Distributed Optimization of Constraint-Coupled Systems via Approximations of the Dual Function

Vom Fachbereich Maschinenbau und Verfahrenstechnik

der Rheinland-Pfälzischen Technischen Universität Kaiserslautern-Landau

zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

genehmigte

Dissertation

von

Herrn

M.Sc. Vassilios Yfantis

aus Düsseldorf

Tag der mündlichen Prüfung: 20.02.2024

Dekan:	Prof. Dr. rer. nat. Roland Ulber

Promotionskommision:

Vorsitender:	Jun. Prof. Dr.-Ing. Patrick Ruediger-Flore
1. Berichterstatter:	Prof. Dr.-Ing. Martin Ruskowski
2. Berichterstatter:	Prof. Dr.-Ing. Sebastian Engell

D 386

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und ohne unerlaubte, fremde Hilfe angefertigt habe. Alle Ausführungen, die aus anderen Schriften wörtlich oder sinngemäß übernommen wurden, sind kenntlich gemacht. Alle verwendeten Quellen sind im Literaturverzeichnis zitiert.

Oldenburg, den 26. Februar 2024

Vassilios Yfantis

Contents

Acknowledgements

I always enjoyed reading the acknowledgements of a thesis, especially when they somewhat reflected on the journey of the author, even though I often found them to be a bit overly "melodramatic". However, after going through all of this myself I find myself with a greater appreciation for emotional acknowledgements. So, I will give it my best try to capture that essence in the following lines.

First I want to sincerely thank my examiners Prof. Dr.-Ing. Martin Ruskowski and Prof. Dr.-Ing. Sebastian Engell. I thank Prof. Ruskowski for giving me the opportunity to work at his chair after I found myself in a "less-than-ideal" situation. The freedom and trust he provided in addition to his interest in and support of my work were instrumental for the success of this thesis. I am very grateful for the excellent working environment and opportunities that he provided me over the years. I also want to thank Prof. Engell for being much more than a co-examiner of the thesis. He has undoubtedly had the biggest influence on my scientific career throughout the years. I am very grateful for his guidance and constructive criticism and feedback. The fact that he continued supporting me even after I left his chair due to circumstances in my personal life speaks volumes about his character. The successful completion of this thesis proves to me that my choice to move to Dortmund for my master's studies with the express intend to work for Prof. Engell later down the road was a correct one. I would also like to thank Jun. Prof. Dr.-Ing. Patrick Ruediger-Flore for agreeing to chair the examination commitee.

Even though it might be a bit unusual, I have to thank Prof. Dr.-Ing. habil. Rudibert King at this point as well. Before I attended his lectures I was seriously considering switching my field of studies and starting from scratch. I thank him for being by far the best lecturer I have ever attended and for making me realize that I can actually work on mathematical topics even within chemical engineering and not only on organic chemistry and thermodynamics. His lectures were my first contact with optimization and they made me want to pursue a PhD in that direction. Without his recommendation to study under Prof. Engell my life would probably have turned out quite differently.

Having worked at three different chairs during my time as a PhD student means that I have had a large number of great people to work with, all of whom deserve to be acknowledged at this point. Listing all of them would make this part of the thesis even longer than it already will end up being, so I will refrain from doing that. Nevertheless,

a few names have to be explicitly mentioned here. From the Process Dynamics and Operations group in Dortmund I thank my office mate Max Cegla, Tim Janus, the alpha-office duo Marc Kalliski and Clemens Lindscheid, as well as all other members of the DYN group for the somewhat short, but still enjoyable time we spent together. From the Mechatronics chair in Kaiserslautern I want to thank Argtim Tika, Iqra Batool, Patrick Bachmann, Michael Danner, Andrea Scheck and all the others for creating a great working environment, despite the "circumstances". I also want to thank Prof. Dr. Naim Bajcinca, without whom I would probably not have found my way to Kaiserslautern and who showed me the difference between good and bad scientific practice.

The bulk of my time as a PhD student was spent at the Chair of Machine Tools and Control Systems. I want to thank all members of the chair for the great years I spent there. This includes Tatjana Legler, Magnus Volkmann, Jonas Weigand, Julian Götz, Andreas Wagner, Simon Lamoth, Vinit Hegiste, Leonhard Kunz, Thomas Barth, Jonathan Nussbaum, Jonas Brozeit, Khalil Abuibaid and Abdullah Farrukh. Special thanks also goes to my former student Mario Klostermeier for continuing my work, finally putting it into decent code, and for his feedback on the defense presentation. I also want to thank all the colleagues from the Innovative Factory Systems group at DFKI and the *SmartFactory*$^{\mathrm{KL}}$, especially Jens Popper and William Motsch for the great collaboration and Team Hamilton, that I was briefly a part of. A big thank you also goes to PD Dr. Achim Wagner for his constant interest in and great discussions about my research. And of course our premium HiWis Patrick Kremser and Nils Bruder deserve a lot of credit for being the heart and soul of WSKL and for the countless delicious meals they prepared for us.

So far I have left out a few colleagues who deserve some special recognition. First, I want to thank Simon Wenzel for laying the groundwork for this thesis. The fact that during my HiWi interview before I moved to Dortmund he extensively explained his research on distributed optimization to me while barely mentioning what the other PhD students were working on (with no ulterior motives of course) probably has had quite an impact on the topic of this thesis. Next, I would like to thank my scheduling partner-in-crime Christian Klanke. I am thankful that even though he was hired to work on distributed optimization he took up my scheduling projects and my initially intended PhD topic when I left Dortmund. Funny how that worked out. I am grateful for our many almost philosophical (and often pointless) discussions on scheduling, for some great shared papers and for his friendship over the years. On the Kaiserslautern-side of my PhD studies I want to thank Matheus Pedrosa for the great collaboration during our time together, but more importantly for his ongoing friendship and great support during one of the hardest periods of my life. Finally, I want to express my gratitude to my longest-serving colleague Nigora Gafur. Thank you for our fruitful collaboration and great discussions, both work-related and personal, for introducing me to yet another research topic (I had too many of those)

in robotics, for the nice papers we wrote together, for always supporting each other in our research and life and most importantly for your friendship.

I would also like to thank Silke Horn, Simon Wenzel and Vinit Hegiste for proofreading parts of the thesis. The thesis was "carried over the finish line" during my time at Gurobi Optimization. I want to especially thank my manager Kostja Siefen and my mentor Ronald van der Velden for supporting me in completing the thesis while at the same time getting me up to speed and teaching me a great deal of new things since I joined Gurobi.

Believe it or not, life mostly takes place outside of work and the university. Without great people in my life this thesis would not have been possible. I want to thank my family and friends who supported me throughout my studies. A big thank you also goes to my teammates at BBC Mehlingen for helping me take my mind off optimization once in a while. Δάσκαλε Γιάννη, δάσκαλε Νίκο, σας ευχαριστώ πολύ που με κάνατε να αγαπήσω τα μαθηματικά και τις θετικές επιστήμες. Χωρίς το Ανωτερότατο Εκπαιδευτικό Ίδρυμα Διάνυσμα σίγουρα δεν θα ήμουν εδώ.

Finally, my biggest gratitude goes to my wonderful wife Lisa. Thank you for always being by my side since the beginning of my studies. Thank you for your constant support and patience in all aspects of life. Thank you for applying the right amount of pressure for me to finally finish this thesis. Thank you for always reminding me about what is truly important in life. And most importantly, thank you for bringing our son Benjamin into our life. This PhD truly pales in comparison to everything you have given me. Without you all of this would never have been possible. I love you!

Oldenburg, 26.02.2024

List of Abbreviations

ADAL	Accelerated Distributed Augmented Lagrangian Method
ADMM	Alternating Direction Method of Multipliers
ALADIN	Augmented Lagrangian-based Alternating Direction Inexact Newton Method
BFGS	Broyden-Fletcher-Goldfarb-Shanno
BTM	Bundle Trust Method
COIN-OR	Computational Infrastructure for Operations Research
CPPS	Cyber-Physical Production System
D-BFGS	Decentral BFGS Method
DG	Duality Gap
DMPC	Distributed Model Predictive Control
DR	Demand Response
DSM	Demand-Side Management
FGM	Fast Gradient Method
FL	Federated Learning
IG	Integrality Gap
KKT	Karush-Kuhn-Tucker
LP	Linear Programming
MILP	Mixed-Integer Linear Programming
MINLP	Mixed-Integer Nonlinear Programming
MIP	Mixed-Integer Programming
MIQCP	Mixed-Integer Quadratically Constrained Programming
MIQCQP	Mixed-Integer Quadratically Constrained Quadratic Programming
MIQP	Mixed-Integer Quadratic Programming
MPC	Model Predictive Control
NAPS	Nearest Axis Point Separation
NCO	Necessary Conditions of Optimality
NLP	Nonlinear Programming
NMPC	Nonlinear Model Predictive Control
QAC	Quadratic Approximation Coordination
QADA	Quadratically Approximated Dual Ascent
QNDA	Quasi-Newton Dual Ascent
QP	Quadratic Programming

RSSD	Relaxation and Successive Distributed Decomposition
SAVLR	Surrogate Absolute Value Lagrangian Relaxation
SG	Subgradient
SLR	Surrogate Lagrangian Relaxation

List of Symbols

$\mathbf{A}$	Constraint matrix
$\mathbf{A}$	Dynamics matrix
$\mathbf{a}$	Parameter vector
a	Parameter
$\text{ave}(\cdot)$	Average
$\mathbf{B}$	Approximated Hessian
$\mathbf{B}$	Random matrix
$\mathbf{B}$	Control input matrix
$B(\cdot)$	Barrier function
$\mathbf{b}$	Constraints right-hand side
b	Bias variable
$\mathbf{C}$	Covariance matrix
$\mathbf{C}$	Random matrix
$\mathbf{c}$	Objective parameter
$\mathbf{D}$	Constraint matrix
$\mathbf{D}$	Diagonal matrix
$\mathbf{d}$	Constraints right-hand side
$\hat{d}(\cdot)$	Cutting plane model
d	Distance variable
$d_B(\cdot)$	Approximated dual function
$d_Q(\cdot)$	Approximated dual function
$d(\cdot)$	Dual function
$\mathbf{F}(\cdot)$	Nonlinear dynamics
$f(\cdot)$	Objective function
$\mathbf{G}$	Constraint matrix
$\mathbf{g}(\cdot)$	Inequality constraints
$\mathbf{g}(\cdot)$	Subgradient
$\mathbf{H}$	Objective matrix
$\mathbf{h}(\cdot)$	Equality constraints
$\mathbf{I}$	Identity matrix
i	Index
$J(\cdot)$	Objective function
j	Index
k	Index
$\mathbf{L}(\cdot)$	Linear mapping
$L(\cdot)$	Loss function
l	Index
$\mathbf{M}$	Vandermonde matrix
M	Big-M parameter
$\mathbf{m}$	Cluster centroid
$\mathbf{N}$	Random matrix
$\mathbf{p}$	Surrogate function parameters
$p(\cdot)$	Primal function
p	Constraints right-hand side
$\mathbf{Q}$	Quadratic approximation matrix
$\mathbf{q}$	Quadratic approximation vector
q_0	Quadratic approximation constant
$\mathbf{R}(\cdot)$	Resource function
$\mathbf{R}$	Resource matrix
$\mathbf{r}$	Resources
$r(\cdot)$	Approximated primal residual
$\mathbf{S}$	Random matrix
$\mathbf{s}$	Search direction
$\underline{s}$	Singular value bound
$\overline{s}$	Singular value bound
T	Time
t	Iteration index
$\mathbf{U}$	Unitary matrix
$\mathbf{u}$	Variable
$\mathbf{V}$	Unitary matrix
$\mathbf{v}$	Variable
$\mathbf{w}$	Normal vector
$\mathbf{w}$	Residual
w	Binary variable

$\mathbf{x}$	Variable
$\mathbf{y}$	Data point
$\mathbf{y}$	Subgradient change
$\mathbf{z}$	Variable
z	Label
z	Objective value
$\mathcal{A}$	Active set
$\mathcal{B}$	Bundle
$\mathcal{BC}$	Bundle cuts
$\mathcal{D}$	Data set
$\mathcal{E}$	Ellipsoid
$\mathcal{I}$	Index set
$\mathcal{J}$	Index set
$\mathcal{K}$	Index set
$\mathcal{L}(\cdot)$	Lagrange function
$\hat{\mathcal{L}}_\rho(\cdot)$	Augmented Lagrange function
$\mathcal{M}$	Constraint set
$\mathcal{N}$	Normal distribution
$\mathcal{U}$	Feasible set
$\mathcal{U}$	Uniform distribution
$\mathcal{X}$	Feasible set
$\mathcal{Y}$	Data set
$\mathcal{Y}$	Feasible set
$\arg(\cdot)$	Argument
$\text{ave}(\cdot)$	Average
inf	Infimum
lim	Limit
min / max	Objective sense
$\text{prox}(\cdot)$	Proximal operator
$\text{relint}(\cdot)$	Relative interior
α	Step size
$\boldsymbol{\beta}$	Parameter
β	Parameter
γ	Ellipsoid radius
$\Delta\boldsymbol{\lambda}$	NAPS radius
ϵ	Tolerance
$\boldsymbol{\zeta}$	Contraction parameter
θ	Parameter
$\boldsymbol{\Lambda}$	Set of dual variables
λ	Dual variables
μ	ADMM parameter
μ	Barrier parameter
μ	Mean
$\boldsymbol{\nu}$	Dual variables
$\boldsymbol{\xi}$	Subgradient
ρ	Regularization parameter
$\boldsymbol{\Sigma}$	Singular value matrix
σ	Singular value
τ	Age parameter
τ	ADMM parameter
$\phi(\cdot)$	Function
$\boldsymbol{\chi}$	Variable
$\hat{\boldsymbol{\Psi}}$	Observation
$\psi(\cdot)$	Approximated function
$\|\cdot\|$	Norm
∇	Gradient
∇^2	Hessian
∂	Subdifferential
$\emptyset$	Empty set
$\lceil\cdot\rceil$	Ceiling function
$\lfloor\cdot\rfloor$	Floor function

