrial & Engineering Chemistry Research*, 42(13):3056–3074, 2003. doi: 10.1021/ie020923y. URL https://doi.org/10.1021/ie020923y.

[250] A. Marousi and V. M. Charitopoulos. Game theoretic optimisation in process and energy systems engineering: A review. *Frontiers in Chemical Engineering*, 5, 2023. ISSN 2673-2718. doi: 10.3389/fceng.2023.1130568. URL https://www.frontiersin.org/articles/10.3389/fceng.2023.1130568.

[251] S. Martello and P. Toth. A mixture of dynamic programming and branch-and-bound for the subset-sum problem. *Management Science*, 30(6):765–771, 1984. doi: 10.1287/mnsc.30.6.765. URL https://doi.org/10.1287/mnsc.30.6.765.

[252] S. Martello and P. Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., USA, 1990. ISBN 0471924202.

[253] S. Martello, D. Pisinger, and P. Toth. Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, 45(3):414–424, 1999. ISSN 00251909, 15265501. URL http://www.jstor.org/stable/2634886.

[254] I. J. Martinez-Moyano. History of the beer game. *System Dynamics Review*, 40(2):e1767, 2024. doi: https://doi.org/10.1002/sdr.1767. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/sdr.1767.

[255] McKinsey. Six steps to successful supply chain collaboration, 7 2012. URL https://www.mckinsey.com/capabilities/operations/our-insights/six-steps-to-successful-supply-chain-collaboration.

[256] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

[257] D. F. Mendoza, J. E. A. Graciano, F. dos Santos Liporace, and G. A. C. Le Roux. Assessing the reliability of different real-time optimization methodologies. *Cana-*

dian Journal of Chemical Engineering, 94(3):485–497, 3 2016. ISSN 1939019X. doi: 10.1002/cjce.22402. URL https://onlinelibrary.wiley.com/doi/full/10.1002/cjce.22402https://onlinelibrary.wiley.com/doi/abs/10.1002/cjce.22402https://onlinelibrary.wiley.com/doi/10.1002/cjce.22402.

[258] A. Merchant, S. Batzner, S. Schoenholz, M. Aykol, G. Cheon, and E. Cubuk. Scaling deep learning for materials discovery. *Nature*, 624:1–6, 11 2023. doi: 10.1038/s41586-023-06735-9.

[259] H.-J. Michael Shi, M. Qiming Xuan, F. Oztoprak, and J. Nocedal. On the Numerical Performance of Derivative-Free Optimization Methods Based on Finite-Difference Approximations. Technical report, 2021.

[260] B. V. Mishra, E. Mayer, J. Raisch, and A. Kienle. Short-term scheduling of batch processes. a comparative study of different approaches. *Industrial & Engineering Chemistry Research*, 44(11):4022–4034, 2005. doi: 10.1021/ie049662d. URL https://doi.org/10.1021/ie049662d.

[261] I. Mitrai and P. Daoutidis. Decomposition of integrated scheduling and dynamic optimization problems using community detection. *Journal of Process Control*, 90:63–74, 2020. ISSN 0959-1524. doi: https://doi.org/10.1016/j.jprocont.2020.04.003. URL https://www.sciencedirect.com/science/article/pii/S0959152420302006.

[262] I. Mitrai and P. Daoutidis. Efficient solution of enterprise-wide optimization problems using nested stochastic blockmodeling. *Industrial & Engineering Chemistry Research*, 60(40):14476–14494, 2021. doi: 10.1021/acs.iecr.1c01570. URL https://doi.org/10.1021/acs.iecr.1c01570.

[263] I. Mitrai and P. Daoutidis. A multicut generalized benders decomposition approach for the integration of process operations and dynamic optimization for continuous systems. *Computers & Chemical Engineering*, 164:107859, 2022. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2022.107859. URL https://www.sciencedirect.com/science/article/pii/S0098135422001971.

[264] I. Mitrai and P. Daoutidis. Taking the human out of decomposition-based optimization via artificial intelligence: Part ii. learning to initialize, 2023.

[265] I. Molan and M. Schmidt. Using neural networks to solve linear bilevel problems with unknown lower level. *Optimization Letters*, 2023. doi: 10.1007/s11590-022-01958-7.

[266] D. Mora-Mariano, M. A. Gutiérrez-Limón, and A. Flores-Tlacuahuac. A lagrangean decomposition optimization approach for long-term planning, scheduling and control. *Computers & Chemical Engineering*, 135:106713, 2020. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2019.106713. URL https://www.sciencedirect.com/science/article/pii/S0098135419307021.

[267] R. Moriconi, K. S. S. Kumar, and M. P. Deisenroth. High-dimensional bayesian optimization with manifold gaussian processes. *ArXiv*, abs/1902.10675, 2019. URL https://api.semanticscholar.org/CorpusID:67856064.

[268] M. R. Morris, J. N. Sohl-Dickstein, N. Fiedel, T. B. Warkentin, A. Dafoe, A. Faust, C. Farabet, and S. Legg. Levels of agi for operationalizing progress on the path to agi. 2023. URL https://api.semanticscholar.org/CorpusID:265033463.

[269] M. Mousa, D. van de Berg, N. Kotecha, E. A. del Rio Chanona, and M. Mowbray. An analysis of multi-agent reinforcement learning for decentralized inventory control systems. *Computers & Chemical Engineering*, 188:108783, 2024. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2024.108783. URL https://www.sciencedirect.com/science/article/pii/S0098135424002011.

[270] M. Mowbray, T. Savage, C. Wu, Z. Song, B. A. Cho, E. A. Del Rio-Chanona, and D. Zhang. Machine learning for biochemical engineering: A review. *Biochemical Engineering Journal*, 172:108054, 2021. ISSN 1369-703X. doi: https://doi.org/10.1016/j.bej.2021.108054. URL https://www.sciencedirect.com/science/article/pii/S1369703X21001303.

[271] M. Mowbray, M. Vallerio, C. Perez-Galvan, D. Zhang, E. del Rio-Chanona, and F. Navarro-Brull. Industrial data science – a review of machine learning applications for chemical and process industries. *Reaction Chemistry & Engineering*, 7, 01 2022. doi: 10.1039/D1RE00541C.

[272] S. Müller, M. Feurer, N. Hollmann, and F. Hutter. Pfns4bo: in-context learning for bayesian optimization. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[273] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018020.

[274] K. G. Murty. *Linear programming*. Wiley, 1983. URL https://cir.nii.ac.jp/crid/1130282272158039040.

[275] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, jan 1965. ISSN 0010-4620. doi: 10.1093/comjnl/7.4.308. URL https://academic.oup.com/comjnl/article-lookup/doi/10.1093/comjnl/7.4.308.

[276] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, 1 1965. ISSN 0010-4620. doi: 10.1093/comjnl/7.4.308. URL https://academic.oup.com/comjnl/article-lookup/doi/10.1093/comjnl/7.4.308.

[277] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012. doi: 10.1137/100802001. URL https://doi.org/10.1137/100802001.

[278] Y. Nesterov and V. Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, Nov. 2015. doi: 10.1007/s10208-015-9296-2. URL https://doi.org/10.1007/s10208-015-9296-2.

[279] M. Neufang, E. Pajak, D. van de Berg, Y. S. Lee, and E. A. del Rio Chanona. Surrogate-based optimization techniques for process systems engineering, 2024. URL https://arxiv.org/abs/2412.13948.

[280] H. Nikkhah, Z. Aghayev, A. Shahbazi, V. M. Charitopoulos, S. Avraamidou, and B. Beykal. Bi-level data-driven enterprise-wide optimization with mixed-integer nonlinear scheduling problems. *Digital Chemical Engineering*, 14:100218, 2025. ISSN 2772-5081. doi: https://doi.org/10.1016/j.dche.2025.100218. URL https://www.sciencedirect.com/science/article/pii/S277250812500002X.

[281] C. Ning and F. You. Optimization under uncertainty in the era of big data and deep learning: When machine learning meets mathematical programming. *Computers & Chemical Engineering*, 125:434–448, 2019. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2019.03.034. URL https://www.sciencedirect.com/science/article/pii/S0098135419300687.

[282] R. Nishihara, L. Lessard, B. Recht, A. Packard, and M. I. Jordan. A general analysis of the convergence of admm. *ArXiv*, abs/1502.02009, 2015.

[283] J. Nocedal and S. J. Wright. Numerical optimization. In *Springer Series in Operations Research and Financial Engineering*. 2006. doi: 10.1201/b19115-11.

[284] F. Oliveira, V. Gupta, S. Hamacher, and I. E. Grossmann. A lagrangean decomposition approach for oil supply chain investment planning under uncertainty with risk considerations. *Comput. Chem. Eng.*, 50:184–195, 2013.

[285] S. Olofsson, E. S. Schultz, A. Mhamdi, A. Mitsos, M. P. Deisenroth, and R. Misener. Design of dynamic experiments for black-box model discrimination. *ArXiv*, abs/2102.03782, 2021. URL `https://api.semanticscholar.org/CorpusID:231846691`.

[286] A. Oroojlooy jadid, M. Nazari, L. Snyder, and M. Takáč. A deep q-network for the beer game: A reinforcement learning algorithm to solve inventory optimization problems. *Neural Information Processing Systems (NIPS), Deep Reinforcement Learning Symposium 2017*, 08 2017.

[287] G. Paaß and S. Giesselbach. *Introduction*, pages 1–17. Springer International Publishing, Cham, 2023. ISBN 978-3-031-23190-2. doi: 10.1007/978-3-031-23190-2_1. URL `https://doi.org/10.1007/978-3-031-23190-2_1`.

[288] D. Palomar and M. Chiang. A tutorial on decomposition methods for network utility maximization. *IEEE Journal on Selected Areas in Communications*, 24(8):1439–1451, 2006. doi: 10.1109/JSAC.2006.879350.

[289] D. P. Palomar and M. Chiang. A tutorial on decomposition methods for network utility maximization. *IEEE Journal on Selected Areas in Communications*, 24(8):1439–1451, aug 2006. ISSN 07338716. doi: 10.1109/JSAC.2006.879350.

[290] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Prentice-Hall, Inc., USA, 1982. ISBN 0131524623.

[291] I. Pappas, D. Kenefake, B. Burnak, S. Avraamidou, H. S. Ganesh, J. Katz, N. A. Diangelakis, and E. N. Pistikopoulos. Multiparametric programming in process systems engineering: Recent developments and path forward. *Frontiers in Chemical Engineering*, 2, 2021. ISSN 2673-2718. doi: 10.3389/fceng.2020.620168. URL `https://www.frontiersin.org/journals/chemical-engineering/articles/10.3389/fceng.2020.620168`.

[292] Y. Park, R. A. Rossi, Z. Wen, G. Wu, and H. Zhao. Structured policy iteration for linear quadratic regulator. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[293] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. De-

Vito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[294] J. Paulson, G. Makrigiorgos, and A. Mesbah. Adversarially robust bayesian optimization for efficient auto-tuning of generic control structures under uncertainty. *AIChE Journal*, 01 2022. doi: 10.1002/aic.17591.

[295] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[296] P. Perdikaris, M. Raissi, A. Damianou, N. D. Lawrence, and G. E. Karniadakis. Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2198), 2017.

[297] H. Perez and I. Grossmann. Extensions to generalized disjunctive programming: hierarchical structures and first-order logic. *Optimization and Engineering*, 25:1–40, 12 2023. doi: 10.1007/s11081-023-09831-x.

[298] P. Petsagkourakis, I. O. Sandoval, E. Bradford, F. Galvanin, D. Zhang, and E. A. del Rio-Chanona. Chance Constrained Policy Optimization for Process Control and Optimization. *arXiv*, 7 2020. URL `http://arxiv.org/abs/2008.00030`.

[299] S. Phuyal, D. Bista, and R. Bista. Challenges, Opportunities and Future Directions of Smart Manufacturing: A State of Art Review. *Sustainable Futures*, 2:100023, 1 2020. ISSN 26661888. doi: 10.1016/j.sftr.2020.100023.

[300] D. Pirouz. An overview of partial least squares. *SSRN Electronic Journal*, 10 2006. doi: 10.2139/ssrn.1631359.

[301] M. Pishvaee, J. Razmi, and S. Torabi. An accelerated benders decomposition algorithm for sustainable supply chain network design under uncertainty: A case study of medical needle and syringe supply chain. *Transportation Research Part E: Logistics and Transportation Review*, 67:14–38, 2014. ISSN 1366-5545. doi: https://doi.org/10.1016/j.tre.2014.04.001. URL `https://www.sciencedirect.com/science/article/pii/S1366554514000520`.

[302] E. N. Pistikopoulos. From multi-parametric programming theory to mpc-on-a-chip multi-scale systems applications. *Computers & Chemical Engineering*, 47:57–66, 2012. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2012.06.031. URL https://www.sciencedirect.com/science/article/pii/S0098135412002153. FOCAPO 2012.

[303] E. N. Pistikopoulos and N. A. Diangelakis. Towards the integration of process design, control and scheduling: Are we getting closer? *Computers & Chemical Engineering*, 91:85–92, 2016. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2015.11.002. URL https://www.sciencedirect.com/science/article/pii/S0098135415003609. 12th International Symposium on Process Systems Engineering & 25th European Symposium of Computer Aided Process Engineering (PSE-2015/ESCAPE-25), 31 May - 4 June 2015, Copenhagen, Denmark.

[304] E. N. Pistikopoulos, N. A. Diangelakis, R. Oberdieck, M. M. Papathanasiou, I. Nascu, and M. Sun. Paroc—an integrated framework and software platform for the optimisation and advanced model-based control of process systems. *Chemical Engineering Science*, 136: 115–138, 2015. ISSN 0009-2509. doi: https://doi.org/10.1016/j.ces.2015.02.030. URL https://www.sciencedirect.com/science/article/pii/S0009250915001451. Control and Optimization of Smart Plant Operations.

[305] N. Poon and J. Martins. Adaptive Constraint Aggregation for Structural Optimization Using Adjoint Sensitivities. *2005 Canadian Aeronautics and Space Institute Annual General Meeting*, pages 1–12, 2005.

[306] M. J. D. Powell. *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation*, pages 51–67. Springer Netherlands, Dordrecht, 1994. ISBN 978-94-015-8330-5. doi: 10.1007/978-94-015-8330-5_4. URL https://doi.org/10.1007/978-94-015-8330-5_4.

[307] M. J. D. Powell. The BOBYQA algorithm for bound constrained optimization without derivatives. Technical report, 2009.

[308] W. Powell. *Introduction to Approximate Dynamic Programming*, chapter 4, pages 111–165. John Wiley & Sons, Ltd, 2011. ISBN 9781118029176. doi: https://doi.org/10.1002/9781118029176.ch4. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118029176.ch4.

[309] A. Prouvost, J. Dumouchelle, M. Gasse, D. Ch'etelat, and A. Lodi. Ecole: A library for learn-

ing inside milp solvers. *ArXiv*, abs/2104.02828, 2021. URL https://api.semanticscholar.org/CorpusID:233168592.

[310] S. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764, 2003. ISSN 0967-0661. doi: https://doi.org/10.1016/S0967-0661(02)00186-7. URL https://www.sciencedirect.com/science/article/pii/S0967066102001867.

[311] A. Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135, 01 2010.

[312] C. E. Rasmussen. *Gaussian Processes in Machine Learning*, pages 63–71. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-28650-9. doi: 10.1007/978-3-540-28650-9_4. URL https://doi.org/10.1007/978-3-540-28650-9_4.

[313] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2016. ISBN 0-262-18253-X.

[314] J. B. Rawlings and C. T. Maravelias. Bringing new technologies and approaches to the operation and control of chemical process systems. *AIChE Journal*, 65(6):e16615, 2019. doi: https://doi.org/10.1002/aic.16615. URL https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.16615.

[315] J. B. Rawlings, D. Q. Mayne, and M. Diehl. *Model predictive control: theory, computation, and design*, volume 2. Nob Hill Publishing Madison, WI, 2017.

[316] J. Reaidy, P. Massotte, and D. Diep. Comparison of negotiation protocols in dynamic agent-based manufacturing systems. *International Journal of Production Economics*, 99 (1):117–130, 2006. ISSN 0925-5273. doi: https://doi.org/10.1016/j.ijpe.2004.12.011. URL https://www.sciencedirect.com/science/article/pii/S0925527304004566. Control and Management of Productive Systems.

[317] L. M. Rios and N. V. Sahinidis. Derivative-free optimization: A review of algorithms and comparison of software implementations. In *Journal of Global Optimization*, volume 56, pages 1247–1293. Springer, 7 2013. doi: 10.1007/s10898-012-9951-y. URL https://link.springer.com/article/10.1007/s10898-012-9951-y.

[318] J. Rodriguez, B. Nicholson, C. Laird, and V. Zavala. Benchmarking admm in nonconvex nlps. *Computers & Chemical Engineering*, 119, 08 2018. doi: 10.1016/j.compchemeng.2018.08.036.

[319] N. Rodríguez-Barroso, G. Stipcich, D. Jiménez-López, J. A. Ruiz-Millán, E. Martínez-Cámara, G. González-Seco, M. V. Luzón, M. A. Veganzones, and F. Herrera. Federated learning and differential privacy: Software tools analysis, the sherpa.ai fl framework and methodological guidelines for preserving data privacy. *Information Fusion*, 64: 270–292, 2020. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2020.07.009. URL `https://www.sciencedirect.com/science/article/pii/S1566253520303213`.

[320] S. Sabet, F. Farokhi, and M. Shokouhifar. A novel artificial bee colony algorithm for the knapsack problem. In *2012 International Symposium on Innovations in Intelligent Systems and Applications*, pages 1–5, 2012. doi: 10.1109/INISTA.2012.6247029.