Kurzfassung

Diese Arbeit befasst sich mit der verteilten Optimierung von Restriktions-gekoppelten Systemen. Diese Problemklasse tritt häufig in Systemen auf, die aus mehreren einzelnen Teilsystemen bestehen, die durch geteilte begrenzte Ressourcen gekoppelt sind. Das Ziel ist es, jedes Teilsystem verteilt zu optimieren und gleichzeitig sicherzustellen, dass die systemübergreifenden Restriktionen eingehalten werden. Durch die Einführung dualer Variablen für die systemübergreifenden Restriktionen kann das systemweite Problem in seine einzelne Teilprobleme zerlegt werden. Diese resultierenden Teilprobleme können dann durch iterative Anpassung der dualen Variablen koordiniert werden. In dieser Arbeit werden zwei neue Algorithmen vorgestellt, die die Eigenschaften des dualen Optimierungsproblems ausnutzen. Beide Algorithmen bestimmen in jeder Iteration eine quadratische Surrogatfunktion der dualen Funktion, die zur Anpassung der dualen Variablen optimiert wird. Der Quadratically Approximated Dual Ascent (QADA) Algorithmus bestimmt die Surrogatfunktion durch die Lösung eines Regressionsproblems, während der Quasi-Newton Dual Ascent (QNDA) Algorithmus die Surrogatfunktion iterativ über ein Quasi-Newton-Schema aktualisiert. Beide Algorithmen verwenden Schnittebenen, um die Nichtdifferenzierbarkeit der dualen Funktion zu berücksichtigen. Die vorgeschlagenen Algorithmen werden mit Algorithmen aus der Literatur für eine große Anzahl verschiedener Benchmark-Probleme verglichen und zeigen in den meisten Fällen ein besseres Konvergenzverhalten. Zusätzlich zu generischen konvexen und gemischt-ganzzahligen Optimierungsproblemen wird die verteilte Optimierung basierend auf dualer Zerlegung auf verteilte modellprädiktive Regelungs- und verteilte K-Means Clustering-Probleme angewendet.

Abstract

This thesis deals with the distributed optimization of constraint-coupled systems. This problem class is often encountered in systems consisting of multiple individual subsystems, which are coupled through shared limited resources. The goal is to optimize each subsystem in a distributed manner while still ensuring that system-wide constraints are satisfied. By introducing dual variables for the system-wide constraints the system-wide problem can be decomposed into individual subproblems. These resulting subproblems can then be coordinated by iteratively adapting the dual variables. This thesis presents two new algorithms that exploit the properties of the dual optimization problem. Both algorithms compute a quadratic surrogate function of the dual function in each iteration, which is optimized to adapt the dual variables. The Quadratically Approximated Dual Ascent (QADA) algorithm computes the surrogate function by solving a regression problem, while the Quasi-Newton Dual Ascent (QNDA) algorithm updates the surrogate function iteratively via a quasi-Newton scheme. Both algorithms employ cutting planes to take the nonsmoothness of the dual function into account. The proposed algorithms are compared to algorithms from the literature on a large number of different benchmark problems, showing superior performance in most cases. In addition to general convex and mixed-integer optimization problems, dual decomposition-based distributed optimization is applied to distributed model predictive control and distributed K-means clustering problems.

Publications

Parts of this thesis have previously appeared in peer-reviewed publications and pre-prints. These publications are indicated by references throughout the thesis and they are listed below with references to the respective chapters. A list of further publications by the author are listed in Appendix D.

Published

Peer-reviewed Journals

[YWW+23]: **V. Yfantis**, S. Wenzel. A. Wagner, M. Ruskowski, S. Engell, 2023. **Hierarchical distributed optimization of constraint-coupled convex and mixed-integer programs using approximations of the dual function**. *EURO Journal on Computational Optimization*. DOI: https://doi.org/10.1016/j.ejco.2023.100058.

Conference Proceedings

[YR22]: **V. Yfantis**, M. Ruskowski, 2022. **A Hierarchical Dual Decomposition-based Distributed Optimization Algorithm combining Quasi-Newton Steps and Bundle Methods**. *30th Mediterranean Conference on Control and Automation (MED)*. DOI: https://doi.org/10.1109/MED54222.2022.9837219.

[YGW+22]: **V. Yfantis**, N. Gafur, A. Wagner M. Ruskowski, 2022. **Hierarchical Distributed Model Predictive Control based on Dual Decomposition and Quadratic Approximation**. *30th Mediterranean Conference on Control and Automation (MED)*. DOI: https://doi.org/10.1109/MED54222.2022.9837180.

[YMB+22]: **V. Yfantis**, W. Motsch, N. Bach, A. Wagner M. Ruskowski, 2022. **Optimal Load Control and Scheduling through Distributed Mixed-integer Linear Programming**. *30th Mediterranean Conference on Control and Automation (MED)*. DOI: https://doi.org/10.1109/MED54222.2022.9837224.

Preprints

[YWR23]: **V. Yfantis**, A. Wagner M. Ruskowski, 2023. **Federated K-Means Clustering via Dual Decomposition-based Distributed Optimization**. *arXiv Preprint* https://arxiv.org/abs/2307.13267

Chapter	**Statement**	**Reference**
1	Parts of the description copied from	[YWW+23]
2.2.2	Partly published in	[YWW+23]
3	Partly published in	[YWW+23]
4	Partly published in	[YWW+23]
4.1.5	First version of the algorithm published in	[YGW+22]
4.2	First version of the algorithm published in	[YR22]
6	Partly published in	[YWW+23]
6.1.2	Adapted from	[YR22]
7	Partly submitted to	[YWR24]
7.2.3	Adapted from	[YGW+22]
8	Partly published in	[YWR23]
9.2.1	Preliminary results for DSM published in	[YMB+22]

1 Introduction

The solution of optimization problems, i.e., making optimal decisions with respect to a specified objective while satisfying a number of constraints, has been of interest for many years [DPW09]. These problems are dealt with in the area of operations research, the history of which can be traced back to George B. Dantzig and the development of the simplex algorithm for linear programming [Dan51], even though earlier work on the topic exists, e.g. by Kantorovich [Kan39]. The terms "operations" and "programming" in the context of optimization refer to their origins within military applications, on which Dantzig was working during World War II [Dan90]. It did not take long for commercial optimization software to be developed [Bix12]. Through the ever-increasing hardware capabilities as well as due to continuous algorithmic developments, the performance of state-of-the-art optimization solvers has been drastically increasing in recent years [AW13, KBP+22]. Fig. 1.1 illustrates the performance increase of the commercial optimization solver Gurobi [Gur23] from version to version for mixed-integer linear programming (MILP) problems[1] (v1.1/2009 – v10.0/2022). The figure also shows that more and more problems can be solved within acceptable computation times. In addition to commercial solvers, many open-source projects, like the Computational Infrastructure for Operations Research (COIN-OR) [Lou03], provide state-of-the-art optimization algorithms to the scientific community. However, some problems might still be unsolvable, either from a computational or a structural point of view.

Many industrial applications of optimization require the solution of a system-wide problem over a network of agents [NL18]. Solving a system-wide optimization problem in a centralized fashion in such a setting can become computationally intractable if a large number of agents are involved. Furthermore, in production systems there has been a trend towards increased modularity and autonomy of subsystems in recent years [RHH+20]. This gives rise to distributed decision structures where the involved subsystems have a certain autonomy and pursue individual goals while only having access to local information [YYW+19]. In these cases the exchange of information between the subsystems or between the subsystems and a coordinating unit is often restricted as the subsystems do not want to share private information, e.g., their objective functions, local constraints, production parame-

[1] The benchmark set includes 4535 models that can be solved by at least one version with a 10000 second time limit. Unsolved models are assigned a 100000 seconds solution time. Speed-up is only measured for models that take longer than 100 seconds to solve. All computations were performed on Intel Xeon CPU EE3-1240 v5 @ 3.50Ghz, 4 cores, 32 GB RAM. Source: `gurobi.com` (last visited 24.03.2023)

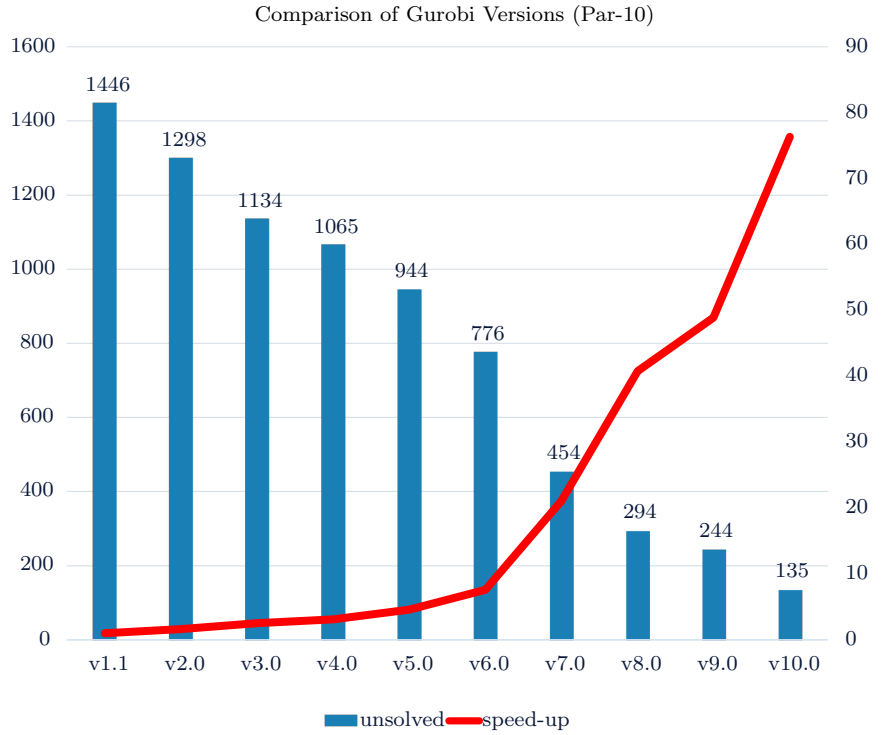


Figure 1.1: Version-to-version performance evolution of the commercial optimization solver Gurobi for MILP benchmark problems. Source: `gurobi.com` (last visited 24.03.2023)

ters, etc. [HTP17, LZ18]. This is often the case in industrial complexes where production systems are coupled through interconnected networks of materials and energy [WPS+16]. The involved subsystems may not be willing or able to exchange the information required for the centralized solution of a system-wide optimization problem, e.g., because they belong to different business units or different companies. Another area necessitating the solution of large-scale optimization problems is machine learning [SCZ+19, GGN21]. In addition to the size of the underlying optimization problems, data sovereignty plays an important role in many machine learning applications [LFT+20]. Training data may be distributed over multiple nodes of a network. Sharing this data between different nodes or between the nodes and a coordinator can be prohibitive due to bandwidth limitations or due to privacy concerns [KMR15].

Distributed optimization methods offer a way to circumvent the aforementioned issues by splitting the aggregated optimization problem into smaller subproblems[2], solving the subproblems locally, and coordinating these solutions by suitable mechanisms such that the coordinated solutions for the subproblems converge to the system-wide solution and system-wide constraints are met. The design of distributed optimization algorithms involves the choice of the decomposition method and of the synchronization mechanism and depends on the possible communication of data between subproblems and the coordinating instance.

[2]The terms subsystem and subproblem are used interchangeably throughout this thesis. A subproblem describes the optimization problem associated with a subsystem.

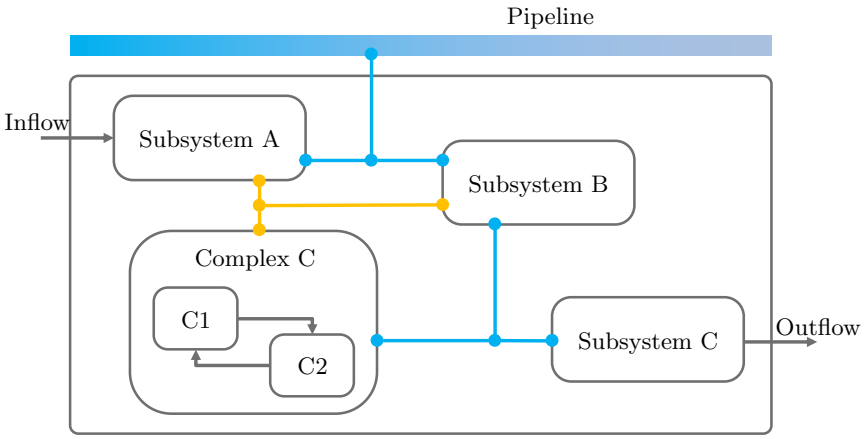


Figure 1.2: Decentralized MPC.

Figure 1.3: Example of an integrated production network (adapted from [WRE20, Wen20]).

Maxeiner [Max21] provides an overview of applications for distributed optimization, referring to it as cross-entity optimization, including the coordination of cars [JLW+15, ZLL+16, GMM14], trains [SGK17, YTY+17] or unmanned aerial vehicles (UAVs) [LMZ+18, PLF+18], distributed optimization of energy systems, both in an industrial and residential setting [NFN16, RNF23, SGC+22, YMB+22, JT16, ZGG13, ZGG12, ALT19, AD14, WYW13, JI13, NSH15, BCC+15] and digital infrastructure [ZLG+15]. One of the main possible application areas for distributed optimization is the coordination of production systems. It can be applied in a natural way to production plants that are coupled through shared limited resources. An example of an integrated production network, where different subsystems are coupled through shared resources is depicted in Fig. 1.3. A common shared resource is the available electricity in the case of demand-side management (DSM). General reviews for industrial applications of DSM are provided by Wang et al. [WZD+15], Shoreh et al. [SSS+16] and Siano [Sia14]. Motsch et al. present an architecture for the application of price-based DSM for autonomous cyber-physical production modules (CPPMs) [MDS+20]. Yfantis et al. employ dual decomposition-based distributed optimization for the integrated load control and scheduling of flexible manufacturing plants with shared energy [YMB+22]. Motsch et al. use a game theoretic approach to distributedly adapt production schedules based on energetic considerations [MYW+23]. A general overview of distributed optimization in the context of energy systems is presented by Yang et al. [YYW+19]. Shared energy does also play an important role in the distributed optimization of smart buildings (cf. [ZTL+11, ZGG13, PAL14, ESK+22]).

Other frequently shared resources are gases or steam. Che et al. employ dynamic pricing of gas transmission capacities for optimal resource allocation [CWS18]. Gunnerud and Foss present the optimal operation of an oil field [GF10]. He et al. simultaneously schedule electricity and natural gas systems [HWL+17]. Mouret et al. apply distributed optimization for the integration of refinery planning and crude-oil scheduling [MGP11]. Martí et

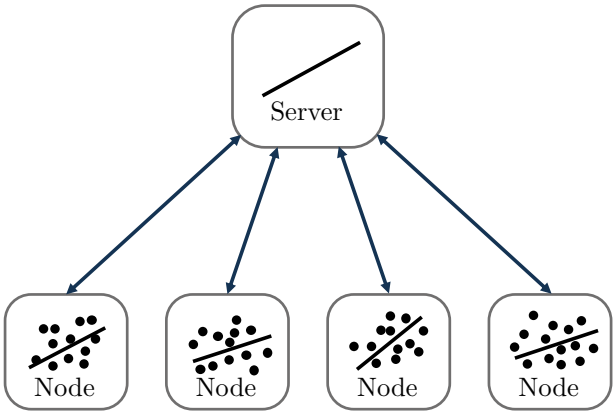


Figure 1.4: Example of a distributed machine learning problem with nodes training models on private data while being coordinated by a server training a global model.

al. coordinate decentral controllers within an oxygen distribution network [MSN+13]. Van den Heever and Grossmann integrate the production planning and reactive scheduling of a hydrogen supply network via a decomposition approach [vdHG03]. Wenzel et al. also employ price-based coordination to manage the steam among three large petrochemical plants [WPS+16].

Naturally, distributed optimization is not only encountered in industrial applications. A major field of research in recent years has been federated learning, which describes the distributed solution of machine learning problems [LFT+20]. Machine learning models are trained on data, which is often confidential and cannot be shared if it is stored in different nodes of a larger network. Fig. 1.4 illustrates a setting in which a global model has to be trained without access to locally stored data. Federated learning aims at training a model over the entire dataset by coordinating the training of local models in each node. Hegiste et al. demonstrate the application of federated learning in the context of manufacturing [HLR22]. Other applications include healthcare [XGS+21] or user behavior prediction in social networks [KMR15]. Several applications are reviewed by Li et al. [LFT+20].

Even though distributed optimization offers a multitude of benefits, its application still faces many challenges. Wenzel [Wen20] analyzes the coordination landscape in the process industries and classifies the challenges into organizational frameworks, awareness and acceptance, legal aspects, technical limitations, information technology infrastructure, presence of uncertainties, and availability of suitable models and algorithms. While all of these aspects are equally important for the successful application of distributed optimization, this thesis focuses on the last point, i.e., the availability of suitable algorithms.

The main issue arising from the decomposition of an optimization problem into several smaller subproblems is the resulting slow rate of convergence toward the system-wide optimum. While the subproblems are usually much easier to solve than the original problem, many communication rounds might be required until an optimal or even just feasible solution can be found for the original problem. The rate of convergence can generally be

accelerated by increasing the amount of information shared by the subproblems. However, if certain information is confidential, like in many of the applications mentioned above, this might not be a viable option. Then the only way to improve the rate of convergence is to employ distributed optimization algorithms that take advantage of the available information as efficiently as possible.

This thesis focuses on dual decomposition where system-wide constraints that couple the subproblems are relaxed by introducing additional variables into the subproblems, solving the modified subproblems in a distributed fashion, and coordinating the solution process by iteratively adapting the additional variables. This makes it possible to realize a high degree of privacy, as no or little sensitive information has to be shared between the subproblems. The coordination of the subproblems can be performed by a central coordination algorithm that exchanges information with the subproblems (hierarchically), by directly exchanging information between the subproblems (networked optimization), or by solving the subproblems in a completely decentralized manner (non-cooperative games) [YJ10]. In this work, hierarchical algorithms are considered that coordinate the solutions of the subproblems by iteratively adapting and broadcasting the additional variables that result from the relaxation of the system-wide coupling constraints. On the one hand, the hierarchical structure ensures that no sensitive information has to be shared between subproblems, as communication is only established between the coordinator and the subproblems. On the other hand, the presence of the central coordinator enables to converge to the system-wide optimum of the aggregated problem which is usually not possible through a fully decentralized approach if system-wide coupling constraints must be satisfied.

The type and the amount of information shared between the subproblems and the coordinator influence the efficiency of distributed optimization algorithms. The exchanged information may include the contributions of the subsystems to the system-wide constraint functions [ME20] or the residual of the system-wide constraints [WRE20], the optimal objective function values of the subsystems in each iteration, gradients of the subsystems' objective functions and constraints, and the Hessians of the Lagrange functions of the subsystems [HFD16]. The first two choices lead to a high degree of privacy of the subsystems whereas algorithms that exchange the full information on the subsystem solutions are motivated by reducing the memory demand or computation time compared to the system-wide solution rather than assuring privacy. In the iterations of a hierarchical distributed optimization algorithm all subproblems are solved in parallel and return information to the coordinator, i.e., they are optimized synchronously. In contrast, asynchronous algorithms only require the solution of a subset of the subproblems in each iteration, leading to a trade-off between collected information and computational efficiency [BLY+15]). Usually, fewer iterations are required if all subproblems are solved in each iteration.

As mentioned before, dual decomposition-based algorithms generally exhibit a slow rate of convergence. This issue was addressed by, e.g., Maxeiner and Engell [ME20] and Wenzel et al. [WRE20] where efficient use of information from previous iterations was made. In this thesis, a new algorithm is proposed that uses some of the elements of the quadratic approximation coordination (QAC) algorithm proposed by Wenzel et al. [WRE20, Wen20]. In contrast to the QAC algorithm, the new algorithm approximates the dual function of the system-wide optimization problem by a quadratic function by solving a regression problem in each iteration. This requires to exchange the values of the Lagrange functions of the subproblems at each iteration but still maintains the privacy of the local constraints and of the contributions to the system-wide constraints. As will be shown, this improves the rate of convergence for convex problems with real-valued decision variables and in particular leads to an efficient distributed solution of integer programs. A regression-based approximation of the dual function requires an initialization phase where the necessary number of initial data points are collected. This is avoided by a second presented algorithm that approximates the dual function based on quasi-Newton updates.

The remainder of this thesis is organized as follows: Chapter 2 introduces general concepts of optimization as well as some important optimization problem classes that will be encountered throughout the thesis. Subsequently, distributed optimization is extensively discussed, mainly focusing on dual decomposition while still providing overviews of alternative approaches. Several distributed optimization algorithms which will serve as references for the newly proposed approaches are discussed in Chapter 3. The discussion focuses on algorithms that employ a hierarchical coordination structure where only first-order information is shared between the subproblems and the coordinator. Other related algorithms which employ different communication structures, exchanged information, and synchronization strategies, are also discussed briefly. Chapter 4 discusses algorithms that update the dual variables based on an optimization of a smooth surrogate function. This includes the QAC algorithm introduced by Wenzel et al. [WRE20] as well as the newly proposed algorithms which compute a quadratic surrogate function of the dual function. The convergence properties of these algorithms are discussed at the end of the chapter for different classes of problems in a semi-formal manner, based on known results for the same type of problems and similar algorithms.

A large part of the thesis is devoted to the evaluation of the performance of the proposed algorithms in comparison to known approaches for a large set of benchmark problems for different problem classes and applications. Chapter 5 demonstrates the application of distributed optimization with an illustrative example of the coordination of a small fictitious resource network. Chapter 6 provides numerical comparisons for a large set of general benchmark problems within different problem classes. Chapters 7 and 8 show how dual decomposition-based distributed optimization can be applied to distributed model predictive control and distributed machine learning respectively. The thesis is concluded with

an extensive discussion of potential future applications and algorithmic improvements, as well as with some practical considerations in Chapter 9.

Notation

Boldfaced upper-case letters are used to denote matrices ($\mathbf{X}$) and boldfaced lower-case letters to denote vectors ($\mathbf{x}$). The notation $[\mathbf{x}]_l$ denotes the l-th element of a vector $\mathbf{x}$. Similarly, $[\mathbf{X}]_{l,j}$ denotes the (l,j)-th element of a matrix $\mathbf{X}$. The vector containing only ones is denoted by $\mathbf{1}$ while the vector containing only zeros is denoted by $\mathbf{0}$. $\mathbf{I}$ denotes the identity matrix of appropriate dimensions. Vector inequalities $\mathbf{x} \bowtie \mathbf{y}$ with $\bowtie \in \{<, >, \leq, \geq\}$ are interpreted component-wise. The iteration index of the distributed optimization algorithms is denoted by t. The value of a variable $\mathbf{x}$ in iteration t is denoted by $\mathbf{x}^{(t)}$ while $\mathbf{x}_i$ indicates that a variable belongs to subproblem i. The Euclidean norm is denoted by $\|\mathbf{x}\|_2 = \sqrt{\mathbf{x}^T\mathbf{x}}$ while $\|\mathbf{X}\|_F = \sqrt{\sum_{l=1}^{n}\sum_{k=1}^{n}|[\mathbf{X}]_{l,k}|^2}$ denotes the Frobenius norm of a matrix. $S\mathbb{R}^{n\times n}$ denotes symmetric matrices with n rows/columns. The notation $\mathbf{x}^*$ indicates the optimum of an optimization problem. The domain of a function $f(\mathbf{x})$ is denoted by dom f.

2 Optimization

This thesis deals with the distributed optimization of constraint-coupled problems via dual decomposition. To this end, this chapter presents some general concepts of optimization, namely, the necessary conditions of optimality as well as the notion of convexity, which play an essential role in dual decomposition-based distributed optimization. Throughout this thesis different classes of optimization problems are solved, both as part of the distributed optimization algorithms and as subproblems of the constraint-coupled system-wide problems. Therefore, the solution of major optimization problem classes encountered throughout this thesis, namely, linear programming (LP), quadratic programming (QP), nonlinear programming (NLP), and mixed-integer programming (MIP) are discussed. Finally, dual decomposition-based distributed optimization is introduced and subsequently compared to other distributed optimization methods, focusing on the structure of problems these methods can be applied to.

2.1 Fundamentals of optimization

The field of mathematical optimization deals with the solution of optimization problems.

Definition 1: Constrained optimization problem

A problem of the form

$$\min_{\mathbf{x}\in\mathbb{K}^{n_\mathbf{x}}} f(\mathbf{x}), \tag{2.1a}$$

$$\text{s.t. } \mathbf{g}(\mathbf{x}) \leq \mathbf{0}, \tag{2.1b}$$

$$\mathbf{h}(\mathbf{x}) = \mathbf{0}, \tag{2.1c}$$

with an objective function $f\colon \mathbb{K}^{n_\mathbf{x}} \to \mathbb{R}$, component-wise inequality constraints $\mathbf{g}\colon \mathbb{K}^{n_\mathbf{x}} \to \mathbb{R}^{n_\mathbf{g}}$ and component-wise equality constraints $\mathbf{h}\colon \mathbb{K}^{n_\mathbf{x}} \to \mathbb{R}^{n_\mathbf{h}}$ is called a constrained optimization problem. The set

$$\mathcal{X} = \{\mathbf{x} \in \mathbb{K}^{n_\mathbf{x}} |\ \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \wedge \mathbf{h}(\mathbf{x}) = \mathbf{0}\} \tag{2.2}$$

denotes the feasible set of problem (2.1).

Throughout this thesis both $\mathbb{K} = \mathbb{R}$ and $\mathbb{K} = \mathbb{R} \times \mathbb{Z}$ are considered, i.e., continuous and mixed-integer problems. The goal of the optimization problem (2.1) is to find the values of the decision variables $\mathbf{x}^*$ which minimize the objective function $f(\mathbf{x})$ while satisfying the constraints (2.1b) and (2.1c).