[321] S. Sachio, M. Mowbray, M. Papathanasiou, E. del Rio-Chanona, and P. Petsagkourakis. Integrating process design and control using reinforcement learning. *Chemical Engineering Research and Design*, 183, 10 2021. doi: 10.1016/j.cherd.2021.10.032.

[322] A. M. Sampat, E. Martin, M. Martin, and V. M. Zavala. Optimization formulations for multi-product supply chain networks. *computers & Chemical Engineering*, 104:296–310, 2017. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2017.04.021. URL `https://www.sciencedirect.com/science/article/pii/S0098135417301849`.

[323] O. Santander, V. Kuppuraj, C. A. Harrison, and M. Baldea. Integrated deep learning-production planning-economic model predictive control framework for large-scale processes. a fluid catalytic cracker-fractionator case study. *Computers & Chemical Engineering*, 167: 107977, 2022. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2022.107977. URL `https://www.sciencedirect.com/science/article/pii/S0098135422003118`.

[324] O. Santander, I. Giannikopoulos, M. A. Stadtherr, and M. Baldea. An integrated stochastic deep learning–short-term production scheduling–optimal control framework for general batch processes. *Industrial & Engineering Chemistry Research*, 62(5):2124–2137, 2023. doi: 10.1021/acs.iecr.2c02638. URL `https://doi.org/10.1021/acs.iecr.2c02638`.

[325] T. Santoso, S. Ahmed, M. Goetschalckx, and A. Shapiro. *A stochastic programming approach for supply chain network design under uncertainty.* 07 2003. doi: 10.18452/8297.

[326] R. W. Sargent. Introduction: 25 years of progress in process systems engineering. *Computers and Chemical Engineering*, 28(4):437–439, 4 2004. ISSN 00981354. doi: 10.1016/j.compchemeng.2003.09.032.

[327] R. W. H. Sargent. Integrated Design and Optimization of Processes. *Chemical Engineering Progress*, 63(9):71–78, 1967.

[328] M. Sarkis, A. Bernardi, N. Shah, and M. M. Papathanasiou. Emerging challenges and opportunities in pharmaceutical manufacturing and distribution. *Processes*, 9(3), 2021. ISSN 2227-9717. doi: 10.3390/pr9030457. URL https://www.mdpi.com/2227-9717/9/3/457.

[329] T. Savage and E. A. del Rio Chanona. Human-algorithm collaborative bayesian optimization for engineering systems, 2024. URL https://arxiv.org/abs/2404.10949.

[330] T. Savage, H. F. Almeida-Trasvina, E. A. del Río-Chanona, R. Smith, and D. Zhang. An adaptive data-driven modelling and optimization framework for complex chemical process design. In *Computer Aided Chemical Engineering*, pages 73–78. Elsevier, 2020. doi: 10.1016/b978-0-12-823377-1.50013-6. URL https://doi.org/10.1016/b978-0-12-823377-1.50013-6.

[331] J. Schoukens and L. Ljung. Nonlinear system identification: A user-oriented road map. *IEEE Control Systems Magazine*, 39(6):28–99, 2019. doi: 10.1109/MCS.2019.2938121.

[332] A. M. Schweidtmann, A. D. Clayton, N. Holmes, E. Bradford, R. A. Bourne, and A. A. Lapkin. Machine learning meets continuous flow chemistry: Automated optimization towards the Pareto front of multiple objectives. *Chemical Engineering Journal*, 352:277–282, 11 2018. ISSN 13858947. doi: 10.1016/j.cej.2018.07.031.

[333] A. M. Schweidtmann, D. Bongartz, W. R. Huster, and A. Mitsos. Deterministic global process optimization: Flash calculations via artificial neural networks. In A. A. Kiss, E. Zondervan, R. Lakerveld, and L. Özkan, editors, *29th European Symposium on Computer Aided Process Engineering*, volume 46 of *Computer Aided Chemical Engineering*, pages 937–942. Elsevier, 2019. doi: https://doi.org/10.1016/B978-0-12-818634-3.50157-0. URL https://www.sciencedirect.com/science/article/pii/B9780128186343501570.

[334] A. M. Schweidtmann, W. R. Huster, J. T. Lüthje, and A. Mitsos. Deterministic global process optimization: Accurate (single-species) properties via artificial neural networks. *Computers and Chemical Engineering*, 121:67–74, 2 2019. ISSN 00981354. doi: 10.1016/j.compchemeng.2018.10.007.

[335] A. M. Schweidtmann, E. Esche, A. Fischer, M. Kloft, J.-U. Repke, S. Sager, and A. Mitsos. Machine learning in chemical engineering: A perspective. *Chemie Ingenieur Tech-*

nik, 93(12):2029–2039, 2021. doi: https://doi.org/10.1002/cite.202100083. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cite.202100083.

[336] B. Settles. *Active Learning*. Morgan & Claypool Publishers, 2012. ISBN 1608457257.

[337] N. Shah. Pharmaceutical supply chains: key issues and strategies for optimisation. *computers & Chemical Engineering*, 28(6):929–941, 2004. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2003.09.022. URL https://www.sciencedirect.com/science/article/pii/S0098135403002333. FOCAPO 2003 Special issue.

[338] N. Shah. Process industry supply chains: Advances and challenges. *computers & Chemical Engineering*, 29(6):1225–1235, 2005. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2005.02.023. URL https://www.sciencedirect.com/science/article/pii/S009813540500013X. Selected Papers Presented at the 14th European Symposium on Computer Aided Process Engineering.

[339] N. Shah, C. Pantelides, and R. Sargent. A general algorithm for short-term scheduling of batch operations—ii. computational issues. *Computers & Chemical Engineering*, 17(2):229–244, 1993. ISSN 0098-1354. doi: https://doi.org/10.1016/0098-1354(93)80016-G. URL https://www.sciencedirect.com/science/article/pii/009813549380016G. An International Journal of Computer Applications in Chemical Enginnering.

[340] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016. doi: 10.1109/JPROC.2015.2494218.

[341] A. Shaiju and I. R. Petersen. Formulas for discrete time lqr, lqg, leqg and minimax lqg optimal control problems. *IFAC Proceedings Volumes*, 41(2):8773–8778, 2008. ISSN 1474-6670. doi: https://doi.org/10.3182/20080706-5-KR-1001.01483. URL https://www.sciencedirect.com/science/article/pii/S1474667016403629. 17th IFAC World Congress.

[342] H.-J. M. Shi, Y. Xie, M. Q. Xuan, and J. Nocedal. Adaptive finite-difference interval estimation for noisy derivative-free optimization. *SIAM Journal on Scientific Computing*, 44(4):A2302–A2321, 2022. doi: 10.1137/21M1452470. URL https://doi.org/10.1137/21M1452470.

[343] B. J. Shields, J. Stevens, J. Li, M. Parasram, F. Damani, J. I. Martinez Alvarado, J. M. Janey, R. P. Adams, and A. G. Doyle. Bayesian reaction optimization as a tool for chemical

synthesis. *Nature*, 590:89, 2021. doi: 10.1038/s41586-021-03213-y. URL https://doi.org/10.1038/s41586-021-03213-y.

[344] J. Shin, T. A. Badgwell, K.-H. Liu, and J. H. Lee. Reinforcement Learning – Overview of recent progress and implications for process control. *Computers & Chemical Engineering*, 127:282–294, 8 2019. ISSN 0098-1354. doi: 10.1016/J.COMPCHEMENG.2019.05.029. URL https://www.sciencedirect.com/science/article/pii/S0098135419300754.

[345] S. Shin, P. Hart, T. Jahns, and V. M. Zavala. A hierarchical optimization architecture for large-scale power networks. *IEEE Transactions on Control of Network Systems*, 6(3): 1004–1014, 2019. doi: 10.1109/TCNS.2019.2906917.

[346] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. Van Den Driessche, T. Graepel, and D. Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 10 2017. ISSN 14764687. doi: 10.1038/nature24270. URL https://www.nature.com/articles/nature24270.

[347] T. Simatupang, B. Hun, and R. Sridharan. Applying the theory of constraints to supply chain collaboration. *Supply Chain Management: An International Journal*, 9, 02 2004. doi: 10.1108/13598540410517584.

[348] J. M. Simkoff and M. Baldea. Production scheduling and linear mpc: Complete integration via complementarity conditions. *Computers & Chemical Engineering*, 125:287–305, 2019. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2019.01.024. URL https://www.sciencedirect.com/science/article/pii/S0098135418310974.

[349] H. A. Simon. The Architecture of Complexity. Technical Report 6, 1962.

[350] A. Sinha, P. Malo, and K. Deb. A review on bilevel optimization: From classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22 (2):276–295, 2018. doi: 10.1109/TEVC.2017.2712906.

[351] D. D. Sleator and R. E. Tarjan. A data structure for dynamic trees. In *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, STOC '81, page 114–122, New York, NY, USA, 1981. Association for Computing Machinery. ISBN 9781450373920. doi: 10.1145/800076.802464. URL https://doi.org/10.1145/800076.802464.

[352] A. Slowik and H. Kwasnicka. Evolutionary algorithms and their applications to engineering problems. *Neural Computing and Applications*, 32(16):12363–12379, Aug 2020.

ISSN 1433-3058. doi: 10.1007/s00521-020-04832-8. URL https://doi.org/10.1007/s00521-020-04832-8.

[353] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Proceedings of 25th International Conference on Neural Information Processing Systems*, volume 2, pages 2951–2959, Red Hook (NY), 2012. Curran Associates Inc.

[354] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, M. Prabhat, and R. Adams. Scalable bayesian optimization using deep neural networks. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2171–2180, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/snoek15.html.

[355] N. Sorek, E. Gildin, F. Boukouvala, B. Beykal, and C. A. Floudas. Dimensionality reduction for production optimization using polynomial approximations. *Computational Geosciences*, 21(2):247–266, 4 2017. ISSN 15731499. doi: 10.1007/s10596-016-9610-3. URL https://link.springer.com/article/10.1007/s10596-016-9610-3.

[356] R. Sousa, S. Liu, L. Papageorgiou, and N. Shah. Global supply chain planning for pharmaceuticals. *Chemical Engineering Research & Design - CHEM ENG RES DES*, 89:2396–2409, 11 2011. doi: 10.1016/j.cherd.2011.04.005.

[357] W. Spendley, G. R. Hext, and F. R. Himsworth. Sequential Application of Simplex Designs in Optimisation and Evolutionary Operation. *Technometrics*, 4(4):441–461, 11 1962. ISSN 15372723. doi: 10.1080/00401706.1962.10490033. URL http://www.tandfonline.com/doi/abs/10.1080/00401706.1962.10490033.

[358] I. standards committee. Enterprise-control system integration. URL https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa95. Online; accessed 21 August 2024.

[359] G. Stephanopoulos. RESEARCH PROGRAM ON ARTIFICIAL INTELLIGENCE IN PROCESS ENGINEERING. *Chemical Engineering Education*, 20(4):182–185, 9 1986. ISSN 00092479.

[360] G. Stephanopoulos. Artificial intelligence in process engineering-current state and future trends. *Computers and Chemical Engineering*, 14(11):1259–1270, 11 1990. ISSN 00981354. doi: 10.1016/0098-1354(90)80006-W.

[361] F. Strozzi, J. Bosch, and J. Zaldívar. Beer game order policy optimization under changing customer demand. *Decision Support Systems*, 42(4):2153–2163, 2007. ISSN 0167-9236. doi: https://doi.org/10.1016/j.dss.2006.06.001. URL https://www.sciencedirect.com/science/article/pii/S0167923606000790. Decision Support Systems in Emerging Economies.

[362] Y. Sui, A. E. Ch, E. Zurich, J. W. Burdick, A. Krause, and K. Ch. Safe Exploration for Optimization with Gaussian Processes. Technical report.

[363] Y. Sun, N. V. Sahinidis, A. Sundaram, and M. S. Cheon. Derivative-free optimization for chemical product design, 3 2020. ISSN 22113398. URL https://doi.org/10.1016/j.coche.2019.11.006.

[364] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

[365] R. Takapoui, N. Moehle, S. Boyd, and A. Bemporad. A simple effective heuristic for embedded mixed-integer quadratic programming. Technical report. URL https://doi.org/.

[366] W. Tang and P. Daoutidis. Distributed control and optimization of process system networks: A review and perspective. *Chinese Journal of Chemical Engineering*, 27(7): 1461–1473, jul 2019. ISSN 1004-9541. doi: 10.1016/J.CJCHE.2018.08.027. URL https://www.sciencedirect.com/science/article/abs/pii/S100495411830853X.

[367] Y. Tang, S. Agrawal, and Y. Faenza. Reinforcement learning for integer programming: Learning to cut. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9367–9376. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/tang20a.html.

[368] M. Tawarmalani and N. V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103:225–249, 2005.

[369] S. Terrazas-Moreno and I. Grossmann. A multiscale decomposition method for the optimal planning and scheduling of multi-site continuous multiproduct plants. *Chemical Engineering Science - CHEM ENG SCI*, 66:4307–4318, 10 2011. doi: 10.1016/j.ces.2011.03.017.

[370] A. Thebelt, J. Kronqvist, M. Mistry, R. M. Lee, N. Sudermann-Merx, and R. Misener. ENTMOOT: A Framework for Optimization over Ensemble Tree Models. *Computers & Chemical*

Engineering, page 107343, may 2021. ISSN 00981354. doi: 10.1016/j.compchemeng.2021.107343. URL https://linkinghub.elsevier.com/retrieve/pii/S0098135421001216.

[371] A. Thebelt, J. Wiebe, J. Kronqvist, C. Tsay, and R. Misener. Maximizing information from chemical engineering data sets: Applications to machine learning. *Chemical Engineering Science*, 252:117469, 2022. ISSN 0009-2509. doi: https://doi.org/10.1016/j.ces.2022.117469. URL https://www.sciencedirect.com/science/article/pii/S0009250922000537.

[372] K. M. Thompson and N. D. Badizadegan. Valuing information in complex systems: An integrated analytical approach to achieve optimal performance in the beer distribution game. *IEEE Access*, 3:2677–2686, 2015. doi: 10.1109/ACCESS.2015.2505730.

[373] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-weka: combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, page 847–855, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450321747. doi: 10.1145/2487575.2487629. URL https://doi.org/10.1145/2487575.2487629.

[374] J. Tönshoff, M. Ritzert, H. Wolf, and M. Grohe. RUN-CSP: unsupervised learning of message passing networks for binary constraint satisfaction problems. *CoRR*, abs/1909.08387, 2019. URL http://arxiv.org/abs/1909.08387.

[375] N. Triantafyllou and M. M. Papathanasiou. Deep learning enhanced mixed integer optimization: Learning to reduce model dimensionality. *Computers & Chemical Engineering*, 187:108725, 2024. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2024.108725. URL https://www.sciencedirect.com/science/article/pii/S0098135424001431.

[376] C. Tsay and M. Baldea. 110th anniversary: Using data to bridge the time and length scales of process systems. *Industrial & Engineering Chemistry Research*, 58(36):16696–16708, 2019. doi: 10.1021/acs.iecr.9b02282. URL https://doi.org/10.1021/acs.iecr.9b02282.

[377] C. Tsay, R. C. Pattison, M. Baldea, B. Weinstein, S. J. Hodson, and R. D. Johnson. A superstructure-based design of experiments framework for simultaneous domain-restricted model identification and parameter estimation. *Computers and Chemical Engineering*, 107:408–426, 12 2017. ISSN 00981354. doi: 10.1016/j.compchemeng.2017.02.014.

[378] C. Tsay, M. Baldea, J. Shi, A. Kumar, and J. Flores-Cerrillo. Data-driven models and algorithms for demand response scheduling of air separation units. In M. R. Eden, M. G.

Ierapetritou, and G. P. Towler, editors, *13th International Symposium on Process Systems Engineering (PSE 2018)*, volume 44 of *Computer Aided Chemical Engineering*, pages 1273–1278. Elsevier, 2018. doi: https://doi.org/10.1016/B978-0-444-64241-7.50207-X. URL https://www.sciencedirect.com/science/article/pii/B978044464241750207X.

[379] C. Tsay, A. Kumar, J. Flores-Cerrillo, and M. Baldea. Optimal demand response scheduling of an industrial air separation unit using data-driven dynamic models. *Computers & Chemical Engineering*, 126:22–34, 2019. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2019.03.022. URL https://www.sciencedirect.com/science/article/pii/S0098135418313371.

[380] C. Tsay, Y. Cao, Y. Wang, J. Flores-Cerrillo, and M. Baldea. Identification and online updating of dynamic models for demand response of an industrial air separation unit. *IFAC-PapersOnLine*, 54(3):140–145, 2021. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2021.08.232. URL https://www.sciencedirect.com/science/article/pii/S240589632101003X. 16th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2021.

[381] C. Tsay, J. Kronqvist, A. Thebelt, and R. Misener. Partition-based formulations for mixed-integer optimization of trained relu neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 3068–3080. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/17f98ddf040204eda0af36a108cbdea4-Paper.pdf.

[382] K. I. Tsianos, S. Lawlor, and M. G. Rabbat. Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning. In *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1543–1550, 2012. doi: 10.1109/Allerton.2012.6483403.

[383] H. Uster, G. Easwaran, E. Akcali, and S. Çetinkaya. Benders decomposition with alternative multiple cuts for a multi-product closed-loop supply chain network design model. *Naval Research Logistics (NRL)*, 54:890 – 907, 12 2007. doi: 10.1002/nav.20262.

[384] D. van de Berg, P. Petsagkourakis, N. Shah, and E. A. del Rio-Chanona. Data-driven coordination of expensive black-boxes. In Y. Yamashita and M. Kano, editors, *14th International Symposium on Process Systems Engineering*, volume 49 of *Computer Aided Chemical Engineering*, pages 1159–1164. Elsevier, 2022. doi: https://doi.org/10.1016/

B978-0-323-85159-6.50193-7. URL https://www.sciencedirect.com/science/article/pii/B9780323851596501937.

[385] D. van de Berg, T. Savage, P. Petsagkourakis, D. Zhang, N. Shah, and E. A. del Rio-Chanona. Data-driven optimization for process systems engineering applications. *Chemical Engineering Science*, 248:117135, 2 2022. ISSN 0009-2509. doi: 10.1016/J.CES.2021.117135.

[386] D. van de Berg, R. X. J. Jimbo, N. Shah, and E. A. del Rio-Chanona. Tractable data-driven solutions to hierarchical planning-scheduling-control. In A. C. Kokossis, M. C. Georgiadis, and E. Pistikopoulos, editors, *33rd European Symposium on Computer Aided Process Engineering*, volume 52 of *Computer Aided Chemical Engineering*, pages 649–654. Elsevier, 2023. doi: https://doi.org/10.1016/B978-0-443-15274-0.50103-7. URL https://www.sciencedirect.com/science/article/pii/B9780443152740501037.

[387] D. van de Berg, P. Petsagkourakis, N. Shah, and E. A. del Rio-Chanona. Data-driven coordination of subproblems in enterprise-wide optimization under organizational considerations. *AIChE Journal*, 69(4):e17977, 2023. doi: https://doi.org/10.1002/aic.17977. URL https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.17977.

[388] D. van de Berg, N. Shah, and A. del Rio-Chanona. High-dimensional derivative-free optimization via trust region surrogates in linear subspaces. In F. Manenti and G. V. Reklaitis, editors, *34th European Symposium on Computer Aided Process Engineering / 15th International Symposium on Process Systems Engineering*, volume 53 of *Computer Aided Chemical Engineering*, pages 3193–3198. Elsevier, 2024. doi: https://doi.org/10.1016/B978-0-443-28824-1.50533-0. URL https://www.sciencedirect.com/science/article/pii/B9780443288241505330.

[389] D. van de Berg, N. Shah, and E. A. del Rio-Chanona. Hierarchical planning-scheduling-control — optimality surrogates and derivative-free optimization. *Computers and Chemical Engineering*, 188:108726, 2024. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2024.108726. URL https://www.sciencedirect.com/science/article/pii/S0098135424001443.

[390] V. V. Vazirani. *Knapsack*, pages 68–73. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. ISBN 978-3-662-04565-7. doi: 10.1007/978-3-662-04565-7_8. URL https://doi.org/10.1007/978-3-662-04565-7_8.

[391] P. Veličković and C. Blundell. Neural algorithmic reasoning. *Patterns*, 2(7):100273, 2021. ISSN 2666-3899. doi: https://doi.org/10.1016/j.patter.2021.100273. URL https://www.sciencedirect.com/science/article/pii/S2666389921000994.

[392] V. Venkatasubramanian. The promise of artificial intelligence in chemical engineering: Is it here, finally? *AIChE Journal*, 65(2):466–478, 2 2019. ISSN 15475905. doi: 10.1002/aic.16489. URL https://aiche.onlinelibrary.wiley.com/doi/full/10.1002/aic.16489https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.16489https://aiche.onlinelibrary.wiley.com/doi/10.1002/aic.16489.

[393] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[394] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, may 2006. ISSN 00255610. doi: 10.1007/s10107-004-0559-y. URL https://link.springer.com/article/10.1007/s10107-004-0559-y.

[395] X. Wan, J. F. Pekny, and G. V. Reklaitis. Simulation based optimization of supply chains with a surrogate model. In A. Barbosa-Póvoa and H. Matos, editors, *European Symposium on Computer-Aided Process Engineering-14*, volume 18 of *Computer Aided Chemical Engineering*, pages 1009–1014. Elsevier, 2004. doi: https://doi.org/10.1016/S1570-7946(04)80234-7. URL https://www.sciencedirect.com/science/article/pii/S1570794604802347.

[396] G. G. Wang and S. Shan. Review of Metamodeling Techniques in Support of Engineering Design Optimization. *Journal of Mechanical Design*, 129(4):370–380, 05 2006. ISSN 1050-0472. doi: 10.1115/1.2429697. URL https://doi.org/10.1115/1.2429697.

[397] S. Wang, J. Sun, and Z. Xu. Hyperadam: A learnable task-adaptive adam for network training. In *AAAI Conference on Artificial Intelligence*, 2018. URL https://api.semanticscholar.org/CorpusID:53715634.

[398] Z. Wang and M. Ierapetritou. A Novel Surrogate-Based Optimization Method for Black-Box Simulation with Heteroscedastic Noise. *Industrial and Engineering Chemistry Research*, 56(38):10720–10732, 9 2017. ISSN 15205045. doi: 10.1021/acs.iecr.7b00867. URL https://pubs.acs.org/doi/abs/10.1021/acs.iecr.7b00867.

[399] Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. De Freitas. Bayesian optimization in a billion dimensions via random embeddings. *J. Artif. Int. Res.*, 55(1):361–387, jan 2016. ISSN 1076-9757.

[400] Z. Wang, J. Zhang, T.-H. Chang, J. Li, and Z.-Q. Luo. Distributed stochastic consensus optimization with momentum for nonconvex nonsmooth problems. *IEEE Transactions on Signal Processing*, 69:4486–4501, 2021. doi: 10.1109/tsp.2021.3097211. URL `https://doi.org/10.1109%2Ftsp.2021.3097211`.

[401] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep kernel learning. In A. Gretton and C. C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 370–378, Cadiz, Spain, 09–11 May 2016. PMLR. URL `https://proceedings.mlr.press/v51/wilson16.html`.

[402] Z. T. Wilson and N. V. Sahinidis. The ALAMO approach to machine learning. *Computers and Chemical Engineering*, 106:785–795, 11 2017. ISSN 00981354. doi: 10.1016/j.compchemeng.2017.02.010.

[403] W. Winston and J. Goldberg. *Operations research: applications and algorithms*. 01 2004. ISBN 9780534423629.

[404] S. Wolfram. *A New Kind of Science*. Wolfram Media, 2002.

[405] S. Wolfram. Can ai solve sceince? `https://writings.stephenwolfram.com/2024/03/can-ai-solve-science/`, 2024. Accessed: 17th July 2024.

[406] D. P. Woodruff. Sketching as a tool for numerical linear algebra. *Found. Trends Theor. Comput. Sci.*, 10(1–2):1–157, oct 2014. ISSN 1551-305X. doi: 10.1561/0400000060. URL `https://doi.org/10.1561/0400000060`.

[407] D. P. Xenos, I. Mohd Noor, M. Matloubi, M. Cicciotti, T. Haugen, and N. F. Thornhill. Demand-side management and optimal operation of industrial electricity consumers: An example of an energy-intensive chemical plant. *Applied Energy*, 182:418–433, 2016. ISSN 0306-2619. doi: https://doi.org/10.1016/j.apenergy.2016.08.084. URL `https://www.sciencedirect.com/science/article/pii/S0306261916311710`.

[408] Y. Xu, M. Liu, Q. Lin, and T. Yang. Admm without a fixed penalty parameter: Faster convergence with new adaptive penalization. In *Proceedings of the 31st International Conference*

on Neural Information Processing Systems, NIPS'17, page 1267–1277, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[409] Y. Xu, S. Zhu, S. Yang, C. Zhang, R. Jin, and T. Yang. Learning with Non-Convex Truncated Losses by SGD. *35th Conference on Uncertainty in Artificial Intelligence, UAI 2019*, 5 2018. URL https://arxiv.org/abs/1805.07880v1.

[410] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu. 2019.

[411] T. Yang, X. Yi, J. Wu, Y. Yuan, D. Wu, Z. Meng, Y. Hong, H. Wang, Z. Lin, and K. H. Johansson. A survey of distributed optimization. *Annual Reviews in Control*, 47:278–305, 2019. ISSN 1367-5788. doi: https://doi.org/10.1016/j.arcontrol.2019.05.006. URL https://www.sciencedirect.com/science/article/pii/S1367578819300082.

[412] A. C. Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 160–164, 1982. doi: 10.1109/SFCS.1982.38.

[413] H. Yaşarcan. (2014) a mathematical model of the beer game. 2015. URL https://api.semanticscholar.org/CorpusID:195258010.

[414] D. Yue and F. You. Stackelberg-game-based modeling and optimization for supply chain design and operations: A mixed integer bilevel programming framework. *Computers & Chemical Engineering*, 102:81–95, 2017. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2016.07.026. URL https://www.sciencedirect.com/science/article/pii/S0098135416302460. Sustainability & Energy Systems.

[415] V. Zavala. Chapter seven - managing conflicts among decision-makers in multiobjective design and operations. In G. Ruiz-Mercado and H. Cabezas, editors, *Sustainability in the Design, Synthesis and Analysis of Chemical Engineering Processes*, pages 169–180. Butterworth-Heinemann, Oxford, 2016. ISBN 978-0-12-802032-6. doi: https://doi.org/10.1016/B978-0-12-802032-6.00007-4. URL https://www.sciencedirect.com/science/article/pii/B9780128020326000074.

[416] V. M. Zavala and A. D. Smith. The shape of complex systems. *Nature Chemical Engineering*, 1(7):494–494, Jul 2024. ISSN 2948-1198. doi: 10.1038/s44286-024-00101-w. URL https://doi.org/10.1038/s44286-024-00101-w.

[417] M. Zečević, M. Willig, D. S. Dhami, and K. Kersting. Causal parrots: Large language models may talk causality but are not causal. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=tv46tCzs83.

[418] D. Zhang, H. Dai, N. Malkin, A. Courville, Y. Bengio, and L. Pan. Let the flows tell: solving graph combinatorial optimization problems with gflownets. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2024. Curran Associates Inc.

[419] G. Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003. ISSN 0925-2312. doi: https://doi.org/10.1016/S0925-2312(01)00702-0. URL https://www.sciencedirect.com/science/article/pii/S0925231201007020.

[420] Q. Zhang, K. Ding, T. Lyv, X. Wang, Q. Yin, Y. Zhang, J. Yu, Y. Wang, X. Li, Z. Xiang, X. Zhuang, Z. Wang, M. Qin, M. Zhang, J. Zhang, J. Cui, R. Xu, H. Chen, X. Fan, H. Xing, and H. Chen. Scientific large language models: A survey on biological & chemical domains, 2024. URL https://arxiv.org/abs/2401.14656.

[421] Y. Zhang, Y. Zhang, Y. Gan, L. Yao, and C. Wang. Causal graph discovery with retrieval-augmented generation based large language models, 2024. URL https://arxiv.org/abs/2402.15301.

[422] F. Zhao, I. E. Grossmann, S. García-Muñoz, and S. D. Stamatis. Flexibility index of black-box models with parameter uncertainty through derivative-free optimization. *AIChE Journal*, 2021. ISSN 15475905. doi: 10.1002/aic.17189.

[423] J. Zhao, T. Li, and J. Qian. Application of particle swarm optimization algorithm on robust PID controller tuning. In *Lecture Notes in Computer Science*, volume 3612, pages 948–957. Springer Verlag, 2005. doi: 10.1007/11539902{_}118. URL https://link.springer.com/chapter/10.1007/11539902_118.

[424] M. Zhu, A. Mroz, L. Gui, K. Jelfs, A. Bemporad, E. del Rio-Chanona, and Y. Lee. Discrete and mixed-variable experimental design with surrogate-based approach, 04 2024.

[425] X. Zhu, X. Guo, W. Wang, and J. Wu. A genetic programming-based iterative approach for the integrated process planning and scheduling problem. *IEEE Transactions on Automation Science and Engineering*, 19(3):2566–2580, 2022. doi: 10.1109/TASE.2021.3091610.

[426] J. G. Ziegler and N. Nichols. Optimum settings for automatic controllers. *Journal of Dynamic Systems Measurement and Control-transactions of The Asme*, 115:220–222, 1942.

[427] K. J. Åström and T. Hägglund. Automatic tuning of simple regulators. Technical report, Department of Automatic Control, Lund Institute of Technology (LTH), 1984. URL https://lup.lub.lu.se/search/ws/files/47932505/7269.pdf.

A

Supplementary information for Chapter 3

A.1 Finite difference approximation algorithms

A.1.1 Finite Difference Approximation

In the absence of gradient information, a natural idea would be to use finite difference approximation methods to approximate first- or second-order derivatives as necessary to be used within conventional first- or second-order optimization frameworks.

The forward difference approximation of the partial derivative of f with respect to the ith dimension of $\mathbf{x}$ is defined as follows:

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = \frac{f(x_1, ..., x_i + h, ..., x_{n_x}) - f(x_1, ..., x_i, ..., x_{n_x})}{h} \tag{A.1}$$

This follows from the definition of the partial derivative of a function f with respect to x_i in the case where h approaches 0. In the numerical implementation, the smaller the h, the more accurate the approximation, unless h is so small that it leads to numerical difficulties with floating point operations. Notice that using this forward difference method, each gradient approximation requires $n_x + 1$ operations.

When additionally more precision is required, or second-order derivative information needs to be approximated, one can use the central difference method:

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = \frac{f(x_1, ..., x_i + h, ..., x_{n_x}) - f(x_1, ..., x_i - h, ..., x_{n_x})}{2h} \tag{A.2}$$

The solution order accuracy also improves to $\mathcal{O}(h^2)$ compared to the $\mathcal{O}(h)$ of the forward difference method, while requiring only twice the number of function evaluations. These function evaluations can then be reused in the approximation of the second derivative of f with respect to x_i and x_j:

$$\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_{i'}} = \frac{f(\mathbf{x} + h_i + h_{i'}) - f(\mathbf{x} + h_i) - f(\mathbf{x} + h_{i'}) + 2f(\mathbf{x}) + f(\mathbf{x} - h_i - h_{i'}) - f(\mathbf{x} - h_i) - f(\mathbf{x} - h_{i'})}{2h_i h_{i'}} \tag{A.3}$$

where h_i denotes the n-dimensional vectors consisting of zeroes apart from the ith dimension, which consists of the step size h. In this case, on top of the $2n_x + 1$ evaluations required in (A.2), only an additional $n_x^2 - n_x$ terms are required, making the joint approximation of the gradient and Hessian $\mathcal{O}(n_x^2)$.

A.1.2 Newton's method via finite differences

Newton's method is a second-order method that requires the use of second-order derivative information. The second order information helps in capturing the curvature of the problem when computing the search direction. This should lead to convergence in fewer iterations than first-order methods. Newton steps can be interpreted in various ways [50], as the minimizer of the second-order Taylor approximation, the steepest descent step using a Hessian norm or as the solution of the linearized optimality condition. In fact, after constructing the second order Taylor expansion of f around a current iterate $\mathbf{x}^k$, we get:

$$f(\mathbf{x}^k + \mathbf{s}) \approx f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^\intercal \mathbf{s} + \frac{1}{2}\mathbf{s}^\intercal \nabla^2 f(\mathbf{x}^k)\mathbf{s}, \tag{A.4}$$

where $s \in \mathbb{R}^{n_x}$ is the search direction $\mathbf{x} - \mathbf{x}^k$. The minimizer of (A.4) then yields the search direction at iteration k:

$$\mathbf{s}^k = -\nabla^2 f(\mathbf{x}^k)^{-1} \nabla f(\mathbf{x}^k) \tag{A.5}$$

In practice, backtracking line search is used to find the step size that gives a satisfying improvement in the search direction rather than using a fixed step size $\alpha \in \mathbb{R}$. The only difference to the

traditional Newton method is that both gradient and Hessian information are approximated, and that the backtracking also makes sure that no constraints are violated after taking the step.

Algorithm 11: Approximate Newton

Input: Black-box objective f and constraints $g_l, l = 1..n_g$, Initial guess $\mathbf{x}^0$, Step size α, Maximum number of function evaluations $N_{f,max}$

Initialisation: Function evaluation counter $n_f := 1$, Iteration counter $k := 0$
while *($N_{f,max} \geq n_f$)* **do**
  Compute gradient $\nabla f(\mathbf{x}_j)$ and Hessian $H(\mathbf{x}^k)$ from (A.2) and (A.3) ;
  $\mathbf{s} \leftarrow -H^{-1}\nabla f(\mathbf{x}^k)$;
  $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k + \alpha\mathbf{s}$; (*)
  Record all n_s function evaluations ; $n_f \leftarrow n_f + n_s$; $k \leftarrow k + 1$;
end
(*) means that backtracking can be used to find α instead

A.1.3 BFGS via finite differences

BFGS, named after its discoverers Broyden, Fletcher, Goldfarb, and Shanno, is the most popular quasi-Newton algorithm [283]. It seeks the same search direction as in (A.5), but avoids the repeated computation and inversion of the Hessian. Instead, it updates the inverse of the Hessian at iteration $k + 1$:

$$B^{k+1} = (I - \rho^k \mathbf{s}^k \mathbf{v}^{k\intercal})B^k(I - \rho^k \mathbf{v}^k \mathbf{s}^{k\intercal}) + \rho^k \mathbf{s}^k \mathbf{s}^{k\intercal} \tag{A.6}$$

where $B^k \approx H^{k^{-1}} \in \mathbb{R}^{n_x \times n_x}$, $\rho^k = \frac{1}{\mathbf{v}^{k\intercal}\mathbf{s}^k} \in \mathbb{R}$, $\mathbf{s}^k = \mathbf{x}^{k+1} - \mathbf{x}^k$ and $\mathbf{v}^k = \nabla\mathbf{f}(\mathbf{x}^k) - \nabla\mathbf{f}(\mathbf{x}^{k-1}) \in \mathbb{R}^{n_x}$, α^k being the step length and $\mathbf{s}^k$ the search direction at iteration k. Again, in practice, backtracking is used to find α^k and the constraint handling is identical to that in the previous method. In the finite difference version of BFGS, the gradients are approximated at each step, and the Hessian only in the first step.