Definition 2: Optimum

The vector $\mathbf{x}^* \in \hat{\mathcal{X}} \subset \mathbb{R}^{n_\mathbf{x}}$ is a local optimum (minimum) of the optimization problem (2.1) in the region $\hat{\mathcal{X}} \subset \mathcal{X}$, if

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

If condition (2.3) holds for all $\mathbf{x} \in \mathcal{X}$, i.e., for the entire feasible set, then $\mathbf{x}^*$ is a global optimum (minimum) of problem (2.1).

2.1.1 Conditions of optimality

A locally optimal point $\mathbf{x}^*$ is also called a solution to the optimization problem (2.1). A necessary condition of optimality (NCO) for unconstrained optimization problems with real-valued decision variables ($\mathcal{X} = \mathbb{R}^{n_\mathbf{x}}$) is that the gradient of the objective function vanishes at the optimum, i.e.,

$$\mathbf{x}^* \text{ is a local optimum} \Leftrightarrow \nabla f(\mathbf{x}^*) = 0.$$

The same does not apply to constrained optimization problems like (2.1). To formulate the NCO the Lagrange function must be defined.

Definition 3: Lagrange function [BV04]

The function

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) := f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x}) + \boldsymbol{\nu}^T \mathbf{h}(\mathbf{x}), \tag{2.4}$$

with $\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}}$, $\boldsymbol{\lambda} \in \mathbb{R}^{n_\mathbf{g}}$ and $\boldsymbol{\nu} \in \mathbb{R}^{n_\mathbf{h}}$ is called the Lagrange function or Lagrangian of the optimization problem (2.1). The variables $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$ are called Lagrange multipliers or dual variables. In contrast, the variables $\mathbf{x}$ are called primal variables.

The Lagrange function and the dual variables form the basis for dual decomposition-based distributed optimization and will be discussed more thoroughly in Sec. 2.2.2. Furthermore, they can be used to formulate the NCO for constrained optimization problems, the so-called Karush-Kuhn-Tucker (KKT) conditions [Kar39, KT51].

Definition 4: Karush-Kuhn-Tucker conditions [BV04]

If a pair of primal and dual variables $\mathbf{x}^*$ and $(\boldsymbol{\lambda}^*, \boldsymbol{\nu}^*)$ are (locally) optimal, they have to satisfy the following conditions:

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\nu}^*) = \mathbf{0}, \qquad \text{(Stationarity)} \tag{2.5a}$$

$$\mathbf{g}(\mathbf{x}^*) \leq \mathbf{0},\ \mathbf{h}(\mathbf{x}^*) = \mathbf{0}, \qquad \text{(Primal feasibility)} \tag{2.5b}$$

$$\boldsymbol{\lambda}^* \geq \mathbf{0}, \qquad \text{(Dual feasibility)} \tag{2.5c}$$

$$[\boldsymbol{\lambda}^*]_l \cdot [\mathbf{g}(\mathbf{x}^*)]_l = 0,\ l = 1, \ldots, n_{\mathbf{g}}. \qquad \text{(Complementarity)} \tag{2.5d}$$

Note that the KKT conditions are necessary conditions, not sufficient ones. A sufficient condition is the positive definiteness of the Hessian of the Lagrange function, i.e.,

$$\nabla^2_{\mathbf{xx}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\nu}^*) > 0. \tag{2.6}$$

The KKT conditions are necessary if some additional constraint regularity conditions, called constraint qualifications, are satisfied [NW06]. Many optimization algorithms aim at finding a solution that satisfies the KKT conditions.

2.1.2 Convexity

The problem formulation (2.1) corresponds to a generic nonlinear programming (NLP) problem (cf. Sec. 2.1.5). Depending on the properties of the objective function and constraints different subclasses of optimization problems can be defined. One of the most important classes is that of convex optimization problems, which describe the minimization of a convex function over a convex feasible set.

Definition 5: Convex set [BV04]

A set $\mathcal{C}$ is convex if the line segment between any two points in $\mathcal{C}$ also lies in $\mathcal{C}$, i.e.,

$$\theta \mathbf{x}_1 + (1 - \theta)\mathbf{x}_2 \in \mathcal{C},\ \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{C} \subset \mathbb{R}^{n_{\mathbf{x}}}, \theta \in [0, 1]. \tag{2.7}$$

The concept of convexity of a set is illustrated in Fig. 2.1. The definition of a convex function is similar.

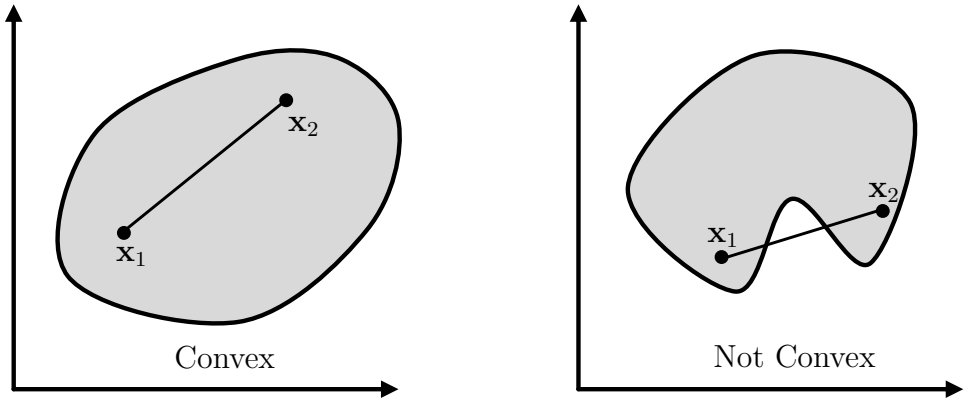


Figure 2.1: Illustration of convex sets. In the left case, the line segment between any two points of the set also lies in the set. This does not hold for the right case.

Definition 6: Convex function [BV04]

A function $f\colon \text{dom } f \to \mathbb{R}$ is convex, if dom f is a convex set and

$$f(\theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2) \leq \theta f(\mathbf{x}_1) + (1-\theta)f(\mathbf{x}_2),\ \forall \mathbf{x}_1, \mathbf{x}_2 \in \text{dom } f, \theta \in [0,1] \tag{2.8}$$

holds.

Geometrically condition (2.8) implies that all points belonging to the line segment connecting any two points on the graph of the function $f(\mathbf{x})$ always lie above the graph of $f(\mathbf{x})$. This is illustrated in Fig. 2.2.
Using the Definitions 5 and 6 a convex optimization problem can be defined.

Definition 7: Convex optimization problem

Problem (2.1) is a convex optimization problem if the objective function $f(\mathbf{x})$ and the inequality constraints $[\mathbf{g}(\mathbf{x})]_l,\ l = 1, \ldots, n_\mathbf{g}$, are convex and the equality constraints $\mathbf{h}(\mathbf{x})$ are affine, i.e., if $f(\mathbf{x})$ is a convex function and $\mathcal{X}$ is a convex set.

The most important property of convex optimization problems is that if a local optimum $\mathbf{x}^*$ is found, it is also a global optimum, even though it might not be a unique one [BV04, NW06].

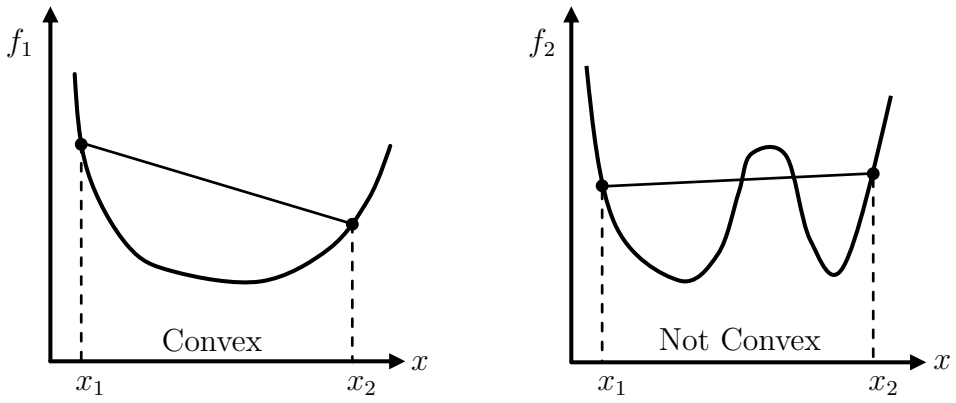


Figure 2.2: Illustration of convex functions. In the left case, the line segment between any two points on the graph lies above the graph. This does not hold for the right case.

2.1.3 Linear programming

One of the earliest studied problem classes of constrained optimization is linear programming (LP).

Definition 8: Linear program

An optimization problem of the form

$$\max_{\mathbf{x}\in\mathbb{R}^{n_\mathbf{x}}} \mathbf{c}^T\mathbf{x}, \tag{2.9a}$$

$$\text{s.t. } \mathbf{A}\mathbf{x} = \mathbf{b}, \tag{2.9b}$$

$$\mathbf{x} \geq \mathbf{0} \tag{2.9c}$$

with $\mathbf{c} \in \mathbb{R}^{n_\mathbf{x}}$, $\mathbf{A} \in \mathbb{R}^{n_\mathbf{b} \times n_\mathbf{x}}$ and $\mathbf{b} \in \mathbb{R}^{n_\mathbf{b}}$ is called a linear program.

The most important properties of LPs are that they are always convex and that their optimal solution (if one exists) lies at a vertex of the feasible set $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}} | \ \mathbf{A}\mathbf{x} \leq \mathbf{b} \wedge \mathbf{x} \geq \mathbf{0}\}$. This is exploited by the simplex algorithm, which moves from vertex to vertex in the direction of the biggest objective improvement until the optimal vertex is found [Dan51].

Fig. 2.3 depicts the constraints and the feasible set $\mathcal{X}$ for the following linear program:

$$\max_{x_1,x_2} x_1 + 2x_2, \tag{2.10a}$$

$$\text{s.t. } 2x_1 + 3x_2 \leq 12, \tag{2.10b}$$

$$x_1 + x_2 \leq 5, \tag{2.10c}$$

$$0 \leq x_1 \leq 4 \tag{2.10d}$$

$$0 \leq x_2 \leq 3. \tag{2.10e}$$

The feasible set in Fig. 2.3 is convex and the optimal solution lies on one of its vertices.

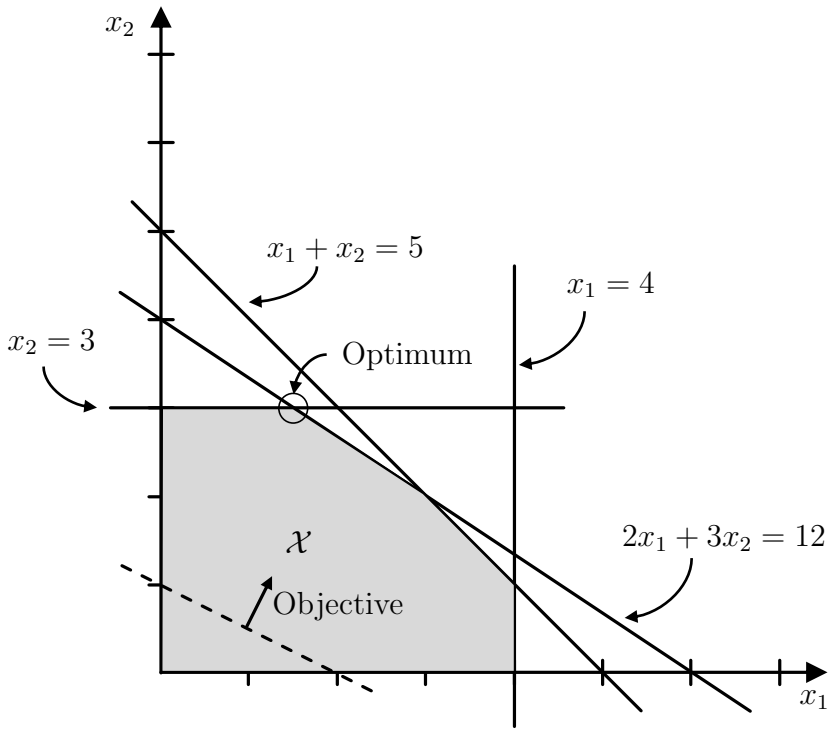


Figure 2.3: Graphic representation of the LP (2.10).

At a vertex, some of the inequality constraints are active while others are inactive.

Definition 9: Active constraints

If $\tilde{\mathbf{x}}$ is a feasible solution for the general optimization problem (2.1), i.e., $\mathbf{g}(\tilde{\mathbf{x}}) \leq \mathbf{0}$ and $\mathbf{h}(\tilde{\mathbf{x}}) = \mathbf{0}$, then the constraints

$$\mathbf{h}(\tilde{\mathbf{x}}) = \mathbf{0} \text{ and } [\mathbf{g}(\tilde{\mathbf{x}})]_l = 0,\ l \in \mathcal{A} \subseteq \{1, \ldots, n_{\mathbf{g}}\},$$

for which the equality holds, are called active constraints. The set of all active constraints is called the active set.

The simplex algorithm for linear programming is a so-called active set method, as it changes the active set of constraints in each iteration until the optimum is found.
An alternative class of algorithms are interior point or barrier methods, which start from a point in the interior of the feasible set and move within the set on a central path until a solution is found. The earliest interior point method for LPs is Karmakar's projective algorithm [Kar84]. The idea of logarithmic barrier-based interior point methods is discussed for quadratic programs in the next section.