Algorithm 12: Approximate BFGS

Input: Black-box objective f and constraints $g_l, l = 1..n_g$, Initial guess $\mathbf{x}^0$, Step size α, Maximum number of function evaluations $N_{f,max}$

Initialisation: Function evaluation counter $n_f := 1$, Iteration counter $k := 0$
Compute gradient $\nabla\mathbf{f} \leftarrow \nabla\mathbf{f}(\mathbf{x}^k)$ and Hessian $H \leftarrow H(\mathbf{x}^k)$ from (A.2) and (A.3) ;
Search space: $\mathbf{s} \leftarrow -H^{-1}\nabla\mathbf{f}$;
while *($N_{f,max} \geq n_f$)* **do**
  Update old inverse Hessian $B_{old} \leftarrow H^{-1}$ and old gradient $\nabla\mathbf{f}_{old} \leftarrow \nabla\mathbf{f}$;
  $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k + \alpha\mathbf{s}$; (*)
  Record all n_s function evaluations ; $n_f \leftarrow n_f + n_s$; $k \leftarrow k + 1$;
  Compute gradient $\nabla\mathbf{f} \leftarrow \nabla\mathbf{f}(\mathbf{x}^k)$ from (A.1) ;
  $\mathbf{v} \leftarrow \nabla\mathbf{f} - \nabla\mathbf{f}_{old}$; $\rho \leftarrow \frac{1}{\mathbf{v}^\intercal\mathbf{s}}$; Compute B from (A.6) ; $\mathbf{s} \leftarrow -B\nabla\mathbf{f}$;
end
(*) backtracking can be used to find α instead

A.1.4 Adam via finite differences

Adam is an efficient stochastic optimization algorithm requiring only first-order gradient information. The algorithm name Adam is derived from adaptive moment estimation. As the name suggests, Adam uses bias-corrected estimates of the first- and second-order momentum, making the algorithm efficient for non-stationary objectives. It also naturally performs a type of step size annealing. This implementation, apart from approximating first-order gradient information, deviates from the original implementation [212] in that it also includes a backtracking step to prevent the updates from becoming infeasible.

Algorithm 13: Adam

Input: Black-box objective f and constraints $g_l, l = 1..n_g$, Initial guess $\mathbf{x}^0$, Step size α, β_1, β_2, δ, Maximum number of function evaluations $N_{f,max}$

Initialisation: Function evaluation counter $n_f := 1$, Iteration counter $k := 0$

Compute gradient $\nabla\mathbf{f} \leftarrow \nabla\mathbf{f}(\mathbf{x}^k)$ from (A.1) ; $\mathbf{s} \leftarrow -\nabla\mathbf{f}$; $\mathbf{m} \leftarrow \nabla\mathbf{f}$; $\mathbf{v} \leftarrow (\nabla\mathbf{f})^2$;

while *($N_{f,max} \geq n_f$)* **do**

  $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k + \alpha\mathbf{s}$; (*)

  Record all n_s function evaluations ; $n_f \leftarrow n_f + n_s$; $k \leftarrow k + 1$;

  Compute gradient $\nabla\mathbf{f} \leftarrow \nabla\mathbf{f}(\mathbf{x}^k)$ from (A.1) ;

  First momentum: $\mathbf{m} = \beta_1\mathbf{m} + (1-\beta_1)\nabla\mathbf{f}$; Corrected term: $\widehat{\mathbf{m}} = \frac{\mathbf{m}}{1-\beta_1^j}$;

  Second momentum: $\mathbf{v} = \beta_2\mathbf{v} + (1-\beta_2)(\nabla\mathbf{f})^2$; Corrected term: $\widehat{\mathbf{v}} = \frac{\mathbf{v}}{1-\beta_2^j}$;

  $\mathbf{s} = -\frac{\widehat{\mathbf{m}}}{\sqrt{\widehat{\mathbf{v}}}+\delta}$;

end

(*) means that backtracking can be used to find α instead

A.2 Gaussian Processes

Gaussian process regression is a method of interpolation popularized by the machine learning community [313]. It aims to describe an unknown function $f : \mathbb{R}^{n_x} \to \mathbb{R}$ using noisy observations, $y = f(\mathbf{x}) + \nu$, where $\nu \sim \mathcal{N}(0, \sigma_\nu^2)$ is Gaussian distributed measurement stochasticity with zero mean and (possibly unknown) variance σ_ν^2. GPs themselves consider a distribution over functions and may be regarded as a generalization of multivariate Gaussian distributions:

$$f(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot,\cdot))$$

where the mean function $m(\cdot)$ can be interpreted as the deterministic part of the function; and the covariance function $k(\cdot,\cdot)$ accounts for correlations between the function values at different points.

One popular choice for the covariance function is the squared-exponential (SE) kernel [313]:

$$k(\mathbf{x}, \mathbf{x}') := \sigma_n^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^\intercal \mathbf{\Lambda}(\mathbf{x} - \mathbf{x}')\right)$$

where σ_n^2 is the covariance magnitude; and $\mathbf{\Lambda} := \text{diag}(\lambda_1 \cdots \lambda_{n_x})$ is a scaling matrix. Underlying this kernel choice is the assumption that the inferred function f is both smooth and stationary. If need be, other kernels can be selected, such as the Matérn class of covariance functions [313]. We furthermore choose a zero mean function:

$$m(\mathbf{x}) := 0$$

This choice is motivated by the fact that the training data is normalized, and therefore, without further information we assume the mean function has the value of the mean of the data.

Maximum likelihood estimation is commonly applied to infer a GP's hyperparameters $\mathbf{\Psi} := [\sigma_n \ \sigma_\nu \ \lambda_1 \ \ldots \ \lambda_{n_x}]^\intercal$, where σ_ν may be excluded in case the measurement noise variance is known. Consider n_D (noisy) function observations, denoted by $\mathbf{y} := [y_1 \cdots y_{n_D})]^\intercal \in \mathbb{R}^{n_D}$, with corresponding inputs gathered in the matrix $X := [\mathbf{x}_1 \ \cdots \ \mathbf{x}_{n_D}] \in \mathbb{R}^{n_x \times n_D}$. The log-likelihood of the observed data, ignoring constant terms, is given by:

$$\mathcal{L}(\mathbf{\Psi}) := -\frac{1}{2}\ln(|K(X)|) - \frac{1}{2}(\mathbf{y})^\intercal \, K(X)^{-1} \, (\mathbf{y})$$

with $K_{i,i'}(X) := k(\mathbf{x}_i, \mathbf{x}_{i'}) + \sigma_\nu^2 \delta_{i,i'}$ for all $(i, i') \in \{1..n_D\}^2$; and Kronecker's delta function $\delta_{i,i'}$.

The predicted distribution of $f(\mathbf{x})$ at an arbitrary input point $\mathbf{x}$, given the input-output data $(\mathbf{X}, \mathbf{y})$ and the maximum-likelihood estimates of $\mathbf{\Psi}$, follows a Gaussian distribution:

$$f(\mathbf{x}) \mid X, \mathbf{y} \sim \mathcal{N}(\mu_f(\mathbf{x}), \sigma_f^2(\mathbf{x})) \tag{A.7}$$

where the posterior mean function μ_f and the posterior variance function σ_f^2 are computed as:

$$\begin{aligned}
\mu_f(\mathbf{x}) &:= \mathbf{r}(\mathbf{x}, \mathbf{x}) \, K(X)^{-1} \, \mathbf{y} \\
\sigma_f^2(\mathbf{x}) &:= \sigma_n^2 - \mathbf{r}(\mathbf{x}, X) \, K(X)^{-1} \, \mathbf{r}(\mathbf{x}, X)^\intercal
\end{aligned}$$

with $\mathbf{r}(\mathbf{x}, X) := [k(\mathbf{x}, \mathbf{x}_1) \ \cdots \ k(\mathbf{x}, \mathbf{x}_{n_D})]$.

In practice, the mean μ_f corresponds to the GP's prediction at $\mathbf{x}$, while the variance σ_f^2 provides a measure of the uncertainty associated to this prediction (Figure 3.2a). Notice that multi-output formulation of the GP could be used instead to compute the non-diagonal terms of the covariance [11]. Common choices for the kernel $k_i(\cdot,\cdot)$ include the squared-exponential (SE) covariance function and Matérn class of covariance functions, with special cases the Matérn 3/2 and Matérn 5/2 [313] with the hyperparamters and the noise level being estimated using maximum likelihood estimation.

A.3 Algorithm convergence

When comparing the finite difference approximation methods, one can draw a clear distinction between the way Newton and BFGS (Figures 3.5a and 3.5b) converge and the way Adam (Figure 3.5c) converges. While ther former are 2nd order methods taking into account the local curvature, Adam resembles steepest descent by following down the local gradient. The Rosenbrock case study suggests that in lower-dimensional problems, the additional function evaluations are worth the second-order information to get to the optimum more quickly. Using Adam is expected to have superior behaviour in higher-dimensional problems where slower, but more consistent progress is achieved after less function evaluations at a time. This is explained through the complexity of derivative approximations. The approximation of gradients scales linearly as opposed to the quadratic scaling of Hessian approximations.

As a rule of thumb, the closer the sampling of next evaluations occurs to the current point, the more the convergence resembles that of gradient descent. This is the case for direct methods such as Nesterov (Figure 3.5d), where search directions are closely sampled, as well as for simplex (Figure 3.5e), suggesting that the polytopes for this function are rather narrow given the mathematical conditioner of the function.

Some model-based methods also display this behaviour of closely sampling around a consistent trajectory of what looks like gradient descent. This is for instance the case for CUATRO local and Py-BOBYQA (Figures 3.5k and 3.5i). Other, more global, model-based methods have more erratic convergence, which happens to be faster for the Rosenbrock function. CUATRO global (Figure 3.5j) for example uses extensive initial latin hypercube sampling and subsequent surrogate construction over trust regions covering most of the search space given its high initial radius. While SQSnobfit displays significant systematic search space exploration (Figure 3.5h), Bayesian Optimization (Figure 3.5h) manages to more efficiently trade off exploitation and exploration due

to its expected improvement function .

DIRECT-l (Figure 3.5f), as the only partition-based method, ignores the given initial starting point and starts off right in the middle of the search space. Oftentimes, this helps in speeding up convergence relative to other methods. The patterns displayed for the DIRECT method showcases a balance between global exploration and biased exploitation around promising solutions. This partition-based exploration following clear geometric patterns makes the method more consistent than more randomized methods.

A.4 Self-Optimizing reactor parameters of system

The parameters for the equation 3.19 are shown in Table A.1:

Table A.1: Self-optimizing case study: parameters of 3.19.

Parameters	Value
D	0.25
c_p	100
ρ	1000
κ	0.59
UA	$5\ 10^5$

B

Supplementary information for Chapter 4

B.1 Distributed optimization

Distributed optimization builds on the concepts of dual decomposition techniques (such as Lagrangean decomposition). For the sake of simplicity, we start from an equality-constrained, general convex optimization problem:

$$\min_{\mathbf{x}\in\mathbb{R}^{n_x}} \quad f(\mathbf{x}) \quad \text{s.t.} \quad A\mathbf{x} = \mathbf{b} \tag{B.1}$$

B.1.1 Method of Multipliers

The method of multipliers is essentially a more robust version of dual decomposition. It addresses the same family of problems, and goes through the same steps, but with the exception that the augmented Lagrangean is used rather than the Lagrangean: $\mathcal{L}(\mathbf{x}\,\lambda) = f(\mathbf{x}) + \lambda^{\intercal}(A\mathbf{x} - \mathbf{b}) + \frac{\rho}{2}||A\mathbf{x} - \mathbf{b}||_2^2$. Then, the following two steps are iterated over until convergence:

$$\begin{aligned} \mathbf{x}^{k+1} &= \arg\min_{\mathbf{x}} .\mathcal{L}(\mathbf{x}, \lambda^k) \\ \lambda^{k+1} &= \lambda^k + \rho(A\mathbf{x}^k - b) \end{aligned} \tag{B.2}$$

The method of multipliers converges under more relaxed conditions than dual ascent. Yet, the

quadratic term removes separability of the $\mathbf{x}$-update and the chance for decomposition, which is where ADMM comes in.

B.1.2 ADMM

In consensus optimization, the goal is to optimize a global objective consisting of n_a sub-objectives using global variables $\mathbf{x}$: $\min_{\mathbf{x}}. \sum_{a=1}^{n_a} f_a(\mathbf{x})$. Then, local copies of $\mathbf{x}$ are created $\mathbf{x}_a$ and the global variable $\mathbf{z}$ is introduced instead of $\mathbf{x}$: $\min_{\mathbf{x}}. \sum_{a=1}^{n_a} f_a(\mathbf{x})$ s.t. $\mathbf{x}_a - z = 0, \quad a = 1..n_a$. For these kinds of problems, the augmented Lagrangean takes the following form: $\mathcal{L}(\mathbf{x}, \lambda) = \sum_{a=1}^{n_a}(f_a(\mathbf{x}_a) + \lambda_a^\intercal(\mathbf{x}_a - \mathbf{z}) + \frac{\rho}{2}||\mathbf{x}_a - \mathbf{z}||_2^2)$. Rather than optimize over $\mathbf{x}$ and $\mathbf{z}$ jointly like in the method of multipliers, ADMM involves one Gauss-Seidel sweep:

$$\begin{aligned} \mathbf{x}_a^{k+1} &= \arg\min_{\mathbf{x_a}}. f_a(\mathbf{x}_a) + \lambda_a^\intercal(\mathbf{x}_a - \mathbf{z}) + \frac{\rho}{2}||\mathbf{x}_a - \mathbf{z}||_2^2 \\ \mathbf{z}^{k+1} &= \arg\min_{\mathbf{z}}. \mathcal{L}(\mathbf{x}^k, \mathbf{z}, \lambda^k) = \frac{1}{N}\sum_{a=1}^{n_a}(\mathbf{x}_a^{k+1} + \frac{1}{\rho}\lambda_a^k) \\ \lambda_a^{k+1} &= \lambda^k + \rho(\mathbf{x}_a^{k+1} - \mathbf{z}^{k+1}) \end{aligned} \tag{B.3}$$

This expression is equivalent to the following where the dual variables λ are replaced by the scaled dual variables ƛ:

$$\begin{aligned} \mathbf{x}_a^{k+1} &= \arg\min_{\mathbf{x_a}}. f_a(\mathbf{x}_a, \mathbf{z}_a) + \frac{\rho}{2}||\mathbf{z}_a - \mathbf{z}^k + \text{ƛ}_a^k||_2^2 \\ \text{ƛ}_a^{k+1} &= \text{ƛ}_a^k + \mathbf{x}_a^{k+1} - \mathbf{z}^{k+1}; \text{ where } \mathbf{z}^{k+1} = \frac{1}{N}\sum_{a=1}^{n_a}\mathbf{x}_a^{k+1} \end{aligned} \tag{B.4}$$

In short, ADMM brings together primal and dual decomposition while augmenting the Lagrangean with an additional quadratic penalty term: Like in dual decomposition, local copies of the shared variables are introduced and optimized privately in the subproblems. This step is then followed by an 'averaging' update on the shared variables. Finally, the Lagrange multipliers are updated leveraging subgradient information that can be extracted from the solution of the subproblems. Any remaining shared constraints can still be penalized in Lagrangean or quadratic penalty (proximity operator) terms. The introduction of local variables and the addition of proximity terms to the shared variables also make the problem robust to any local constraints: If the problem is infeasible for any set of shared variables, it will be in their local copy with a penalization in the objective. The convergence of ADMM is well-established on convex problems. As a rule of thumb, when nonconvexities are introduced, ADMM loses its convergence properties [50] but can

still often be applied in practice.

B.1.3 ALADIN

Houska et al. [181] and Kouzoupis et al. [217] introduced the Augmented Lagrangean based Alternating Direction Inexact Newton Method (ALADIN) to the following canonized form:

$\min_{\mathbf{x}}. \sum_{a=1}^{n_a} f_a(\mathbf{x}_a)$ s.t. $\sum_{a=1}^{n_a} A_a \mathbf{x}_a = \mathbf{b}$.

Their algorithm addresses convergence issues with ADMM on nonconvex problems. It exploits lower-level model information to leverage gradient information and hence improve the convergence rates:

$$\begin{aligned} \mathbf{x}_a^{k+1} &= \arg\min_{\mathbf{x_a}}. f_a(\mathbf{x}_a) + \lambda_a^{k,\intercal} A_a \mathbf{x}_a + \frac{\rho}{2} ||A_a(\mathbf{x}_a - \mathbf{z_a})||_2^2 \\ \Delta\mathbf{x} &= \arg\min_{\mathbf{x}}. \sum_{a=1}^{n_a} (\frac{1}{2} \Delta\mathbf{x}_a^\intercal H_a \Delta\mathbf{x}_a + \mathbf{g}_a^\intercal \Delta\mathbf{x}_a) \text{ s.t. } \sum_{a=1}^{n_a} A_a(\mathbf{x}_a^{k+1} + \Delta\mathbf{x}_a) = \mathbf{b} \\ \mathbf{z}^{k+1} &= \mathbf{x}^{k+1} + \Delta\mathbf{x}, \text{ and extract } \lambda^{k+1} \text{ as Lagrange multiplier of previous constraint} \end{aligned} \tag{B.5}$$

where $\mathbf{g}_a = \nabla \mathbf{f}_a(\mathbf{x}_a)$ and $H_a \approx \nabla^2 f_a(\mathbf{x}_a)$. As such, ALADIN alternates between a decentralised local nonconvex optimization step, followed by a centralised SQP relaxation step around the local decentralised solutions. ALADIN recovers SQP, ADMM, and dual decomposition as special cases when $\rho \to \infty$ (no infeasibility tolerated), $0 < \rho < \infty$ and $\rho = 0$ respectively. The Hessian H_a, usually the most computationally expensive step in SQP, does not have to be accurate and can be computed using cheaper approximations. However, the method requires white-box model expressions for the computation of $\mathbf{g}_a$, which might not always be possible.

B.2 Facility Location case study

The following facility location problem is inspired by Lara et al. [227] with some key differences. In particular, we assume the presence of two suppliers and markets each. We fix the number of facilities to be built to either one or two, which still gives rise to a Generalized Disjunctive Problem (GDP). We also define the distances between agents and facilities using the 1-norm, rather than the 2-norm. The GDP is finally reformulated into an MINLP using big-M constraints, implemented in Pyomo, and solved using Gurobi.

$$\min . \quad \sum_k C_k + \sum_i \sum_k C_{i,k} + \sum_k \sum_j C_{k,j} \tag{B.6a}$$

$$\text{s.t.} \quad Cost_k = ff_k + vf_k \; f_k, \quad \forall k \in \mathcal{K} \tag{B.6b}$$

$$0 \leq f_k \leq mc_K, \quad 0 \leq x_k \leq x_k^U, \quad 0 \leq y_k \leq y_k^u, \quad \forall k \in \mathcal{K} \tag{B.6c}$$

$$\begin{bmatrix} Z_{i,k} \\ C_{i,k} = cs_i \; f_{i,k} + ft_{i,k} + vt_{i,k} \; f_{i,k} \; D_{i,k} \\ 0 \leq f_{i,k} \leq f_{i,k}^U, \quad D_{i,k}^L \leq D_{i,k} \leq D_{i,k}^U \end{bmatrix} \vee \begin{bmatrix} \not{Z}_{i,k} \\ C_{i,k} = 0 \\ f_{i,k} = 0 \end{bmatrix} \quad \forall i \in \mathcal{I}, k \in \mathcal{K} \tag{B.6d}$$

$$\begin{bmatrix} Z_{k,j} \\ C_{k,j} = ft_{k,j} + vt_{k,j} \; f_{k,j} \; D_{k,j} \\ 0 \leq f_{k,j} \leq f_{k,j}^U, \quad D_{k,j}^L \leq D_{k,j} \leq D_{k,j}^U \end{bmatrix} \vee \begin{bmatrix} \not{Z}_{k,j} \\ C_{k,j} = 0 \\ f_{k,j} = 0 \end{bmatrix} \quad k \in \mathcal{K}, \forall j \in \mathcal{J} \tag{B.6e}$$

$$D_{i,k} = |x_i - x_k| + |y_i - y_k|, \quad \forall i \in \mathcal{I}, k \in \mathcal{K} \tag{B.6f}$$

$$D_{k,j} = |x_j - x_k| + |y_j - y_k|, \quad k \in \mathcal{K}, \forall j \in \mathcal{J} \tag{B.6g}$$

$$\sum_k f_{i,k} \leq a_i, \quad \forall i \in \mathcal{I}, \quad \sum_i f_{i,k} \; cv_k = f_k \quad \forall k \in \mathcal{K} \tag{B.6h}$$

$$f_k = \sum_j f_{k,j} \quad \forall k \in \mathcal{K}, \quad \sum_k f_{k,j} = d_j \quad \forall j \in \mathcal{J} \tag{B.6i}$$

$$Z_{i,k}, Z_{k,j} \in \{True, False\} \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, \forall k \in \mathcal{K} \tag{B.6j}$$

The objective is to find the location $\mathbf{x}_k, \mathbf{y}_k$, production $\mathbf{f}_k$, and connecting flows of all facilities $(\mathbf{f}_{i,k}, \mathbf{f}_{k,j})$ that minimize a total cost (B.6a). All of the above model quantities are variables, apart from the following parameters: the set of suppliers $i \in \mathcal{I}$ with fixed locations (x_i, y_i), availability a_i, and cost of material supply cs_i; the set of customers $j \in \mathcal{J}$ with fixed locations (x_j, y_j), and demands d_j. The shared cost parameters include: fixed and variable costs ff_k and vf_k of potential facilities $k \in \mathcal{K}$. These facilities have maximum capacity mc_k, and conversion to product flows cv_k. Other cost parameters include transportation costs between suppliers and facilities as well as facilities and customers, including fixed $ft_{i,k}, ft_{k,j}$ and cost $vt_{i,k}, vt_{k,j}$. Other variables in the problem are the distances between supplier and facility as well as between facility and customer $(D_{i,k}, D_{k,j})$. We use the following boolean variables $Z_{i,k}$ and $Z_{k,j}$ to denote if material is supplied from supplier i to facility k, and from facility k to customer j. These disjunctions can be reformulated using binary variables. Disjunctions (B.6d) and (B.6e), giving the cost of transportation from supplier to facility and from facility to customer respectively, can be reformulated using binary variables. The cost formulations also include bilinear terms. Each facility cost is given by (B.6b). Equations

(B.6f) and (B.6g) define the distances between suppliers, facilities, and customers using the 1-norm. Equations (B.6h) and (B.6i) constrain the supply to and from the facilities respectively.

B.2.1 Decomposition

We are considering two different decompositions, motivated by two separate business scenarios. In the first scenario, the 2 types of nodes, suppliers and markets, each consisting of one or two nodes, are part of the same legal entity and are able to share model information. For each problem, we need to find the following shared variables: The location of the facilities ($\mathbf{z}_1$ and $\mathbf{z}_2$ to denote $\mathbf{x}_k$ and $\mathbf{y}_k$), and their production ($\mathbf{z}_3$ to denote f_k). As such, the problem contains either three or six shared variables - depending on whether we want to build one or two facilities - and can be decomposed into two subproblems:

1) $F_1(\mathbf{z}) = \min_{\mathbf{z}}.$ $$\sum_k \frac{C_k}{2} + \sum_i \sum_k C_{i,k} + \frac{\rho}{2}||\mathbf{z}^I - \mathbf{z}||^2 \tag{B.7a}$$

s.t. $$Cost_k = ff_k + vf_k\ z^I_{3,k}, \quad \forall k \in \mathcal{K} \tag{B.7b}$$

$$0 \le z^I_{3,k} \le mc_K, \quad 0 \le z^I_{1,k} \le x^U_k, \quad 0 \le z^I_{2,k} \le y^u_k, \quad \forall k \in \mathcal{K} \tag{B.7c}$$

$$\begin{bmatrix} Z_{i,k} \\ C_{i,k} = cs_i\ f_{i,k} + ft_{i,k} + vt_{i,k}\ f_{i,k}\ D_{i,k} \\ 0 \le f_{i,k} \le f^U_{i,k}, \quad D^L_{i,k} \le D_{i,k} \le D^U_{i,k} \end{bmatrix} \vee \begin{bmatrix} \not{Z}_{i,k} \\ C_{i,k} = 0 \\ f_{i,k} = 0 \end{bmatrix} \quad \forall i \in \mathcal{I}, k \in \mathcal{K} \tag{B.7d}$$

$$D_{i,k} = |x_i - z^I_{1,k}| + |y_i - z^I_{2,k}|, \quad \forall i \in \mathcal{I}, k \in \mathcal{K} \tag{B.7e}$$

$$\sum_k f_{i,k} \le a_i, \quad \forall i \in \mathcal{I}, \quad \sum_i f_{i,k}\ cv_k = z^I_{3,k} \quad \forall k \in \mathcal{K} \tag{B.7f}$$

$$Z_{i,k} \in \{True, False\} \quad \forall i \in \mathcal{I}, \forall k \in \mathcal{K} \tag{B.7g}$$

(B.7h)

2) $F_2(\mathbf{z}) = \min_{\mathbf{z}}.$ $$\sum_k \frac{C_k}{2} + \sum_k \sum_j C_{k,j} + \frac{\rho}{2}||\mathbf{z}^{II} - \mathbf{z}||^2$$ (B.8a)

s.t. $$Cost_k = ff_k + vf_k \; z^{II}_{3,k}, \quad \forall k \in \mathcal{K}$$ (B.8b)

$$0 \leq z^{II}_{3,k} \leq mc_K, \quad 0 \leq z^{II}_{1,k} \leq x^U_k, \quad 0 \leq z^{II}_{2,k} \leq y^u_k, \quad \forall k \in \mathcal{K}$$ (B.8c)

$$\begin{bmatrix} Z_{k,j} \\ C_{k,j} = ft_{k,j} + vt_{k,j} \; f_{k,j} \; D_{k,j} \\ 0 \leq f_{k,j} \leq f^U_{k,j}, \quad D^L_{k,j} \leq D_{k,j} \leq D^U_{k,j} \end{bmatrix} \vee \begin{bmatrix} \not{Z}_{k,j} \\ C_{k,j} = 0 \\ f_{k,j} = 0 \end{bmatrix} \quad k \in \mathcal{K}, \forall j \in \mathcal{J}$$ (B.8d)

$$D_{k,j} = |x_j - z^{II}_{1,k}| + |y_j - z^{II}_{2,k}|, \quad \forall j \in \mathcal{J}, k \in \mathcal{K}$$ (B.8e)

$$z^{II}_{3,k} = \sum_j f_{k,j} \quad \forall k \in \mathcal{K}, \quad \sum_k f_{k,j} = d_j \quad \forall j \in \mathcal{J}$$ (B.8f)

$$Z_{k,j} \in \{True, False\} \quad \forall j \in \mathcal{J}, \forall k \in \mathcal{K}$$ (B.8g)

In our second scenario, we consider all of the four nodes (supplier and customer) to be their own separate legal entity with privacy considerations. As such, we need to decompose the problem into four. Unfortunately, Equations (B.6h) and (B.6i) prevent each agent from indepently deciding on the amount that is transported between their node and the facilities. As such, the transport variables $f_{i,k}$ and $f_{k,j}$ become part of the set of shared variables as $\mathbf{z}_4$ and $\mathbf{z}_5$ on top of $\mathbf{z}_1, \mathbf{z}_2$, and $\mathbf{z}_3$ of the previous decomposition. Each supplier agent i or customer agent j is now decomposed as one of the following two:

$$F_i(\mathbf{z}) = \min_{\mathbf{z}} . \quad \sum_k \frac{C_k}{2} + \sum_k C_{i,k} + \frac{\rho}{4}||\mathbf{z}^i - \mathbf{z}||^2 \quad \text{(B.9a)}$$

$$\text{s.t.} \quad Cost_k = ff_k + vf_k \; z^i_{3,k}, \quad \forall k \in \mathcal{K} \quad \text{(B.9b)}$$

$$0 \leq z^i_{3,k} \leq mc_K, \quad 0 \leq z^i_{1,k} \leq x^U_k, \quad 0 \leq z^i_{2,k} \leq y^u_k, \quad \forall k \in \mathcal{K} \quad \text{(B.9c)}$$

$$\begin{bmatrix} Z_{i,k} \\ C_{i,k} = cs_i \; z^i_{4,i,k} + ft_{i,k} + vt_{i,k} \; z^i_{4,i,k} \; D_{i,k} \\ 0 \leq z^i_{4,i,k} \leq f^U_{i,k}, \quad D^L_{i,k} \leq D_{i,k} \leq D^U_{i,k} \end{bmatrix} \vee \begin{bmatrix} \not{Z}_{i,k} \\ C_{i,k} = 0 \\ z^i_{4,i,k} = 0 \end{bmatrix} \quad \forall, k \in \mathcal{K} \quad \text{(B.9d)}$$

$$D_{i,k} = |x_i - z^i_{1,k}| + |y_i - z^i_{2,k}|, \quad \forall k \in \mathcal{K} \quad \text{(B.9e)}$$

$$\sum_k z^i_{4,i,k} \leq a_i, \quad \sum_i z^i_{4,i,k} \; cv_k = z^i_{3,k} \quad \forall k \in \mathcal{K} \quad \text{(B.9f)}$$

$$Z_{i,k} \in \{True, False\} \quad \forall k \in \mathcal{K} \quad \text{(B.9g)}$$

$$\text{(B.9h)}$$

$$F_j(\mathbf{z}) = \min_{\mathbf{z}} . \quad \sum_k \frac{C_k}{2} + \sum_k C_{k,j} + \frac{\rho}{4}||\mathbf{z}^j - \mathbf{z}||^2 \quad \text{(B.10a)}$$

$$\text{s.t.} \quad Cost_k = ff_k + vf_k \; z^j_{3,k}, \quad \forall k \in \mathcal{K} \quad \text{(B.10b)}$$

$$0 \leq z^j_{3,k} \leq mc_K, \quad 0 \leq z^j_{1,k} \leq x^U_k, \quad 0 \leq z^j_{2,k} \leq y^u_k, \quad \forall k \in \mathcal{K} \quad \text{(B.10c)}$$

$$\begin{bmatrix} Z_{k,j} \\ C_{k,j} = ft_{k,j} + vt_{k,j} \; z^j_{5,k,j} \; D_{k,j} \\ 0 \leq z^j_{5,k,j} \leq f^U_{k,j}, \quad D^L_{k,j} \leq D_{k,j} \leq D^U_{k,j} \end{bmatrix} \vee \begin{bmatrix} \not{Z}_{k,j} \\ C_{k,j} = 0 \\ z^j_{5,k,j} = 0 \end{bmatrix} \quad k \in \mathcal{K} \quad \text{(B.10d)}$$

$$D_{k,j} = |x_j - z^j_{1,k}| + |y_j - z^j_{2,k}|, \quad \forall k \in \mathcal{K} \quad \text{(B.10e)}$$

$$z^j_{3,k} = \sum_j z^j_{5,k,j} \quad \forall k \in \mathcal{K}, \quad \sum_k z^j_{5,k,j} = d_j \quad \text{(B.10f)}$$

$$Z_{k,j} \in \{True, False\} \quad \forall j \in \mathcal{J}, \forall k \in \mathcal{K} \quad \text{(B.10g)}$$

In the above formulations, we see that Equations (B.9f) and (B.8f) do not involve shared variables. As such, the equality constraints can be used to reduce the degree of freedom in the number of DFO decision variables, while the inequality constraints can reduce the search space for the coordinator on the remaining variables. For this specific problem, there are then $|K|$ elements in $\mathbf{z}_1$, $\mathbf{z}_2$ and $\mathbf{z}_3$, $|I| - 1$ elements in $\mathbf{z}_4$ and $|J| - 1$ elements in $\mathbf{z}_5$.

C

Supplementary information for Chapter 5

C.1 Optimization formulations

C.1.1 Planning formulation

Formulation (C.1) minimizes the planning objective (C.1a) subject to constraints (C.1b) to (C.1j) . The first term includes the material production and storage costs over all sites $l \in L = \{S1, S2, S3\}$, materials $s \in S = \{I1, I2, I21, I22, P1, P2, P3, P4, P5, P6\}$, and timesteps $t \in T = \{1, \ldots, 12\}$; The second term consists of the transport costs of I2 to sites S2 as I21 and to S3 as I23; The third term presents the income from selling the productfs $p \in I_p = \{P1, \ldots, P6\}$. Variable $P_{i,l,t}$ is the production of material i in location l at time t produced at a cost CP_i; Variable $S_{i,l,t}$ denotes the inventory of material i in location l at time t stored at a cost CS_i with initial inventory $S0_{i,l}$; Variable $TP_{l,tc}$ is the transport of material I2 to sites S2 or S3 at the transport control step $tc \in Tc = \{1, \ldots, 5\}$ at a cost CT_l; Variable $SA_{p,t}$ is the sales of product p at time t at the price of SP_p.

$$\min . \quad \sum_{i\in I,t\in T,l\in L}(CP_i P_{i,l,t} + CS_i S_{i,l,t}) + \sum_{l\in\{S2,S3\},tc\in Tc} CT_l TP_{l,tc} - \sum_{p\in I_p,t} SP_p SA_{p,t} \quad \text{(C.1a)}$$

$$\text{s.t.} \quad S_{i,l,t} = S_{i,l,t-1} + P_{i,l,t} - Cons_{i,l,t} - TR_{out,i,l,t} + TR_{in,i,l,t}, \quad i \in I, l \in L, t \in T \setminus \{0\} \quad \text{(C.1b)}$$

$$S_{i,l,0} = S0_{i,l} + P_{i,l,0} - Cons_{i,l,0} - TR_{out,i,l,0}, \quad i \in I, l \in L \quad \text{(C.1c)}$$

$$Cons_{i,l,t} = 1.1 \sum_{i'\in I_{out}(i)} Q_{i',l} P_{i',l,t}, \quad i \in I, l \in L, t \in T \quad \text{(C.1d)}$$

$$TR_{out,i,l,t} = \mathbb{1}_{i=I2,l=S1} \sum_{l'\in\{S2,S3\}} TP_{l,to\_tc(t)}, \quad i \in I, l \in L, t \in T \quad \text{(C.1e)}$$

$$TR_{in,i,l,t} = \mathbb{1}_{(i,l)\in\{(I21,S1),(I22,S2)\},t-LT_l\geq 0} \quad TP_{l,to\_tc(t-TT_{i,l})}, \quad i \in I, l \in L, t \in T \quad \text{(C.1f)}$$

$$P_{i,l,t} \leq M_{big} X_{l,i}, \quad i \in I, l \in L, t \in T \quad \text{(C.1g)}$$

$$U_{l,r} \sum_{p\in I_L(l)} Q_{p,l} P_{p,l,t} \leq A_{l,r}, \quad r \in R, l \in L, t \in T \quad \text{(C.1h)}$$

$$SA_{p,t} \leq D_{p,t}, \quad p \in I_p, t \in T \quad \text{(C.1i)}$$

$$S_{i,l,t} \geq S^*_p \text{ if } t > 4, \text{ else } S_{i,t} \geq 0.25 * S^*_p, \quad i \in I, l \in L, t \in T \quad \text{(C.1j)}$$

Constraints (C.1b) and (C.1c) define the inventory mass balances, where variable $Cons_{i,l,t}$ is the consumption of material i at site l and timestep t and is related in (C.1d) to the production of subsequent materials $I_{out}(i)$ via the conversion factor $Q_{i',l}$. Variables $TR_{out,i,l,t}, TR_{in,i,l,t}$ define the removal and addition of material i at site l and timestep t via transport as defined in (C.1e) and (C.1f): $TR_{out,i,l,t}$ is only relevant to I2 at site 1 and sums up the transport of material I2 to sites 2 and 3 at time t, whereas $TR_{in,i,l,t}$ is only relevant to I21 and I22 at sites 2 and 3 if material I2 was transported from site 1 LT_l (leadtime) steps away. $to\_Tc(\cdot)$ maps the 12 planning-level timesteps t to the 5 transport timesteps tc. Parameter $X_{l,i}$ denotes if a material i is produced at site l and sets the production of non-relevant materials for a given site to zero in (C.1g) with the help of a big-M constant M_{big}. $A_{l,r}$ in (C.1h) denotes the upper limit on the resource utilization $U_{l,r}$ associated with all production at a certain site l for a given resource r at a given time. (C.1i) upper bounds the sales of all products by their demand $D_{p,t}$. C.1j constrains the inventories of materials $S_{i,l,t}$ to be above their safe storage S^*_p at all timesteps, with slight relaxations in the first timesteps.