2.1.4 Quadratic programming

Another important class of convex optimization problems are quadratic programs.

Definition 10: Quadratic program

An optimization problem of the form

$$\min_{\mathbf{x}\in\mathbb{R}^{n_\mathbf{x}}} \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{c}^T\mathbf{x}, \tag{2.11a}$$

$$\text{s.t. } \mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{2.11b}$$

with a symmetric positive (semi-)definite matrix $\mathbf{H} \in S\mathbb{R}^{n_\mathbf{x}\times n_\mathbf{x}}$, $\mathbf{c} \in \mathbb{R}^{n_\mathbf{x}}$, $\mathbf{A} \in \mathbb{R}^{n_\mathbf{b}\times n_\mathbf{x}}$ and $\mathbf{b} \in \mathbb{R}^{n_\mathbf{b}}$ is called a quadratic program (QP).

Note that equality constraints have been omitted in problem (2.11) since they can easily be transformed into inequality constraints:

$$\mathbf{h}(\mathbf{x}) = \mathbf{0} \equiv \mathbf{h}(\mathbf{x}) \geq \mathbf{0} \wedge \mathbf{h}(\mathbf{x}) \leq \mathbf{0}.$$

While active set methods can be used to solve QPs, interior point algorithms have been shown to be very efficient. Let $\mathbf{a}_l$ be the l-th row vector of the matrix $\mathbf{A}$, i.e.,

$$\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_{n_\mathbf{b}}]^T.$$

Then the QP (2.11) can be reformulated as

$$\min_{\mathbf{x}\in\mathbb{R}^{n_\mathbf{x}}} \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{c}^T\mathbf{x}, \tag{2.12a}$$

$$\text{s.t. } [\mathbf{b}]_l - \mathbf{a}_l^T\mathbf{x} \geq 0,\ l = 1, \ldots, n_\mathbf{b}. \tag{2.12b}$$

The inequality-constrained problem (2.12) can now be transformed into an unconstrained one by defining a barrier function

$$B(\mathbf{x}, \mu) \coloneqq \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{c}^T\mathbf{x} - \mu\sum_{l=1}^{n_\mathbf{b}} \log([\mathbf{b}]_l - \mathbf{a}_l^T\mathbf{x}), \tag{2.13}$$

with a barrier parameter $\mu > 0$. Computing the gradient of the barrier function yields

$$\nabla_\mathbf{x} B(\mathbf{x}, \mu) = \mathbf{H}\mathbf{x} + \mathbf{c} + \mu\sum_{l=1}^{n_\mathbf{b}} \frac{1}{[\mathbf{b}]_l - \mathbf{a}_l^T\mathbf{x}}\mathbf{a}_l. \tag{2.14}$$

Then, dual variables $\boldsymbol{\nu} \geq \mathbf{0}$ similar to the Lagrange multipliers with

$$[\boldsymbol{\nu}]_l([\mathbf{b}]_l - \mathbf{a}_l^T\mathbf{x}) = \mu,\ l = 1, \ldots, n_\mathbf{b} \tag{2.15}$$

can be introduced. Using the dual variables the optimality condition for the barrier function can be rewritten as

$$\mathbf{H}\mathbf{x} + \mathbf{c} + \mathbf{A}^T\boldsymbol{\nu} = \mathbf{0}. \tag{2.16}$$

Eq. (2.15) and eq. (2.16) form a system of equations that can be solved, e.g., using Newton's method. Each iteration of the optimization algorithm gives a solution $(\tilde{\mathbf{x}}^*, \tilde{\boldsymbol{\nu}}^*)$, where $\tilde{\boldsymbol{x}}^*$ is a feasible solution of the QP (2.11). While the barrier parameter μ is decreased the found solutions converge towards the optimum of the QP. The interested reader is referred to the textbooks by Boyd and Vandenberghe [BV04] or Nocedal and Wright [NW06] for further details on interior point methods.

2.1.5 Nonlinear programming

If the objective function and/or constraints of the optimization problem (2.1) are general nonlinear functions, it is referred to as a nonlinear programming (NLP) problem. NLPs can generally be solved by using interior point methods or active set methods. Throughout this thesis, the solver IPOPT (Interior-Point Optimizer) [WB06] is used for solving general nonlinear programs. It is based on a primal-dual barrier method, similar to the one discussed in the previous section. In its general formulation, problems of the form

$$\min_{\mathbf{x}} f(\mathbf{x}), \tag{2.17a}$$

$$\text{s.t. } \mathbf{h}(\mathbf{x}) = \mathbf{0}, \tag{2.17b}$$

$$\mathbf{x} \geq \mathbf{0} \tag{2.17c}$$

are solved. Note that general inequality constraints can be handled by introducing positive slack variables $\mathbf{y}$

$$\mathbf{g}(\mathbf{x}) \leq \mathbf{0} \equiv \mathbf{g}(\mathbf{x}) + \mathbf{y} = \mathbf{0} \wedge \mathbf{y} \geq \mathbf{0}.$$

By defining the barrier function

$$B(\mathbf{x}, \mu) \coloneqq f(\mathbf{x}) - \mu \sum_{l=1}^{n_\mathbf{x}} \log([\mathbf{x}]_l), \tag{2.18}$$

the algorithm computes (approximate) solutions for a sequence of barrier problems

$$\min_{\mathbf{x}} B(\mathbf{x}, \mu), \tag{2.19a}$$

$$\text{s.t. } \mathbf{h}(\mathbf{x}) = \mathbf{0}, \tag{2.19b}$$

for decreasing values of the barrier parameter μ. By defining the Lagrange function of problem (2.19),

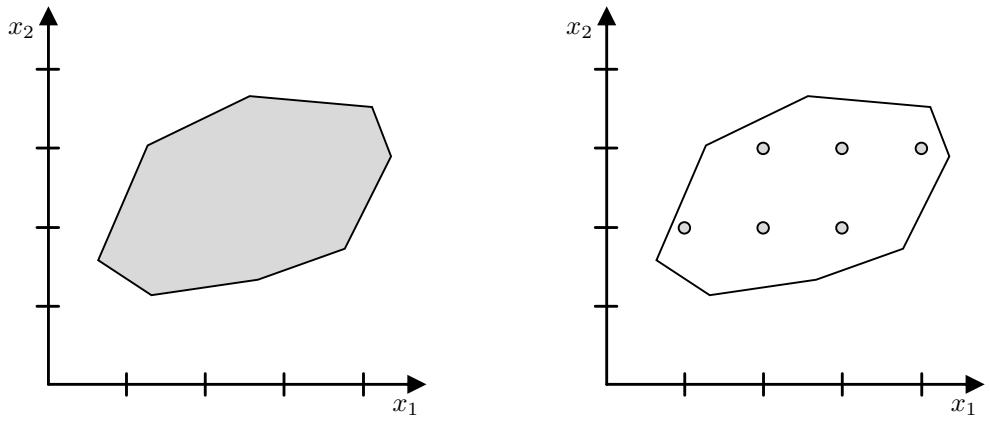
$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{h}(\mathbf{x}) - \mu \sum_{l=1}^{n_\mathbf{x}} \log([\mathbf{x}]_l), \tag{2.20}$$

and the dual variables $\boldsymbol{\nu}$ the KKT conditions can be formulated

$$\nabla f(\mathbf{x}) + \nabla \mathbf{h}(\mathbf{x}) \boldsymbol{\lambda} - \boldsymbol{\nu} = \mathbf{0}, \tag{2.21a}$$

$$\mathbf{h}(\mathbf{x}) = \mathbf{0}, \tag{2.21b}$$

$$[\boldsymbol{\nu}]_l \cdot [\mathbf{x}]_l = \mu, \; l = 1, \ldots, n_\mathbf{x}. \tag{2.21c}$$



(a) Continuous feasible set. (b) Integer feasible set.

Figure 2.4: Illustration of the feasible sets for a continuous optimization problem and its integer counterpart.

The system of equations (2.21) can then be solved by using Newton's method. More details on the implementation of IPOPT can be found in [WB06].

The main characteristic of NLPs is whether they are convex or not. As discussed previously, if a solution satisfying the KKT conditions is found, it is the global optimum of the NLP if it is convex. The Hessian of the Lagrange function is positive definite for convex problems. Thus the sufficient condition is also satisfied. If the NLP is nonconvex usually only a locally optimal solution can be found. The textbooks by Bertsekas [Ber99] and Nocedal and Wright [NW06] give an extensive overview of nonlinear programming methods.

2.1.6 Mixed-integer programming

The optimization problems discussed in the previous section only included continuous variables $\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}}$. However, many practical applications require that some or all variables only take integer values. This gives rise to mixed-integer programming problems.

Definition 11: Mixed-integer program

An optimization problem of the form

$$\min_{\mathbf{x},\mathbf{z}} f(\mathbf{x},\mathbf{z}), \tag{2.22a}$$

$$\text{s.t. } \mathbf{g}(\mathbf{x},\mathbf{z}) \leq \mathbf{0}, \tag{2.22b}$$

$$\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}},\ \mathbf{z} \in \mathbb{Z}^{n_\mathbf{z}} \tag{2.22c}$$

with $f\colon \mathbb{R}^{n_\mathbf{x}} \times \mathbb{Z}^{n_\mathbf{z}} \to \mathbb{R}$ and $\mathbf{g}\colon \mathbb{R}^{n_\mathbf{x}} \times \mathbb{Z}^{n_\mathbf{z}} \to \mathbb{R}^{n_\mathbf{g}}$ is called a mixed-integer program (MIP).

Different classes of MIPs can be defined according to the objective function and constraints, e.g., mixed-integer linear programs (MILP) if both the objective and constraints are linear, mixed-integer quadratic programs (MIQP) if the objective is quadratic and the constraints are linear, etc. An important property for all MIPs is that they are nonconvex, independent of the convexity of the objective function $f(\mathbf{x}, \mathbf{z})$ and the constraints $\mathbf{g}(\mathbf{x}, \mathbf{z})$. The nonconvexity is a result of the integrality constraints. This is illustrated in Fig. 2.4. Fig. 2.4a shows a convex feasible set formed by linear (inequality) constraints with continuous decision variables. If integrality constraints are added to the variables, the feasible set is comprised of integer values, as depicted in Fig. 2.4b. Points on the line segment connecting two feasible points are not integer, i.e., not feasible. Thus the feasible set and hence the corresponding optimization problem is nonconvex (cf. Def. 5 and Def. 7)

MIP problems are usually harder to solve than their continuous counterparts. If both the objective function $f(\mathbf{x}, \mathbf{z})$ and the constraints $\mathbf{g}(\mathbf{x}, \mathbf{z})$ are convex, the branch-&-bound algorithm can be employed to solve them [LD60]. To this end, the integrality constraints are first relaxed and the solution of a continuous convex problem is obtained,

$$(\tilde{\mathbf{x}}^*, \tilde{\mathbf{z}}^*) = \operatorname*{argmin}_{\mathbf{x},\mathbf{z}} f(\mathbf{x}, \mathbf{z}), \tag{2.23a}$$

$$\text{s.t. } \mathbf{g}(\mathbf{x}, \mathbf{z}) \leq \mathbf{0}, \tag{2.23b}$$

$$\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}},\ \mathbf{z} \in \mathbb{R}^{n_\mathbf{z}}. \tag{2.23c}$$

The MIP problem (2.22) and its continuous relaxation (2.23) differ in the omission of the integrality constraints $\mathbf{z} \in \mathbb{Z}^{n_\mathbf{z}}$. Usually, some components of the obtained solution $\tilde{\mathbf{z}}^*$ will violate the integrality constraints. A fractional component of the obtained solution $[\tilde{\mathbf{z}}^*]_l \notin \mathbb{Z}$ is then selected and branching is performed, i.e., two new relaxed problems are generated:

$$\min_{\mathbf{x},\mathbf{z}} f(\mathbf{x}, \mathbf{z}), \tag{2.24a}$$

$$\text{s.t. } \mathbf{g}(\mathbf{x}, \mathbf{z}) \leq \mathbf{0}, \tag{2.24b}$$

$$\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}},\ \mathbf{z} \in \mathbb{R}^{n_\mathbf{z}}, \tag{2.24c}$$

$$[\mathbf{z}]_l \leq \lfloor [\tilde{\mathbf{z}}^*]_l \rfloor, \tag{2.24d}$$

$$\min_{\mathbf{x},\mathbf{z}} f(\mathbf{x}, \mathbf{z}), \tag{2.25a}$$

$$\text{s.t. } \mathbf{g}(\mathbf{x}, \mathbf{z}) \leq \mathbf{0}, \tag{2.25b}$$

$$\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}},\ \mathbf{z} \in \mathbb{R}^{n_\mathbf{z}}, \tag{2.25c}$$

$$[\mathbf{z}]_l \geq \lceil [\tilde{\mathbf{z}}^*]_l \rceil. \tag{2.25d}$$

The relaxation (2.23) is referred to as the root node. Since the feasible sets of the child nodes (2.24) and (2.25) have been reduced compared to the root node (2.23), the latter's optimal objective $f(\tilde{\mathbf{x}}^*, \tilde{\mathbf{z}}^*)$ is a lower bound on the objectives of the former. If the solutions of the nodes violate the integrality constraints, new child nodes are created again by branching on the fractional-valued variables. Once a feasible solution to problem (2.22) has been found it constitutes an upper bound on the optimal objective. The branch-&-bound algorithm prunes branches whose objectives become worse than the best current upper bound. Additionally, the objective value of a node provides a lower bound for the objectives of all subsequent child nodes. The branching continues until the upper and lower bounds are equal (global optimum) or until their relative difference reaches a