C.1.2 Scheduling formulation

The optimal planning targets $P_{p,S2,t}$ for each product $p \in \{P1, P2, P3, P4\}$ of site S2 in the 12 planning timesteps are fed to a separate scheduling formulation as P_s. All scheduling formulations optimize the makespan MS while penalizing the number of changeovers $Y_{i,i',n}$ required to achieve the planning targets as determined in the planning problem. We adapt Maravelias and Grossmann [249] and rely on the state-task network approach with a common continuous-time representation for all units. The formulation also accounts for variable batch sizes, variable processing times and sequence-dependent changeover times. The formulations for the assignment constraints (C.3), duration, finish time, and time-matching constraints (C.4), batch size constraints (C.5), and tightening constraints (C.6) are presented in great detail in Maravelias and Grossmann [249].

Objective:

$$MS + \rho \sum_{i \in I, i' \in I, n \in N} Y_{i,i',n} \tag{C.2a}$$

Variable MS denotes the makespan, and parameter ρ penalizes the sum of changeovers $Y_{i,i',n}$ from task $i \in I = \{I23M1, P1M1, P2M1, P3M1, P4M1, I23M2, P1M2, P2M2, P3M2, P4M2\}$ to task $I' \in I$ at scheduling event $n = \{1, \ldots, N_{last}\}$. N_{last} is 7 unless otherwise specified.

Assignment constraints:

s.t.
$$\sum_{i \in I_R(r)} Ws_{i,n} \leq 1, \quad \sum_{i \in I_R(r)} Wf_{i,n} \leq 1, \quad r \in R, n \in N \tag{C.3a}$$

$$\sum_{i \in I_R(r), n' \in N, n' \leq n} (Ws_{i,n'} - Wf_{i,n'}) \leq 1, \quad r \in R, n \in N \tag{C.3b}$$

$$\sum_{n \in N} (Ws_{i,n} - Wf_{i,n}) = 0, \quad i \in I \tag{C.3c}$$

$$\sum_{i' \in I_R(r)} Ws_{i',n} \geq Wf_{i,n}, \quad r \in R, i \in I, n \in N, n < N_{last}, i \in I_R(r) \tag{C.3d}$$

$$Wf_{i,0} = 0, Ws_{i,N_{last}} = 0, \quad i \in I \tag{C.3e}$$

Variables $Ws_{i,n}$ and $Wf_{i,n}$ denote if the processing of a task i, belonging to the set of tasks $I_R(r)$ that can be scheduled on a certain machine $r \in R = \{M1, M2\}$, starts or ends at a scheduling event n.

Duration, finish time, and time-matching constraints:

$$TCCH_{i,n} = \sum_{i' \in I} \kappa_{i,i'} Y_{i,i',n+1}, \quad i \in I, n \in N \setminus \{N_{last}\} \tag{C.4a}$$

$$TCCH_{i,N_{last}} = 0, \quad T_{N_{last}} = MS \tag{C.4b}$$

$$D_{i,n} = \alpha_i W s_{i,n} + \beta_i B s_{i,n}, \quad i \in I, n \in N \tag{C.4c}$$

$$Tf_{i,n} \geq Ts_{i,n} + D_{i,n} + TCCH_{i,n} - M_{big}(1 - Ws_{i,n}), \quad i \in I, n \in N \tag{C.4d}$$

$$Tf_{i,n} - Tf_{i,n-1} \leq M_{big} W s_{i,n}, \quad i \in I, n \in N \setminus \{0\} \tag{C.4e}$$

$$Tf_{i,n} - Tf_{i,n-1} \leq D_{i,n} + TCCH_{i,n}, \quad i \in I, n \in N \setminus \{0\} \tag{C.4f}$$

$$Ts_{i,n} = T_n, \quad i \in I, n \in N \tag{C.4g}$$

$$Tf_{i,n-1} \leq T_n + M_{big}(1 - Wf_{i,n}), \quad i \in I, n \in N \setminus \{0\} \tag{C.4h}$$

$$Tf_{i,n-1} \geq T_n - M_{big}(1 - Wf_{i,n}), \quad i \in I, n \in N \setminus \{0\} \tag{C.4i}$$

$$T_{n+1} \geq T_n, \quad n \in N \setminus \{N_{last}\} \tag{C.4j}$$

$$T_0 = 0 \tag{C.4k}$$

Variable $TCCH_{i,n}$ denotes the changeover duration for a task i at the end of the scheduling event n and depends on the sum of fixed changeover times $\kappa_{i,i'}$ from one task i to another i'. The changeover durations $TCCH_{i,n}$ and corresponding constraints were not initially accounted for in Maravelias and Grossmann [249]. Variable $D_{i,n}$ is the duration of the processing of task i that starts at event n. α_i denotes the fixed set-up duration for a task i that starts at event n. β_i denotes the variable duration per batch amount. There are three continuous variables that correspond to the batch size of task i at time point n: $Bs_{i,n}$ when it starts, $Bp_{i,n}$ when it is being processed, and $Bf_{i,n}$ when it finishes. $Ts_{i,n}$ and $Tf_{i,n}$ are the start and finish time of the processing of task i that starts at n. T_n denote the time of scheduling event n. M_{big} is a big-M constant.

Batch size constraints:

$$Bmin_i Ws_{i,n} \leq Bs_{i,n} \leq Bmax_i Ws_{i,n}, \quad i \in I, n \in N \tag{C.5a}$$

$$Bmin_i Wf_{i,n} \leq Bf_{i,n} \leq Bmax_i Wf_{i,n}, \quad i \in I, n \in N \tag{C.5b}$$

$$Bmin_i(\sum_{n' \in N, n' < n} Ws_{i,n'} - \sum_{n' \in N, n' \leq n} Wf_{i,n'}) \leq Bp_{i,n} \tag{C.5c}$$

$$Bp_{i,n} \leq Bmax_i(\sum_{n' \in N, n' < n} Ws_{i,n'} - \sum_{n' \in N, n' \leq n} Wf_{i,n'}) \tag{C.5d}$$

$$Bs_{i,n-1} + Bp_{i,n-1} = Bp_{i,n} + Bf_{i,n}, \quad i \in I, n \in N \setminus \{0\} \tag{C.5e}$$

$$Bcons_{i,s,n} = \xi_{i,s} Bs_{i,n}, \quad s \in S, n \in N, i \in I_{s,in}(s) \tag{C.5f}$$

$$Bprod_{i,s,n} = \xi_{i,s} Bf_{i,n}, \quad s \in S, n \in N, i \in I_{s,out}(s) \tag{C.5g}$$

$$Bcons_{i,s,n} \leq Bmax_i \xi_{i,s} Ws_{i,n}, \quad s \in S, n \in N, i \in I_{s,in}(s) \tag{C.5h}$$

$$Bprod_{i,s,n} \leq Bmax_i \xi_{i,s} Wf_{i,n}, \quad s \in S, n \in N, i \in I_{s,out}(s) \tag{C.5i}$$

$$S_{s,n} = S_{s,n-1} + \sum_{i \in I_{s,out}(s)} Bprod_{i,s,n} - \sum_{i \in I_{s,in}(s)} Bcons_{i,s,n}, \quad s \in S, n \in N \tag{C.5j}$$

$$S_{s,N_{last}} \geq P_s, \quad s \in S \tag{C.5k}$$

$$S_{s,0} = S0_s - \sum_{s, i \in I_{in}(s)} Bcons_{i,s,0}, \quad s \in S \tag{C.5l}$$

$$Cost_{CCH} = \sum_{i \in I, i' \in I, n \in N} Y_{i,i',n} \kappa_{i,i'} \tag{C.5m}$$

$Bmin_i$ and $Bmax_{i,n}$ denote the minimum and maximum batch sizes for a given task i. $Bcons_{i,s,n}$ and $Bprod_{i,s,n}$ are the amount of consumption and production of material $s \in S = \{I23, P1, P2, P3, P4\}$ in task i at event n and depend on the mass balance coefficient $\xi_{i,s}$. $S_{s,n}$ and $S0_s$ are the available and initial inventory of material s. The planning targets P_s for each material s are set by the planning problem (apart from I23, which is set to 0). $Cost_{CCH}$ denotes the optimal changeover cost used to correct the planning-level scheduling cost estimate.

Tightening constraints:

$$\sum_{n \in N, i \in I_R(r)} (D_{i,n} + TCCH_{i,n}) \leq MS, \quad r \in R \tag{C.6a}$$

$$\sum_{n' \in N, i \in I_R(r), n' \geq n} (D_{i,n'} + TCCH_{i,n}) \leq MS - T_n, \quad r \in R, n \in N \tag{C.6b}$$

$$\sum_{n' \in N, i \in I_R(r), n' < n} (\alpha_i Wf_{i,n'} + \beta_i Bf_{i,n'} + TCCH_{i,n'}) \leq T_n, \quad r \in R, n \in N \tag{C.6c}$$

$$Y_{i,i',n} \geq Wf_{i,n} + Ws_{i',n} - 1, \quad r \in R, n \in N, i \in I_R(r), i' \in I_R(r), \kappa_{i,i'} > 0, n < N_{last} \tag{C.6d}$$

C.1.3 Control formulation

All productions $m \in \{I3, P1, P2, P3, P4\}$ are modelled according to the chemical reaction $A \rightarrow B \rightarrow C$. A separate optimal control formulation (C.7) is formulated for each batch to optimize a mix of optimal processing time t_f and final energy consumption $E(1)$ (C.7a). Constraints (C.7b) to (C.7d) give the differential equations that determine the concentration profiles $c_A(t), c_B(t), c_C(t)$ as a function of time. The rate of reaction of A to B is first-order in the state variable concentration of A as well as the cooling flowrate control variable $u(t)$ with production-specific reaction constant $k_{1,m}$. Similarly, the rate of reaction of B to C is first-order in the concentration of B but proportional to $u(t)^{0.8}$ instead with reaction constant $k_{2,m}$. (C.7f) determines minimal terminal concentrations of B and C that are proportional to the batch target $Bprod_m$ set by the optimal scheduling decisions and normalized by parameters V_B and V_C. (C.7g) denote the three initial concentrations, and (C.7h) denote path constraints on the ranges of $c_A(t), c_B(t), c_C(t)$. (C.7e) determines the differential expression with respect to time for the energy consumption $E(t)$ to be proportional with respect to the cooling flowrate $u(t)$ and to V^2. Finally, (C.7i) limits the range of the independent time variable t, the control $u(t)$, and the processing time t_f.

$$\min. \quad 2t_f + 0.5E(1) \tag{C.7a}$$

$$\text{s.t.} \quad \frac{\partial c_A(t)}{\partial t} = -t_f k_{1,m} u(t) c_A(t) \tag{C.7b}$$

$$\frac{\partial c_B(t)}{\partial t} = t_f (k_{1,m} u(t) c_A(t) - k_{2,m} u(t)^{0.8} c_B(t)) \tag{C.7c}$$

$$\frac{\partial c_C(t)}{\partial t} = k_{2,m} u(t)^{0.8} c_B(t)) \tag{C.7d}$$

$$\frac{\partial E(t)}{\partial t} = t_f V^2 u(t) \tag{C.7e}$$

$$c_B(1) \geq \frac{Bprod_m}{V_B}, \quad c_C(1) \geq \frac{Bprod_m}{V_C} \tag{C.7f}$$

$$c_A(0) = 17, \quad c_B(0) = 0, \quad c_C(0) = 0 \tag{C.7g}$$

$$c_A(t), c_B(t), c_C(t) \in [0, 100] \tag{C.7h}$$

$$t \in [0, 1], \quad u(t) \in [1, 9], \quad t_f \in [0, 500] \tag{C.7i}$$

C.2 Supplementary results

C.2.1 Integrating scheduling and control

Figure C.1 gives an example of the predicted and actual optimal processing time and energy cost samples of the optimal control training set for one of the products. The mapping for both predictions is well-behaved, almost linear with a slight concave curvature. This mapping could be approximated reasonably well with linear or piecewise-affine linear regression techniques. This goes to show that even though the dynamics of the optimal control problem might be nonlinear, the relationship between the connecting variables in the integration of scheduling and control can be captured reasonably well with simple surrogates. In our case, a small neural network with 5 neurons for the two hidden layers each is sufficient.

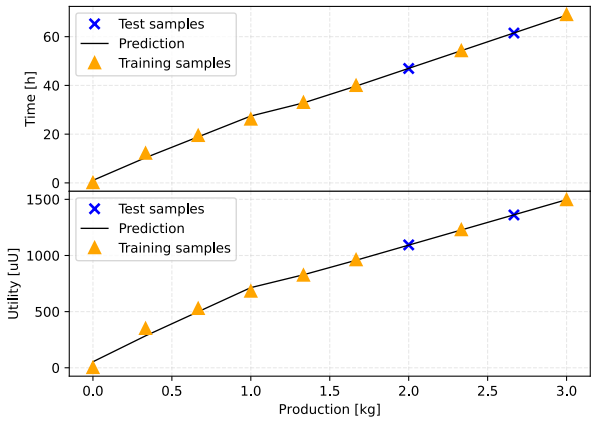


Figure C.1: Predicted and actual optimal batch processing time and energy requirements corresponding to 10 uniformly sampled training and test batch targets.

Table C.1 summarizes the on- and off-line optimization times involved in the solution of the scheduling-only problem (5.10) and the scheduling problem with embedded optimal control surrogates (5.11). Solving these problems constitutes a key part in obtaining the bi and approx tri evaluations (5.8b) and (5.8c) respectively. The online optimization time increases from 10 seconds to over 4 minutes. Even though the surrogates are small, the approx evaluation requires the enumeration of each one of the 5 products by 7 events by 2 machines optimal batch production surrogates. Each of the 70 surrogates introduces discrete variables that lead to an explosion in solution time. While the online optimization time increases significantly after integration of the surrogates, the off-line control problem sampling and training of the neural network remain negligible. Training the neural network for 1,000 epochs takes at most a minute. Given the well-behaved

nature of the mapping, we only require 10 control samples for each machine-product assignment, adding another one-off 5 minutes that are negligible in the overall workflow.

Table C.1: Online optimization, control surrogate sampling and surrogate training times involved in the solution of the scheduling-only and scheduling with embedded control surrogates problems

	Time		
	Optimization	Control sampling (N=10x5)	Training
Scheduling only	10 sec	/	/
Scheduling-control surrogate	250 sec	5 min	1 min

C.2.2 Optimality surrogate training samples

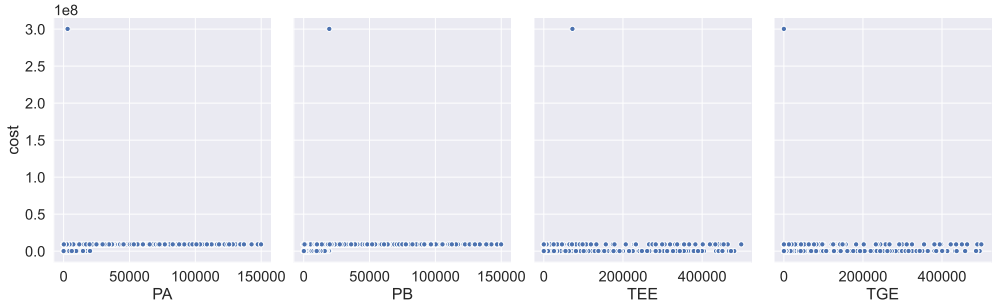
Figure C.2 displays the optimal scheduling cost mapping as a function of the complicating planning target variables for the scheduling-only, integrated scheduling-control, and hierarchical scheduling-control optimality surrogates. The figure showcases that the infeasibilities in the lower levels lead to an ill-behaved, nonlinear mapping between optimal responses and planning targets. Additional challenges arise for the scheduling-only and hierarchical samples, where outliers can have scheduling costs an order of magnitude higher, forcing the surrogates to accurately capture outliers rather than trends in the region we are more interested in.

Figures C.3 to C.5 show the pairwise plots for all variables involved in the optimality surrogate training, filtered for feasible samples. The L-shaped correlations between planning target variables are indicative of the log-sampling approach used for training. While the optimal cost seems to be mostly driven by linear relationships with respect to PA and PB, the mapping is still nonlinear with subtle dependencies on TEE and TGE.

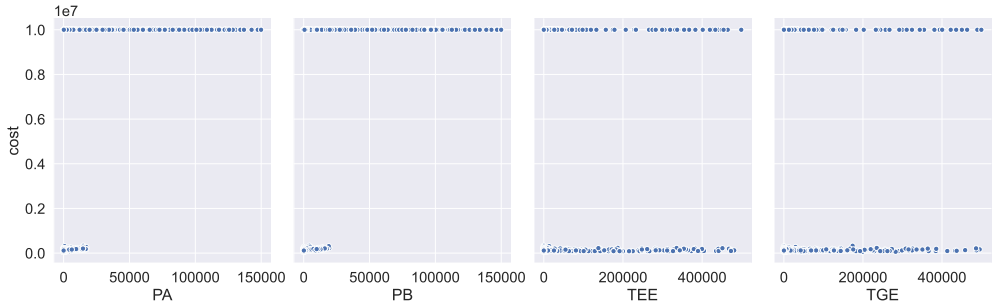
C.2.3 Surrogate sensitivity analysis on bi-level integrated planning-scheduling problem

We instantiate surrogates where we fix the size of the second hidden layer and gradually increase the number of nodes from 5 to 90. We train each model, and optimize the corresponding integrated bi-level planning-scheduling problem with embedded surrogates. We then record the corresponding test accuracy, integrated optimization time, integrated optimization result, and tri metric evaluation of said result.

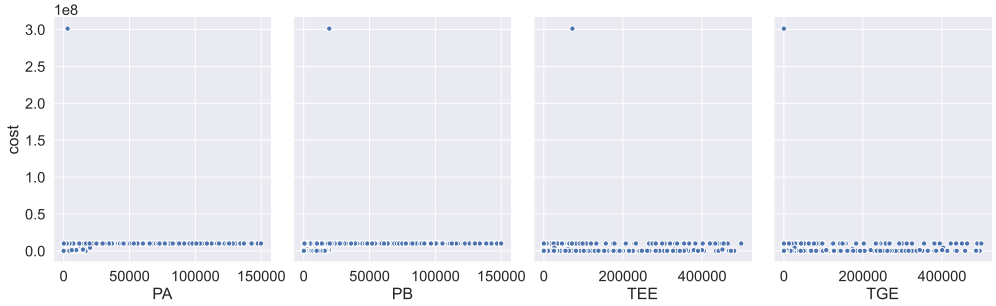
While we do not see any neat linear relationships between the above metrics and the surrogate size, we still gain interesting insights. The best optimization results translate into the best tri



(a) Scheduling-only surrogates



(b) Integrated scheduling-control surrogates



(c) Hierarchical scheduling-control surrogates

Figure C.2: Optimal scheduling cost versus planning target samples for optimality surrogate training.

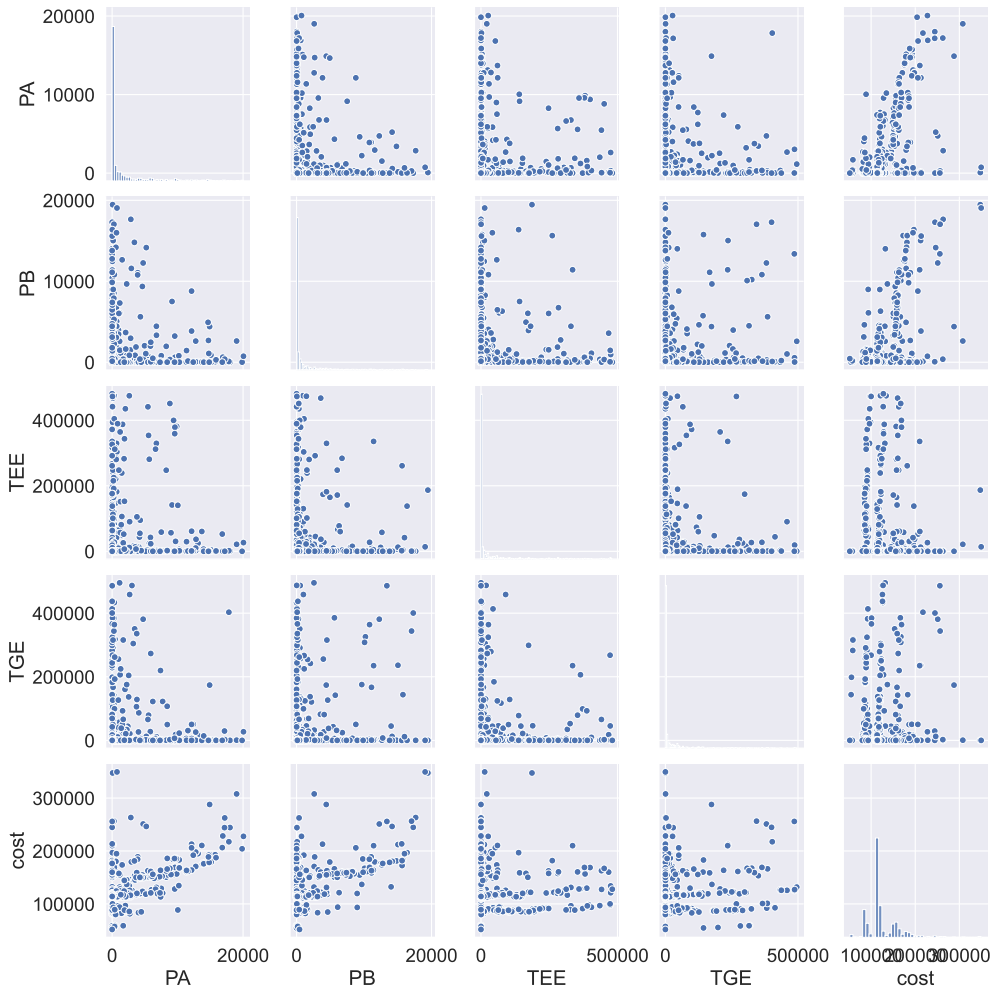


Figure C.3: Pair-wise plots for the feasible samples used for the training of the scheduling-only optimality surrogates.

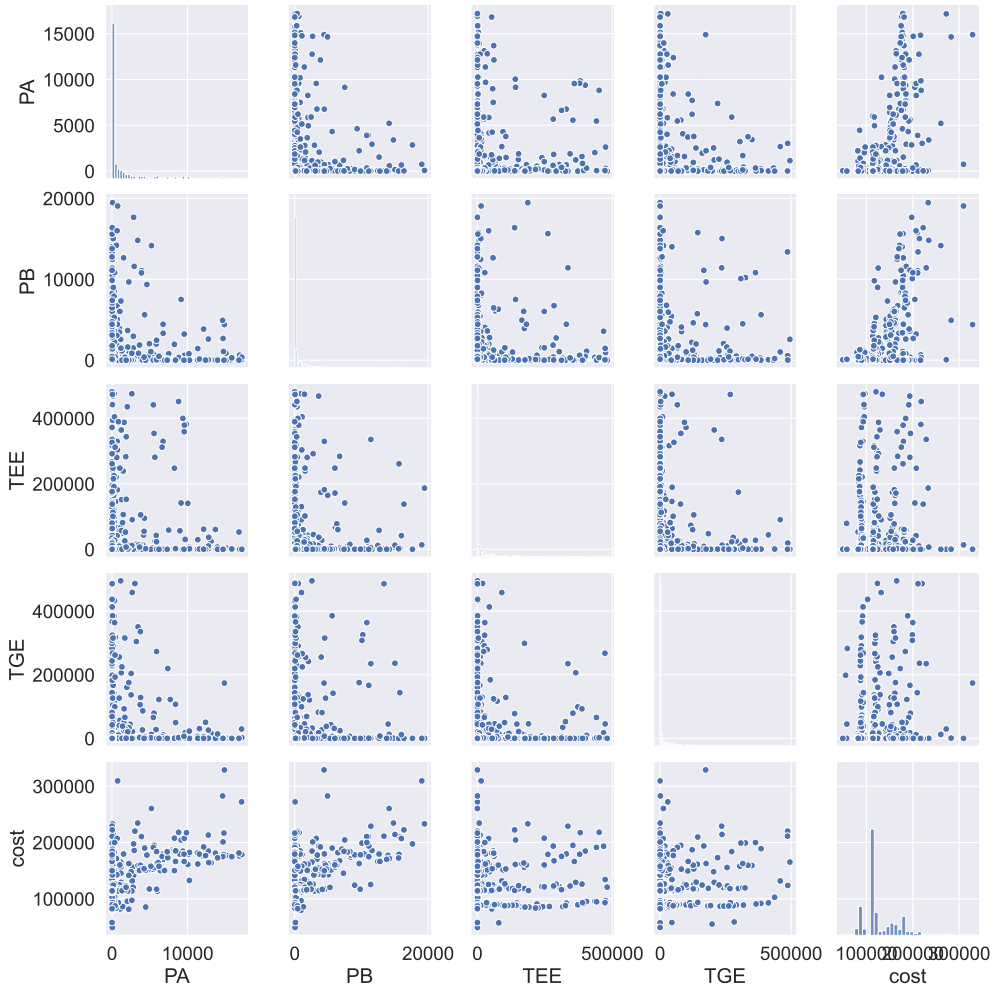


Figure C.4: Pair-wise plots for the feasible samples used for the training of the integrated scheduling-control optimality surrogates.

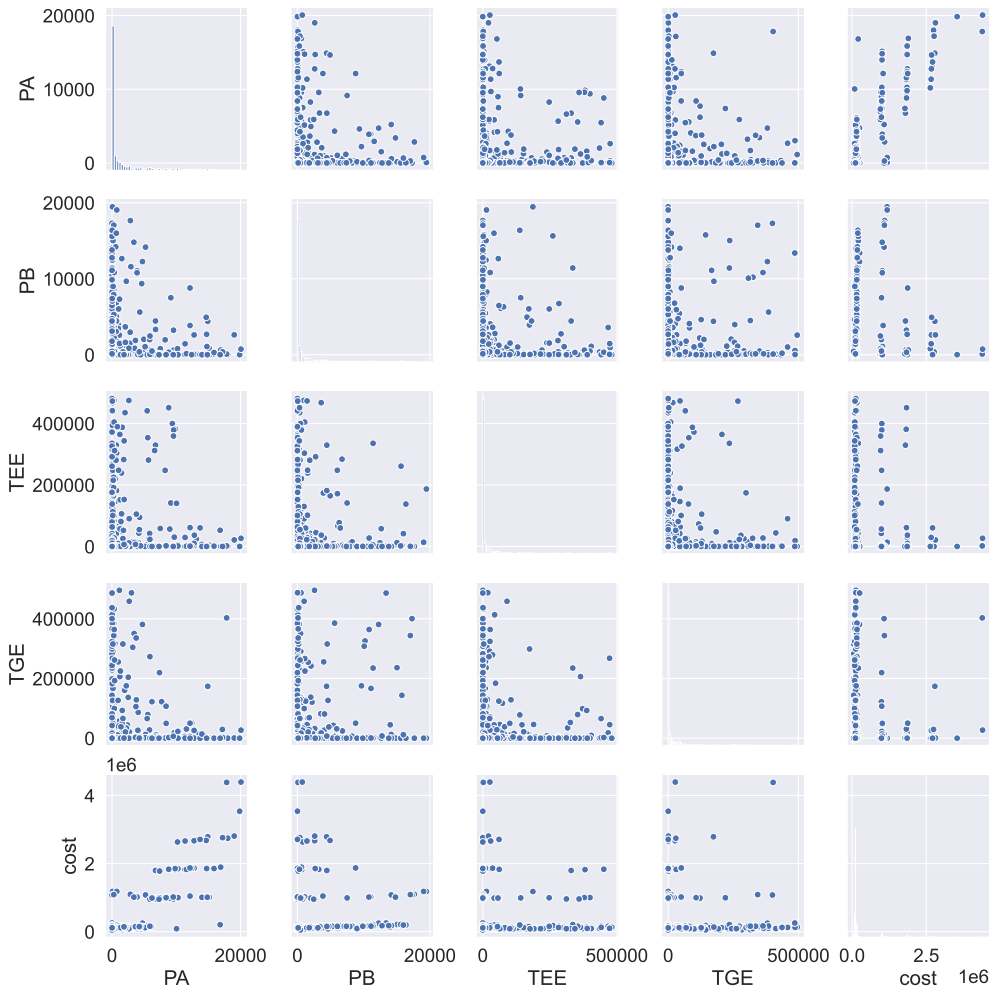


Figure C.5: Pair-wise plots for the feasible samples used for the training of the hierarchical scheduling-control optimality surrogates.

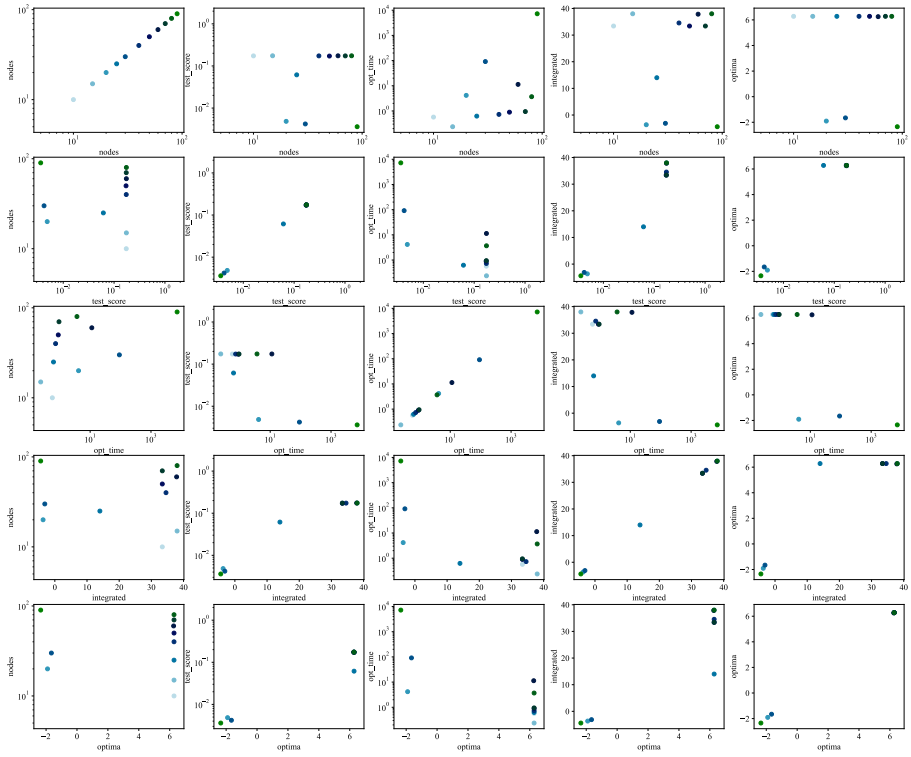


Figure C.6: Pairwise key performance indicators as we increase the size of the surrogates in solving an integrated planning-scheduling problem

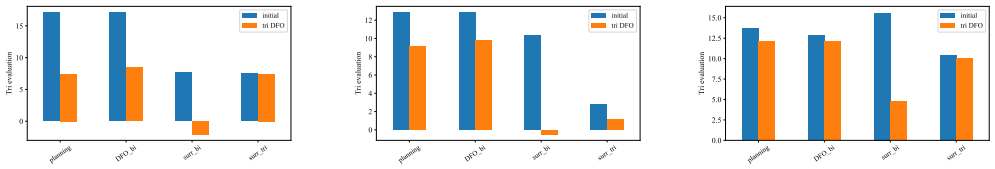
metric evaluations. However, the almost uniform distribution of optimization results gives rise to a sharp bimodal distribution in feasible and infeasible tri metric evaluations. Improving the test accuracy in the training leads to a clear improvement in both. While the best surrogate accuracy and results are found with the bigger surrogates, increasing surrogate size does not lead to a monotonic improvement in these metrics, partly due to the aforementioned bimodal distribution. As expected, this suggests that there is a certain *minimum acceptable* accuracy a surrogate needs to have to lead to good tri metric evaluations. Although surrogate size does not guarantee accuracy, good performance can only be achieved for architectures of a minimum size.

Ideally, we would also quantify the sensitivity of the surrogate performance with respect to different training configurations or slightly updated/modified data, which calls for additional work on model validation or robust/adversarial selection approaches.