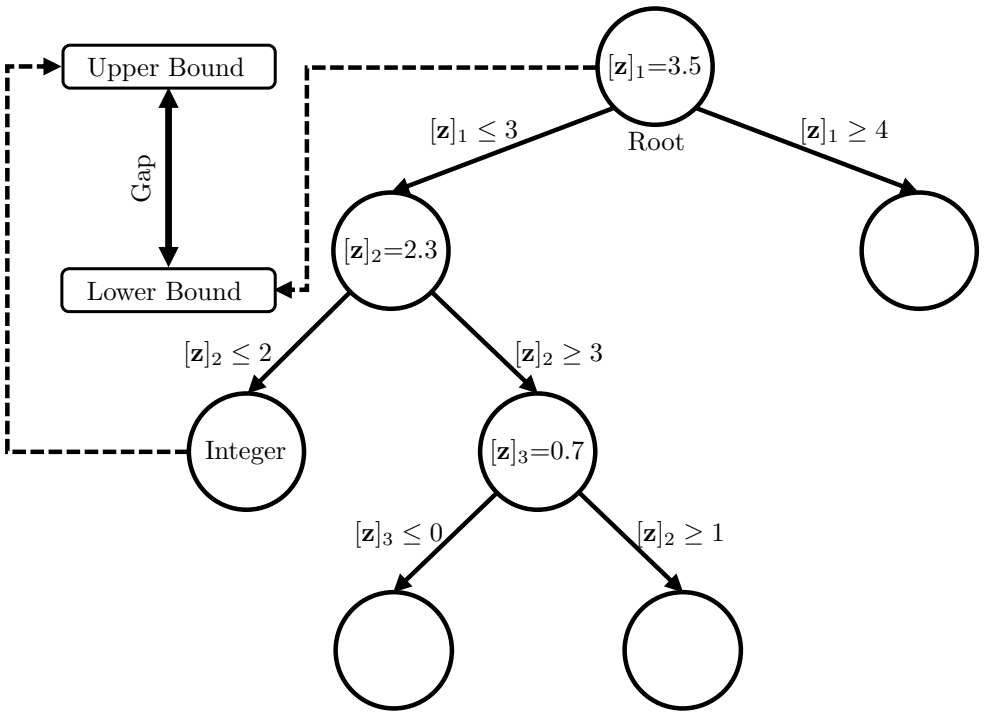


Figure 2.5: Illustration of the branch-&-bound algorithm. Each node in the branch-&-bound tree corresponds to a continuous optimization problem. When an integer variable takes a fractional value, two new nodes are created.

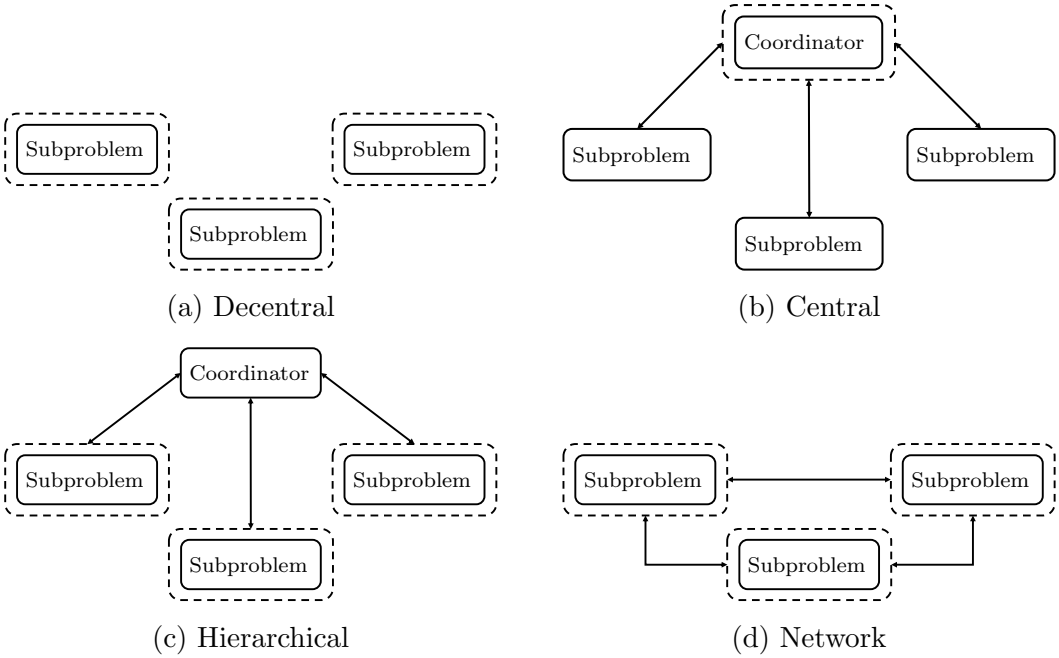
specified tolerance. The difference between the best upper and lower bounds is referred to as the integrality gap. An exemplary branch-&-bound tree is depicted in Fig. 2.5.

The branch-&-bound algorithm is usually computationally expensive since a continuous subproblem has to be solved at each node of the resulting tree. Nevertheless, the performance of commercial solvers like Gurobi [Gur23], CPLEX [IBM22] or Xpress [FIC22] has significantly improved in recent years, enabling the solution of large-scale mixed-integer programs [AW13]. These solvers employ multiple other techniques, e.g., cutting planes [MMW$^+$02], presolve reductions [ABG$^+$20], branching rules [AKM05], and heuristics [DRP05, BFL07, Rot07] that significantly enhance the performance of the branch-&-bound algorithm.

2.2 Distributed optimization

The following section provides an overview of distributed optimization. After presenting a classification of distributed optimization architectures, dual decomposition is discussed in detail. Afterward, related distributed optimization approaches are discussed and compared to dual decomposition. This section has been partially published in [YWW$^+$23].

2.2.1 Classification



(a) Decentral (b) Central

(c) Hierarchical (d) Network

Figure 2.6: Different architectures for optimization problems consisting of multiple subproblems. The dashed lines indicate the solution of an optimization problem.

The choice of a distributed optimization approach for problems consisting of multiple subproblems strongly depends on their communication architecture. Fig. 2.6 depicts the most common architectures (cf. [YJ10, Max21]).

If the subproblems are not coupled or if the coupling is weak, i.e., if it does not affect the feasibility of the overall problem, the optimization can be performed in a decentral manner (Fig. 2.6a). In this case, no information is exchanged between the subproblems. Fig. 2.6b depicts the case where the information of each subproblem is gathered by a central coordination instance which performs a monolithic optimization of the system-wide problem.

Distributed optimization can be employed if a certain degree of autonomy of and confidentiality between the subproblems is desired. In a distributed optimization setting each subproblem is solved independently and shares information either with a coordinator or with the other subproblems. The hierarchical architecture is depicted in Fig. 2.6c. In this case, the subproblems only share information with the coordinator who is responsible for the satisfaction of system-wide constraints. This ensures a high degree of confidentiality between the subproblems. In contrast, Fig. 2.6d shows the case where no coordinator is present. The subproblems directly communicate with their neighbors and exchange information. This setting is often referred to as network optimization.

In addition to the communication architecture, distributed optimization algorithms can also be classified according to their formulation structure. Tosserams et al. propose the distinction between nested and alternating formulations [TER09]. Nested formulations are

bilevel problems, where the subproblems are nested in the function of a master problem (e.g. Benders decomposition). Alternating formulations alternate between the solution of a coordinator problem and the subproblems sequentially (e.g. dual and primal decomposition).

This thesis focuses on distributed optimization problems with a hierarchical communication structure and alternating formulations. The following subsections provide an overview of relevant distributed optimization approaches.

2.2.2 Dual decomposition

This thesis deals with optimization problems of the form

$$\min_{\mathbf{x}_1,\dots,\mathbf{x}_{N_s}} \sum_{i\in\mathcal{I}} f_i(\mathbf{x}_i), \tag{2.26a}$$

$$\text{s.t.} \sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i \leq \mathbf{b}, \tag{2.26b}$$

$$\mathbf{x}_i \in \mathcal{X}_i,\ \forall i \in \mathcal{I}. \tag{2.26c}$$

(2.26) describes an optimization problem consisting of N_s subproblems $i \in \mathcal{I} = \{1,\dots,N_s\}$. Each subproblem has its own set of decision variables $\mathbf{x}_i \in \mathbb{K}^{n_{\mathbf{x}_i}}$ and an objective function $f_i\colon \mathbb{K}^{n_{\mathbf{x}_i}} \to \mathbb{R}$, with $\mathbb{K} \in \{\mathbb{R}, \mathbb{R}\times\mathbb{Z}\}$, i.e., continuous or mixed-integer optimization problems. The subproblems are coupled through the system-wide constraints (2.26b), also referred to as coupling, complicating, or network constraints. The terms $\mathbf{A}_i\mathbf{x}_i$, with $\mathbf{A}_i \in \mathbb{R}^{n_\mathbf{b}\times n_{\mathbf{x}_i}}$ can be interpreted as the utilization of shared limited resources depending on the decision variables $\mathbf{x}_i$, while $\mathbf{b} \in \mathbb{R}^{n_\mathbf{b}}$ represents the availability of these resources. In addition to the system-wide constraints, each subproblem i contains individual constraints $\mathbf{x}_i \in \mathcal{X}_i \subset \mathbb{K}^{n_{\mathbf{x}_i}}$, where $\mathcal{X}_i$ is a non-empty compact set. The system-wide objective function is assumed to be additive in the subproblem objective function values. The goal is to minimize the sum of the objective functions of all subproblems (2.26a), also called a social welfare objective [SZ19], while satisfying the system-wide constraints (2.26b) as well as the individual constraints (2.26c).

Problem (2.26) is separable in its objective function and the subproblems are only coupled through constraints. This class of problems is referred to as constraint-coupled optimization problems [NNC19]. The system-wide or central problem can be decomposed by introducing dual variables $\boldsymbol{\lambda} \in \mathbb{R}^{n_\mathbf{b}}$ for the coupling constraints. With the dual variables, the Lagrange function can be formulated (cf. Def. 3),

$$\mathcal{L}(\mathbf{x},\boldsymbol{\lambda}) = \sum_{i\in\mathcal{I}} f_i(\mathbf{x}_i) + \boldsymbol{\lambda}^T \sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i - \boldsymbol{\lambda}^T\mathbf{b}, \tag{2.27}$$

where $\mathbf{x} = [\mathbf{x}_1^T,\dots,\mathbf{x}_{N_s}^T]^T$. In the Lagrange function (2.27) the system-wide constraints are relaxed and weighed by the dual variables. The optimal value of the Lagrange function for a given value of the dual variables corresponds to the value of the dual function, which

is a key component of dual decomposition-based distributed optimization.

Definition 12: Dual function

The function $d\colon \mathbb{R}^{n_\mathfrak{b}} \to \mathbb{R}$ mapping the dual variables $\boldsymbol{\lambda}$ to the corresponding optimal objective value of the Lagrange function (2.27), i.e.,

$$d(\boldsymbol{\lambda}) \coloneqq \inf_{\mathbf{x}_i \in \mathcal{X}_i, \forall i \in \mathcal{I}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}), \tag{2.28}$$

is called the dual function of the system-wide optimization problem (2.26).

The dual function is a function of the dual variables $\boldsymbol{\lambda}$. The domain of the dual variables is $\boldsymbol{\lambda} \geq \mathbf{0}$, stemming from the KKT conditions (2.5) and from the fact that the system-wide constraints (2.26b) are inequalities [NW06]. Generally, the domain of the dual variables $\boldsymbol{\lambda}$ are all values for which the Lagrange function (2.27) is bounded from below, i.e., all values for which $d(\boldsymbol{\lambda}) > -\infty$.

An important property of the dual function (2.28) is that its values always provide a lower bound for the objective value of the system-wide problem (2.26). This is easily verifiable (cf. [BV04]). Suppose that $\tilde{\mathbf{x}}$ is a feasible solution to problem (2.26). Then, since we have $\sum_{i \in \mathcal{I}} \mathbf{A}_i \tilde{\mathbf{x}}_i - \mathbf{b} \leq \mathbf{0}$ and $\boldsymbol{\lambda} \geq \mathbf{0}$,

$$\boldsymbol{\lambda}^T \left(\sum_{i \in \mathcal{I}} \mathbf{A}_i \tilde{\mathbf{x}}_i - \mathbf{b} \right) \leq 0 \tag{2.29}$$

holds. This implies that

$$\mathcal{L}(\tilde{\mathbf{x}}, \boldsymbol{\lambda}) = \sum_{i \in \mathcal{I}} f_i(\tilde{\mathbf{x}}_i) + \boldsymbol{\lambda}^T \left(\sum_{i \in \mathcal{I}} \mathbf{A}_i \tilde{\mathbf{x}}_i - \mathbf{b} \right) \leq \sum_{i \in \mathcal{I}} f_i(\tilde{\mathbf{x}}_i). \tag{2.30}$$

Therefore,

$$d(\boldsymbol{\lambda}) = \inf_{\mathbf{x}_i \in \mathcal{X}_i, \forall i \in \mathcal{I}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) \leq \mathcal{L}(\tilde{\mathbf{x}}, \boldsymbol{\lambda}) \leq \sum_{i \in \mathcal{I}} f_i(\tilde{\mathbf{x}}_i) \tag{2.31}$$

holds for any feasible primal and dual solution $\tilde{\mathbf{x}}$ and $\boldsymbol{\lambda}$. Since the dual function provides a lower bound for the objective values of the system-wide problem for any feasible solution $\tilde{\mathbf{x}}$, it also does so in the case of the optimal solution $\mathbf{x}^*$. Naturally, one would be interested in the best attainable lower bound, corresponding to the maximum of the dual function. Finding this maximum is referred to as the dual problem.