C.2.4 Qualitative discussion on scaling with relative variable size

It goes without saying that an increase in the size of any level severely degrades the tractability of the integrated method. An increase in the number of complicating variables relative to the scheduling and control formulation size could enable more temporal, spatial, and equipment-wise parallelization in the integrated metric evaluations (useful for bi/approx/tri DFO, and surrogate training samples). However, we quickly run into combinatorial bottlenecks in the number of required compute nodes, especially given their extensive shared memory usage. In the surrogate approach, the tractability of the integrated problem is mostly dependent on the size and number of embedded surrogates. As we increase the dimensionality of complicating variables, we either need to embed *more* or *bigger* surrogates into upper levels, running the risk of rendering the on-line integrated problem intractable. In the DFO approach, we run the risk of the budget becoming too small to make meaningful progress on the solution space as we increase the expense of metric evaluations (scheduling and control problem size), or increase the number of decision variables (planning-only and planning-scheduling variables). Consequently, while the surrogate approach generally is more tractable than the DFO approach if we can find simple, accurate surrogates, it would be too simplistic to say that the surrogate approach always scales better than the DFO approach. Their relative performance as we increase problem-size remains problem-specific.

C.2.5 Benchmarking on perturbed problem instances



(a) Perturbed instance 1 (b) Perturbed instance 2 (c) Perturbed instance 3

Figure C.7: Comparing the best solutions from the three methodologies on perturbed instances

C.3 Planning profiles for best solutions

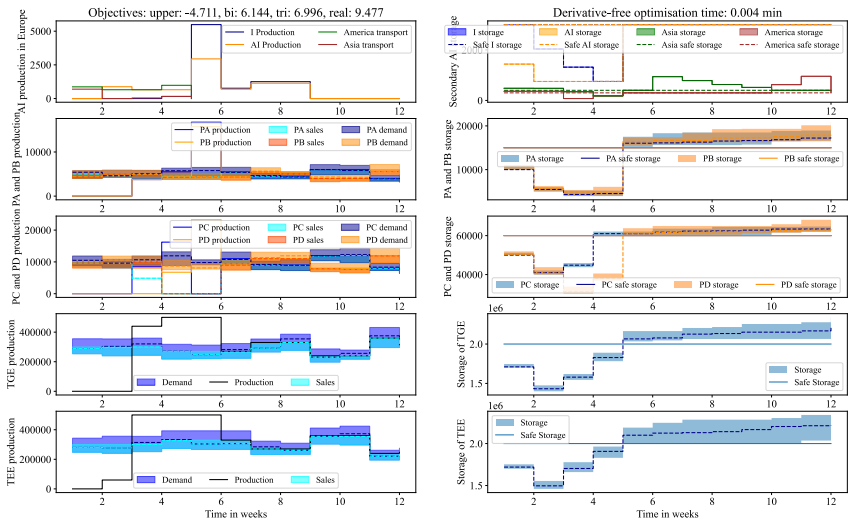
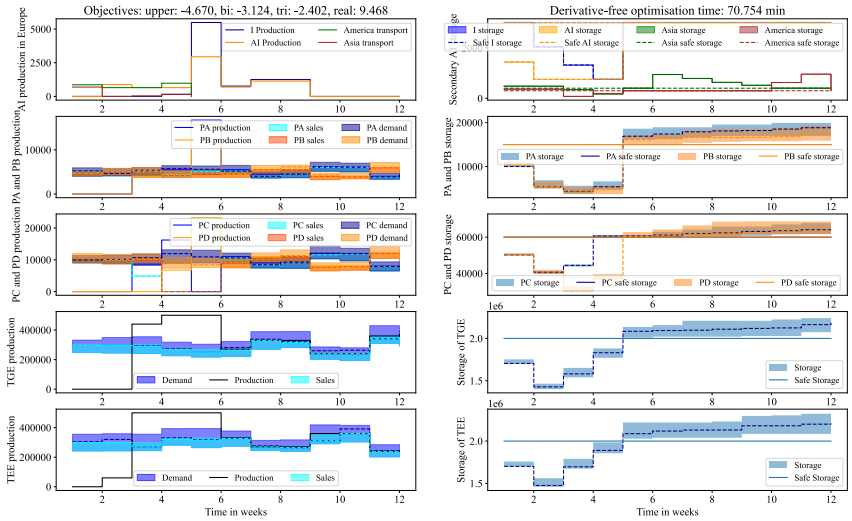
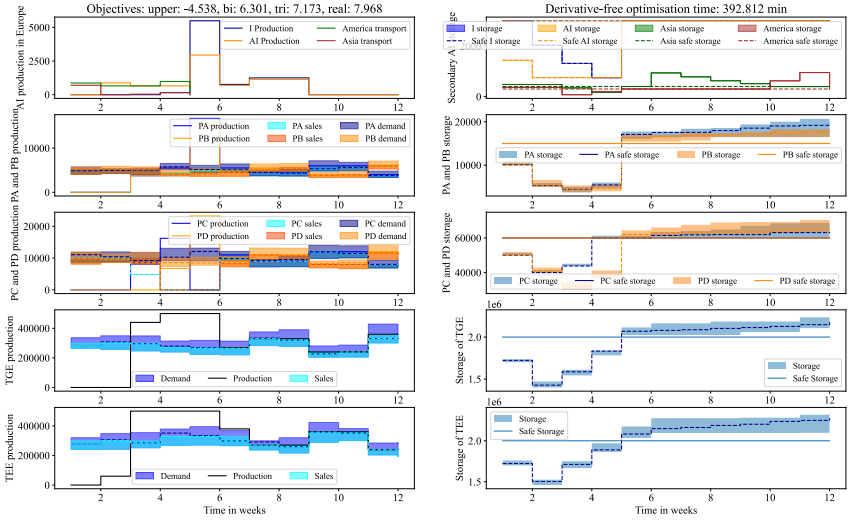


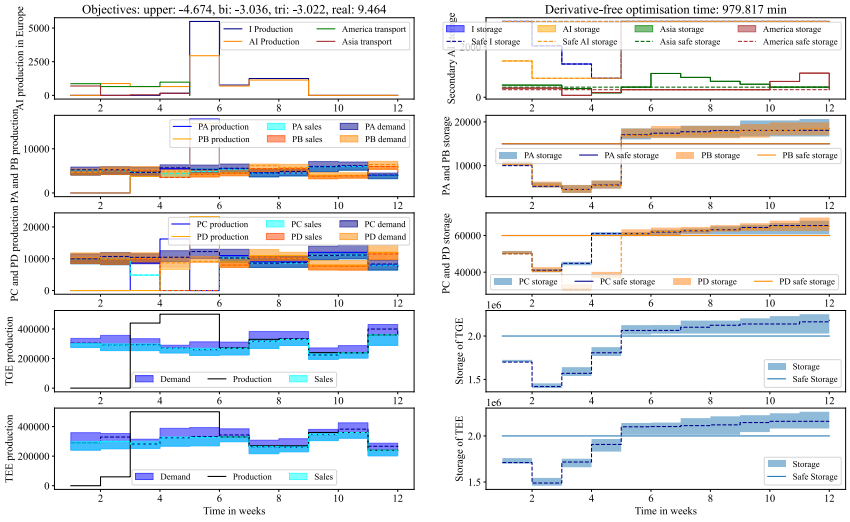
Figure C.8: Solution dashboard and planning profile of the planning-only problem



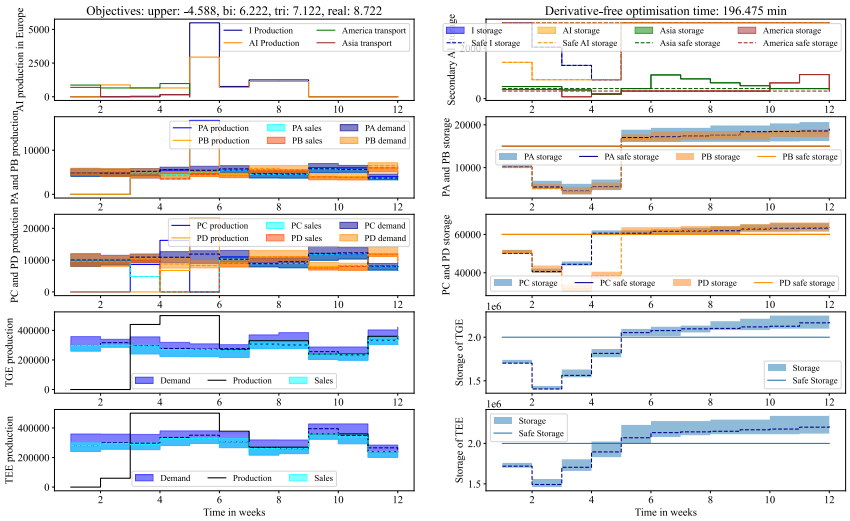
(a) Solution dashboard and planning profile of the bi DFO approach on the problem



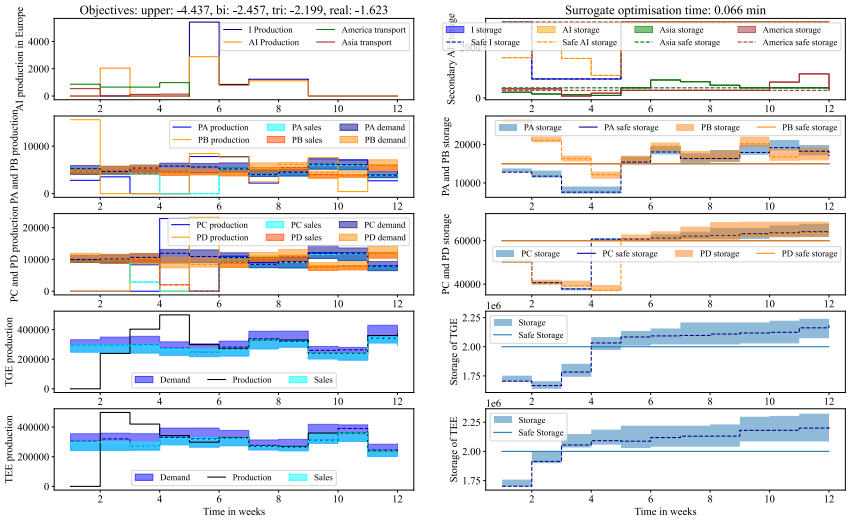
(b) Solution dashboard and planning profile of the tri DFO approach starting from the approx DFO solution



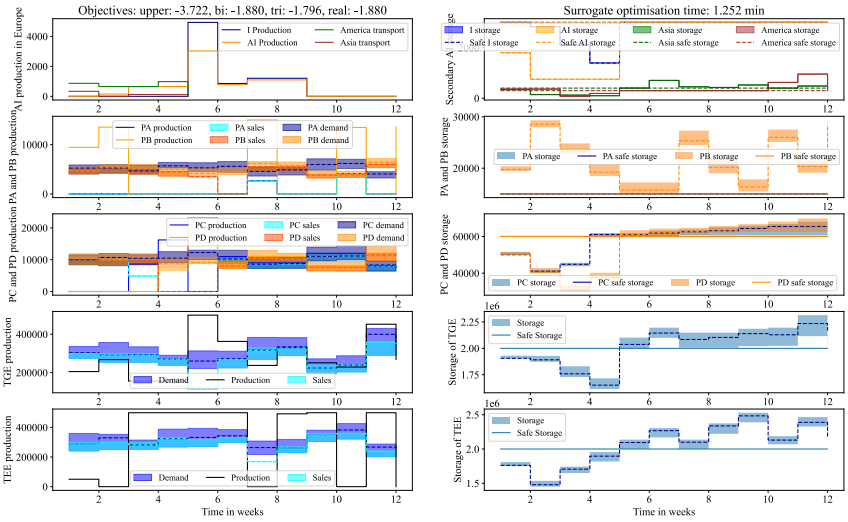
(a) Solution dashboard and planning profile of the approx DFO approach on the problem



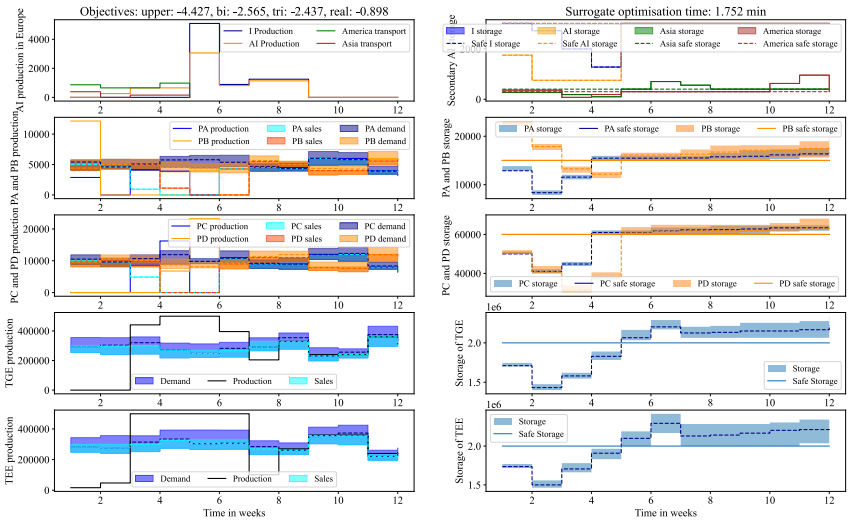
(b) Solution dashboard and planning profile of the tri DFO approach starting from the approx DFO solution



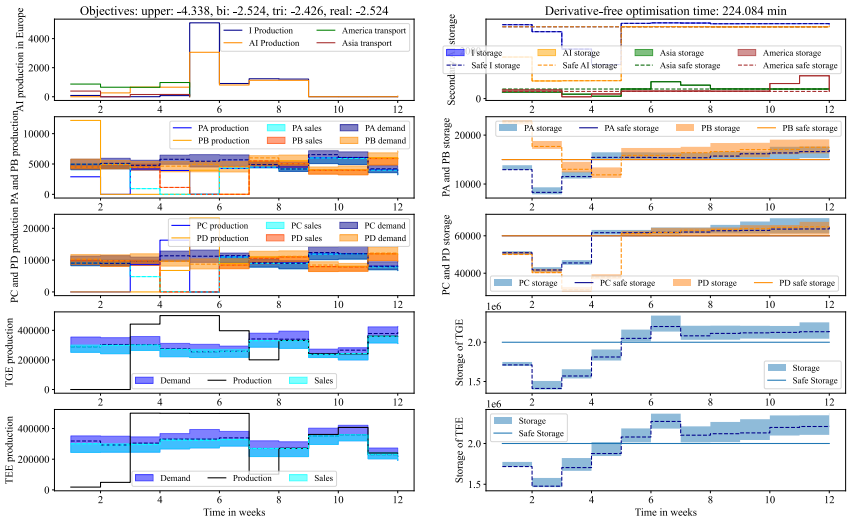
(a) Solution dashboard and planning profile of the best approx surrogate approach



(b) Solution dashboard and planning profile of the best tri surrogate approach



(a) Solution dashboard and planning profile of the best bi surrogate approach



(b) Solution dashboard and planning profile of the tri DFO approach starting from the best bi surrogate approach

D

Supplementary information for Chapter 6

D.1 Formulations

D.1.1 PID controller tuning

$$\min. \quad \int_0^1 [(T(t) - T_{sp}(t))^2 + aF_{in}(t)^2 + bT_c(t)^2 + cF'_{in}(t)^2 + dT'_c(t)^2]\, dt \tag{D.1a}$$

$$\text{s.t.} \quad \frac{\partial m_A(t)}{\partial t} = F_{in}(t)C_{A_f}(t) - r_A(t)V(t) \tag{D.1b}$$

$$\frac{\partial m_B(t)}{\partial t} = (r_A(t) - r_B(t))V(t) \tag{D.1c}$$

$$\frac{\partial m_C(t)}{\partial t} = r_B(t)V(t) \tag{D.1d}$$

$$\frac{\partial V(t)}{\partial t} = F_{in}(t) \tag{D.1e}$$

$$\frac{\partial T(t)}{\partial t} = \frac{F_{in}(t)}{V(t)}(T_f - T(t)) + \frac{\Delta H_{AB}}{\rho C_p} r_A(t) + \tag{D.1f}$$

$$\frac{\Delta H_{BC}}{\rho C_p} r_B(t) + \frac{UA}{V(t)\rho C_p}(T_c(t) - T(t)) \tag{D.1g}$$

$$r_A(t) = k_{0,AB} \exp\left(-\frac{E_{AB}}{RT}\right) C_A(t) \tag{D.1h}$$

$$r_B(t) = k_{0,BC} \exp\left(-\frac{E_{BC}}{RT}\right) C_B(t) \tag{D.1i}$$

$$m_i(t) = C_i(t)V(t), \quad i \in \{A, B, C\}, \quad t \in [0, 1] \tag{D.1j}$$

- $T_c(t)$: Temperature of cooling jacket (K)
- $F_{in}(t)$: Inlet flow rate (m^3/s)
- $T_c'(t)$: Change in temperature of cooling jacket (K/s)
- $F_{in}'(t)$: Change in inlet flow rate (m^3/s^2)
- $m_i(t)$ (for $i \in \{A, B, C\}$): Molar mass of species i (mol)
- $C_i(t)$ (for $i \in \{A, B, C\}$): Concentration of species i (mol/m^3)
- $T(t)$: Temperature in CSTR (K)
- T_f: Feed temperature (K)
- C_{A_f}: Feed concentration of A (mol/m^3)
- V: Reactor volume (m^3)
- ρ: Density of mixture (kg/m^3)
- C_p: Heat capacity of mixture ($J/kg \cdot K$)
- UA: Overall heat transfer coefficient (W/K)
- ΔH_j (for $j \in \{AB, BC\}$): Heat of j reaction (J/mol)
- E_j (for $j \in \{AB, BC\}$): Activation energy for j reaction (J/mol)
- $k_{0,j}$ (for $j \in \{AB, BC\}$): Pre-exponential factor for j reaction (1/s)
- a,b,c,d: weighting factors for objective terms (/)