Definition 13: Dual problem

The optimization problem

$$\max_{\boldsymbol{\lambda} \in \mathbb{R}^{n_\mathbf{b}}} d(\boldsymbol{\lambda}), \tag{2.32a}$$

$$\text{s.t. } \boldsymbol{\lambda} \geq \mathbf{0}, \tag{2.32b}$$

i.e., the problem of finding the dual variables that provide the best lower bound for the system-wide problem (2.26) is referred to as its dual problem. In contrast, the system-wide problem (2.26) is called the primal problem.

The optimal solution of the dual problem (2.32) is denoted by $\boldsymbol{\lambda}^*$. The optimal solution of the dual problem (2.32) always provides a bound on the optimal solution of the primal problem (2.26). In the case of convex primal problems solving the dual problem is equivalent to solving the primal problem, i.e., optimizing the Lagrange function with the optimal dual variables yields the optimal primal variables and the same objective value. These properties are referred to as weak and strong duality.

Definition 14: Weak and strong duality and duality gap

Due to the lower bound property of the dual function the relation

$$\sum_{i \in \mathcal{I}} f_i(\mathbf{x}_i^*) \geq d(\boldsymbol{\lambda}^*) \tag{2.33}$$

holds between the primal and dual optimal solutions. Inequality (2.33) is referred to as weak duality. The difference between the objective values of a feasible primal and a feasible dual solution is called the duality gap,

$$\text{DG} = \sum_{i \in \mathcal{I}} f_i(\mathbf{x}_i) - d(\boldsymbol{\lambda}). \tag{2.34}$$

If the primal problem, i.e., the system-wide problem (2.26), is convex and a constraint qualification condition is satisfied, the optimal duality gap is zero. This implies, that

$$\sum_{i \in \mathcal{I}} f_i(\mathbf{x}_i^*) = d(\boldsymbol{\lambda}^*). \tag{2.35}$$

This condition is referred to as strong duality.

Constraint qualifications are regularity conditions that express whether an optimal solution of the primal problem has to satisfy the KKT conditions [NW06]. A commonly used constraint qualification condition is Slater's condition.

Definition 15: Slater's condition [BV04]

Assume that the individual constraints (2.26c) of the system-wide optimization problem have the general structure $\mathcal{X}_i = \{\mathbf{x}_i \in \mathbb{R}^{n_{\mathbf{x}_i}} |\ \mathbf{g}_i(\mathbf{x}_i) \leq 0,\ \mathbf{h}_i(\mathbf{x}_i) = \mathbf{0}\},\ \forall i \in \mathcal{I}$. Then, if there exists a strictly feasible solution $\tilde{\mathbf{x}}$ to problem (2.26), i.e.,

$$\exists\, \tilde{\mathbf{x}} \in \left\{ \mathbf{x} \in \mathbb{R}^{n_{\mathbf{x}}} |\ \mathbf{g}_i(\mathbf{x}_i) < 0,\ \mathbf{h}_i(\mathbf{x}_i) = \mathbf{0},\ \forall i \in \mathcal{I},\ \sum_{i \in \mathcal{I}} \mathbf{A}_i \mathbf{x}_i < \mathbf{b} \right\}, \tag{2.36}$$

then problem (2.26) satisfies Slater's constraint qualification condition.

Note that $\mathbb{R}$ is used in Def. 15 instead of $\mathbb{K}$, since mixed-interger programs are inherently nonconvex.

Another important property of the dual problem (2.32) is that the dual function is always concave, regardless of whether or not the primal problem is convex. This is also easily verifiable (cf. Theorem 12.10, [NW06]). For any feasible $\tilde{\boldsymbol{\lambda}}$, $\hat{\boldsymbol{\lambda}}$ and $\theta \in [0, 1]$ we have

$$\mathcal{L}(\mathbf{x}, (1-\theta)\tilde{\boldsymbol{\lambda}} + \theta\hat{\boldsymbol{\lambda}}) = (1-\theta)\mathcal{L}(\mathbf{x}, \tilde{\boldsymbol{\lambda}}) + \theta\mathcal{L}(\mathbf{x}, \hat{\boldsymbol{\lambda}}). \tag{2.37}$$

By taking the infimum of both sides in this expression, using the definition of the dual function (2.28), and using the result that the infimum of a sum is greater or equal to the sum of the infima we get

$$d((1-\theta)\tilde{\boldsymbol{\lambda}} + \theta\hat{\boldsymbol{\lambda}}) \geq (1-\theta)d(\tilde{\boldsymbol{\lambda}}) + \theta d(\hat{\boldsymbol{\lambda}}), \tag{2.38}$$

which proves concavity of $d(\boldsymbol{\lambda})$. Since the dual problem (2.32) is a maximization of a concave function over a convex feasible set, it is a convex optimization problem.

Dual decomposition-based distributed optimization algorithms rely on the solution of the dual problem (2.32). Its solution is amendable to distributed computations since the Lagrange function (2.27) is separable due to the relaxation of the system-wide constraints (2.26b). This means that the dual function can be evaluated by solving the individual optimization problems

$$\min_{\mathbf{x}_i \in \mathbb{R}^{n_{\mathbf{x}_i}}} \mathcal{L}_i(\mathbf{x}_i, \boldsymbol{\lambda}), \tag{2.39a}$$

$$\text{s.t. } \mathbf{x}_i \in \mathcal{X}_i \tag{2.39b}$$

in a distributed manner for a given value of the dual variables $\boldsymbol{\lambda}$. In the case of nonconvex primal problems strong duality (2.35) does not hold. Upon convergence, a feasible primal solution is usually recovered through the use of problem-specific heuristics [BLY+15] or by modifying the primal problem a priori [VEG+16]. Note that the lower bound of problem (2.39) is assumed to be attainable, therefore replacing the infimum with the minimum.

Example: Consider the following optimization problem:

$$\min_{x_1,x_2\in\mathbb{R}} 0.5x_1^2 + 0.5(x_2-1)^2, \tag{2.40a}$$

$$\text{s.t. } x_1 + x_2 = 0, \tag{2.40b}$$

$$x_1 \leq 1. \tag{2.40c}$$

The Lagrange function of problem (2.40) is

$$\mathcal{L}(x_1, x_2, \lambda) = 0.5x_1^2 + 0.5(x_2-1)^2 + \lambda(x_1 + x_2) \tag{2.41}$$

and the dual function

$$d(\lambda) = \min_{x_1,x_2\in\mathbb{R}} 0.5x_1^2 + 0.5(x_2-1)^2 + \lambda(x_1 + x_2), \tag{2.42a}$$

$$\text{s.t. } x_1 \leq 1. \tag{2.42b}$$

Note that since the system-wide constraint (2.40b) is an equality, the domain of the dual variable is $\mathbb{R}$. The value of the dual function now depends on whether or not the individual constraint (2.40c) is active. Two cases can be distinguished:

Case 1: $x_1 < 1$ (inactive constraint)

For a given value of λ the value of the dual function (2.42) can be computed by setting the gradient of the Lagrange function (2.41) to zero:

$$\nabla\mathcal{L}(x_1, x_2, \lambda) = \begin{pmatrix} x_1 + \lambda \\ x_2 - 1 + \lambda \end{pmatrix} = \mathbf{0} \Rightarrow x_1 = -\lambda,\ x_2 = 1 - \lambda. \tag{2.43}$$

Substituting the values of x_1 and x_2 in (2.41) gives

$$d(\lambda) = -\lambda^2 + \lambda,\ \forall\lambda > -1. \tag{2.44}$$

Case 2: $x_1 = 1$ (active constraint)

Setting the gradient of the reduced Lagrange function to zero gives

$$\nabla\mathcal{L}(1, x_2, \lambda) = x_2 - 1 + \lambda = 0 \Rightarrow x_2 = 1 - \lambda. \tag{2.45}$$

Again, substituting the values for x_1 and x_2 into (2.41) gives

$$d(\lambda) = -\lambda^2 + 1.5\lambda + 0.5,\ \forall\lambda \leq -1. \tag{2.46}$$

The dual function for problem (2.40) is given by

$$d(\lambda) = \begin{cases} -\lambda^2 + \lambda, & \forall\lambda > -1, \\ -\lambda^2 + 1.5\lambda + 0.5, & \forall\lambda \leq -1. \end{cases} \tag{2.47}$$

It is easy to see that the dual function is continuous as

$$\lim_{\lambda\to-1^+} d(\lambda) = \lim_{\lambda\to-1^-} d(\lambda) = -2. \tag{2.48}$$

However, the dual function does not have continuous derivatives,

$$\lim_{\lambda \to -1^+} \nabla d(\lambda) = \lim_{\lambda \to -1^+} -2\lambda + 1 = 3, \tag{2.49a}$$

$$\lim_{\lambda \to -1^-} \nabla d(\lambda) = \lim_{\lambda \to -1^-} -2\lambda + 1.5 = 4.5, \tag{2.49b}$$

which means that it is not smooth. Hence, as the dual function is always concave, the dual problem

$$\max_{\lambda \in \mathbb{R}} d(\lambda) \tag{2.50}$$

is a convex, but nonsmooth optimization problem. Generally, a nonsmooth optimization problem can be defined as follows:

Definition 16: Nonsmooth optimization problem

An optimization problem

$$\min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}). \tag{2.51}$$

is called nonsmooth, if the gradient of the objective function exhibits discontinuities, i.e.,

$$\exists \tilde{\mathbf{x}} \in \mathcal{X}, \mathbf{s} \in \mathbb{R}^{n_\mathbf{x}}, \lim_{\alpha \to 0^+} \nabla f(\tilde{\mathbf{x}} + \alpha \mathbf{s}) \neq \lim_{\alpha \to 0^-} \nabla f(\tilde{\mathbf{x}} + \alpha \mathbf{s}). \tag{2.52}$$

Nonsmoothness is typical for dual optimization problems and is caused by a changing set of active constraints for different values of the dual variables. Algorithms that aim to solve the nonsmooth dual problem are reviewed in Chapter 3 in the context of dual decomposition-based distributed optimization. These algorithms rely on a hierarchical communication structure, where a coordinator sends the dual variables $\boldsymbol{\lambda}$ to each subproblem and in turn receives the corresponding responses of the subproblems. Dual decomposition-based distributed optimization is often interpreted as a market mechanism, where the coordinator acts as an auctioneer for shared limited resources [Wen20]. In this context, the dual variables can be interpreted as prices for these resources. The subproblems then decide on their resource production or consumption depending on their current price and communicate this decision to the coordinator. The coordinator then adjusts the prices depending on the resource imbalance.

A main advantage of this hierarchical structure is that only limited information has to be shared between the subproblems and the coordinator while no information has to be directly exchanged between the subproblems. This is desirable in a setting where confidentiality between the subproblems has to be maintained, e.g., because they belong to different companies.

Depending on the communication structure and the availability of shared information, different decomposition, and distributed optimization approaches can be utilized. Some

algorithms are briefly discussed in the following section.

2.2.3 Other distributed optimization approaches

This section briefly discusses distributed optimization algorithms other than dual decomposition, namely primal decomposition, Dantzig-Wolfe decomposition, Benders decomposition, and the Jacobi and Gauss-Seidel algorithms.

Primal decomposition

In dual decomposition, prices for shared limited resources are introduced to decouple the subproblems. Thus, this method is also referred to as price coordination. The inverse approach is primal decomposition or resource coordination. Instead of assigning prices to the resources and communicating them to the subproblems, the coordinator directly assigns available resources $\mathbf{u}_i$ to the subproblems. This leads to the definition of the primal function,

$$p(\mathbf{u}) = \min_{\mathbf{x}_1,\ldots,\mathbf{x}_{N_s}} \sum_{i\in\mathcal{I}} f_i(\mathbf{x}_i), \tag{2.53a}$$

$$\text{s.t. } \mathbf{A}_i\mathbf{x}_i \leq \mathbf{u}_i,\ \forall i \in \mathcal{I} \tag{2.53b}$$

$$\mathbf{x}_i \in \mathcal{X}_i,\ \forall i \in \mathcal{I}, \tag{2.53c}$$

which represents the optimal value of the system-wide problem (2.26) for a given resource distribution $\mathbf{u}$, where $\mathbf{u} = [\mathbf{u}_1^T, \ldots, \mathbf{u}_{N_s}^T]^T \in \mathbb{R}^{N_s \cdot n_{\mathbf{b}}}$ is the amount of resources made available to each subsystem. Similar to dual decomposition, primal decomposition-based distributed optimization algorithms rely on the solution of the problem

$$\min_{\mathbf{u}_1,\ldots,\mathbf{u}_{N_s}} p(\mathbf{u}), \tag{2.54a}$$

$$\text{s.t. } \sum_{i\in\mathcal{I}} \mathbf{u}_i \leq \mathbf{b}, \tag{2.54b}$$

which is also amendable to distributed computations. For a given resource distribution $\mathbf{u}$ the value of the primal function can be computed by solving the subproblems

$$\min_{\mathbf{x}_i} f_i(\mathbf{x}_i), \tag{2.55a}$$

$$\text{s.t. } \mathbf{A}_i\mathbf{x}_i \leq \mathbf{u}_i, \tag{2.55b}$$

$$\mathbf{x}_i \in \mathcal{X}_i \tag{2.55c}$$

in a distributed manner.

The main advantage of primal compared to dual decomposition is that if a suitable update scheme for the variables $\mathbf{u}$ is used, such that constraint (2.54b) is satisfied, the corresponding obtained solution to the system-wide problem (2.26) will always be feasible, as long as a feasible solution to the subproblems (2.55) can be found. This is not the case for

dual decomposition, as the system-wide problem (2.26) is generally only feasible once the solution $\boldsymbol{\lambda}^*$ of the dual problem (2.32) has been found. Nevertheless, finding a suitable update scheme for the variables $\mathbf{u}$ such that feasibility is guaranteed is not trivial.

A drawback of primal decomposition is that the subproblems lose their autonomy to a certain extent, as the coordinator directly assigns available resources to them. In contrast, when dual decomposition is employed the subproblems can make completely autonomous decisions and only react to the price signals from the coordinator. From a mathematical point of view, the primal function (2.53) is only convex if the system-wide problem (2.26) is also convex. In contrast, the dual function is always concave. Furthermore, the dual problem (2.32) contains fewer variables ($n_{\mathbf{b}}$) than problem (2.54) ($N_s \cdot n_{\mathbf{b}}$), which improves the scalability for a large number of subproblems. More details on primal decomposition and its relation to dual decomposition are provided by, e.g., Palomar and Chiang [PC06] and Yang and Johansson [YJ10].

Dantzig–Wolfe decomposition

One of the early decomposition-based optimization algorithms is the Dantzig-Wolfe decomposition [DW60], which can be used to solve block diagonal LPs of the form

$$\min_{\mathbf{x}_1,\ldots,\mathbf{x}_{N_s}} \quad \sum_{i\in\mathcal{I}} \mathbf{c}_i^T \mathbf{x}_i, \tag{2.56a}$$

$$\text{s. t.} \quad \sum_{i\in\mathcal{I}} \mathbf{A}_i \mathbf{x}_i = \mathbf{b}, \tag{2.56b}$$

$$\mathbf{D}_i \mathbf{x}_i \leq \mathbf{d}_i, \ \forall i \in \mathcal{I}, \tag{2.56c}$$

$$\mathbf{x}_i \geq \mathbf{0}, \ \forall i \in \mathcal{I}, \tag{2.56d}$$

with $\mathcal{I} = \{1, \ldots, N_s\}$, $\mathbf{c}_i, \mathbf{x}_i \in \mathbb{R}^{n_{\mathbf{x}_i}}$, $\mathbf{A}_i \in \mathbb{R}^{n_{\mathbf{b}} \times n_{\mathbf{x}_i}}$, $\mathbf{b} \in \mathbb{R}^{n_{\mathbf{b}}}$, $\mathbf{D}_i \in \mathbb{R}^{n_{\mathbf{d}_i} \times n_{\mathbf{x}_i}}$ and $\mathbf{d}_i \in \mathbb{R}^{n_{\mathbf{d}_i}}$. In the Dantzig-Wolfe decomposition method problem (2.56) is decomposed into a master problem and N_s subproblems. The solution of the master problem provides a new objective function to the subproblems and ensures that the coupling constraints (2.56b) are satisfied. In turn, the solutions of the subproblems are sent to the master problem, and new columns (new variables) are added to the master problem if they improve the overall objective. This process is repeated until no further improvements can be made.

Dual decomposition-based algorithms consider similar problem structures as the Dantzig-Wolfe decomposition method. However, the size of the master problem, i.e., the dual problem, does not increase throughout the iterations, as the number of dual variables remains fixed.

Benders decomposition

The Dantzig-Wolfe decomposition method relies on column generation. The inverse approach is row generation, i.e., the generation of new constraints for the master problem.

This is the case in the Benders decomposition method [Ben62]. It can be used to solve problems of the form

$$\min_{\mathbf{x}_1,\dots,\mathbf{x}_{N_s},\mathbf{z}} \; c_{\mathbf{z}}^T\mathbf{z} + \sum_{i\in\mathcal{I}} \mathbf{c}_{\mathbf{x}_i}^T\mathbf{x}_i, \tag{2.57a}$$

$$\text{s.t. } \mathbf{A}_{\mathbf{z}_i}\mathbf{z} + \mathbf{A}_{\mathbf{x}_i}\mathbf{x}_i = \mathbf{b}_i, \; \forall i \in \mathcal{I} \tag{2.57b}$$

$$\mathbf{D}\mathbf{z} = \mathbf{d}, \tag{2.57c}$$

$$\mathbf{x}_i \geq \mathbf{0}, \tag{2.57d}$$

$$\mathbf{z} \in \mathbb{K}^{n_{\mathbf{z}}}, \; \mathbf{x}_i \in \mathbb{K}^{n_{\mathbf{x}_i}}, \; \forall i \in \mathcal{I} \tag{2.57e}$$

with $\mathbb{K} \in \{\mathbb{R}, \mathbb{Z}\}$, $\mathbf{c}_{\mathbf{x}_i} \in \mathbb{R}^{n_{\mathbf{x}_i}}$, $\mathbf{c}_{\mathbf{z}} \in \mathbb{R}^{n_{\mathbf{z}}}$, $\mathbf{A}_{\mathbf{x}_i} \in \mathbb{R}^{n_{\mathbf{b}_i}\times n_{\mathbf{x}_i}}$, $\mathbf{A}_{\mathbf{z}_i} \in \mathbb{R}^{n_{\mathbf{b}_i}\times n_{\mathbf{z}}}$, $\mathbf{b}_i \in \mathbb{R}^{n_{\mathbf{b}_i}}$, $\mathbf{D} \in \mathbb{R}^{n_{\mathbf{d}}\times n_{\mathbf{z}}}$ and $\mathbf{d} \in \mathbb{R}^{n_{\mathbf{d}}}$. The variables $\mathbf{z}$ are called complicating variables. The problem structure (2.57) is often encountered in multistage stochastic programming, where the first stage decisions (here-and-now) are modeled by the complicating variables $\mathbf{z}$, while the variables $\mathbf{x}_i$ model the decisions on the nodes of the uncertainty scenario tree [BL11].

For a fixed value of the complicating variables $\tilde{\mathbf{z}} \in \mathcal{Z} = \{\mathbf{z} \in \mathbb{K}^{n_{\mathbf{z}}} | \; \mathbf{D}\mathbf{z} = \mathbf{d}\}$ the residual problem

$$\min_{\mathbf{x}_1,\dots,\mathbf{x}_{N_s}} \; \sum_{i\in\mathcal{I}} \mathbf{c}_{\mathbf{x}_i}^T\mathbf{x}_i, \tag{2.58a}$$

$$\text{s.t. } \mathbf{A}_{\mathbf{x}_i}\mathbf{x}_i = \mathbf{b}_i - \mathbf{A}_{\mathbf{z}_i}\tilde{\mathbf{z}}, \; \forall i \in \mathcal{I}, \tag{2.58b}$$

$$\mathbf{x}_i \geq \mathbf{0}, \; \forall i \in \mathcal{I} \tag{2.58c}$$

can be defined. Problem (2.58) can be decomposed and solved in a distributed manner for a fixed value of the complicating variables $\tilde{\mathbf{z}}$. With the feasible set of each subproblem in problem (2.58) $\mathcal{X}_i(\tilde{\mathbf{z}}) = \{\mathbf{x}_i \in \mathbb{K}^{n_{\mathbf{x}_i}} | \; \mathbf{A}_{\mathbf{x}_i}\mathbf{x}_i = \mathbf{b}_i - \mathbf{A}_{\mathbf{z}_i}\tilde{\mathbf{z}} \wedge \mathbf{x}_i \geq \mathbf{0}\}$ problem (2.57) can be reformulated as

$$\min_{\tilde{\mathbf{z}}\in\mathcal{Z}} \left\{ \mathbf{c}_{\mathbf{z}}^T\tilde{\mathbf{z}} + \min_{\mathbf{x}_i\in\mathcal{X}_i(\tilde{\mathbf{z}}),\forall i\in\mathcal{I}} \; \sum_{i\in\mathcal{I}} \mathbf{c}_{\mathbf{x}_i}^T\mathbf{x}_i \right\}. \tag{2.59}$$

The outer optimization problem in (2.59) is the master problem while the inner problem is the subproblem. In the Benders decomposition algorithm, the dual formulation of the inner problem is used to formulate constraints (cuts) for the master problem. Rahmaniani et al. provide an extensive literature review of the Benders decomposition algorithm [RCG+17].

Benders decomposition is suitable for problems that are coupled through complicating variables while dual decomposition considers complicating constraints. Dual decomposition can also be applied to problems with complicating variables by introducing local copies of these variables in each subproblem and coupling the subproblems through consensus constraints.

Jacobi and Gauss-Seidel algorithms

Dual and primal decomposition as well as the Dantzig-Wolfe and Benders decomposition algorithms exploit a special structure of the constraints of the overall problem with separable objective functions. In contrast, the Jacobi and Gauss-Seidel algorithms deal with problems that are coupled in their objective function,

$$\min_{\mathbf{x}_1,\ldots,\mathbf{x}_{N_s}} f(\mathbf{x}_1,\ldots,\mathbf{x}_{N_s}), \tag{2.60a}$$

$$\mathbf{x}_i \in \mathcal{X}_i. \tag{2.60b}$$

In the Jacobi algorithm, all variables are updated simultaneously in each iteration t according to

$$\mathbf{x}_i^{(t+1)} = \arg\min_{\mathbf{x}_i \in \mathcal{X}_i} f(\mathbf{x}_1^{(t)},\ldots,\mathbf{x}_i,\ldots,\mathbf{x}_{N_s}^{(t)}). \tag{2.61}$$

After all variables have been updated they are shared among all subsystems.

In the Gauss-Seidel algorithm, also called block-coordinate descent, the variables are updated sequentially while fixing the previously updated variables,

$$\mathbf{x}_i^{(t+1)} = \arg\min_{\mathbf{x}_i \in \mathcal{X}_i} f(\mathbf{x}_1^{(t+1)},\ldots,\mathbf{x}_{i-1}^{(t+1)},\mathbf{x}_i,\mathbf{x}_{i+1}^{(t)}\ldots,\mathbf{x}_{N_s}^{(t)}). \tag{2.62}$$

More details on these algorithms are provided in the textbook by Bertsekas and Tsitsiklis [BT89] or by Palomar and Chiang [PC06].

3 Dual Decomposition-based Distributed Optimization Algorithms

The general idea of dual decomposition was introduced by Everett in the early 1960s [Eve63]. Dual decomposition can be regarded as a hierarchical scheme where a coordination algorithm computes values of the dual variables, which are communicated to the subproblems. The subproblems solve their individual optimization problems for the received values of the dual variables and communicate their results back to the coordinator. What information is communicated to the coordinator depends on the specific dual decomposition-based algorithm. Dual decomposition-based distributed optimization can also be interpreted as a market mechanism where an auctioneer sets prices for shared resources and the subproblems compute their optimal resource utilization according to these prices [Wal87, WRE20]. In this context the dual variables are called prices or shadow prices [GKW07].

In this chapter, those dual decomposition-based distributed optimization algorithms which are used as a reference for comparison of the proposed algorithms are discussed. This includes the subgradient method, the bundle trust method (BTM), and the alternating direction method of multipliers (ADMM). Other related dual decomposition-based algorithms are also briefly reviewed. The contents of this chapter have been published in [YWW+23].

3.1 Subgradient method

The simplest distributed optimization algorithm that is based on dual decomposition is the subgradient method. Methods based on subgradients were originally developed in the Soviet Union and used to solve nonsmooth optimization problems [SKR85]. In this algorithm, the dual variables are updated along a subgradient direction of the dual function.

Definition 17: Subgradient [BKM14]

A vector $\boldsymbol{\xi} \in \mathbb{R}^{n_{\boldsymbol{\chi}}}$ is a subgradient of a concave function $\phi\colon \mathbb{R}^{n_{\boldsymbol{\chi}}} \to \mathbb{R}$ at the point $\boldsymbol{\chi}_0 \in \mathbb{R}^{n_{\boldsymbol{\chi}}}$, if

$$\phi(\boldsymbol{\chi}) \leq \boldsymbol{\xi}^T(\boldsymbol{\chi} - \boldsymbol{\chi}_0) + \phi(\boldsymbol{\chi}_0) \tag{3.1}$$

holds for all $\boldsymbol{\chi} \in \operatorname{dom} \phi$. The set of all subgradients is called the subdifferential $\partial\phi(\boldsymbol{\chi}_0)$ of the function $\phi(\boldsymbol{\chi})$ at the point $\boldsymbol{\chi}_0$.

The subgradient is a generalization of the gradient for nonsmooth (non-differentiable) convex functions. Note that (3.1) technically defines a supergradient of a concave function. However, the term subgradient is commonly used in the literature for both convex and concave functions. Geometrically the subgradient/supergradient is a normal vector to a supporting hyperplane of a convex/concave function. Fig. 3.1 illustrates different subgradients both for differentiable (χ_0) and non-differentiable (χ_0') points.

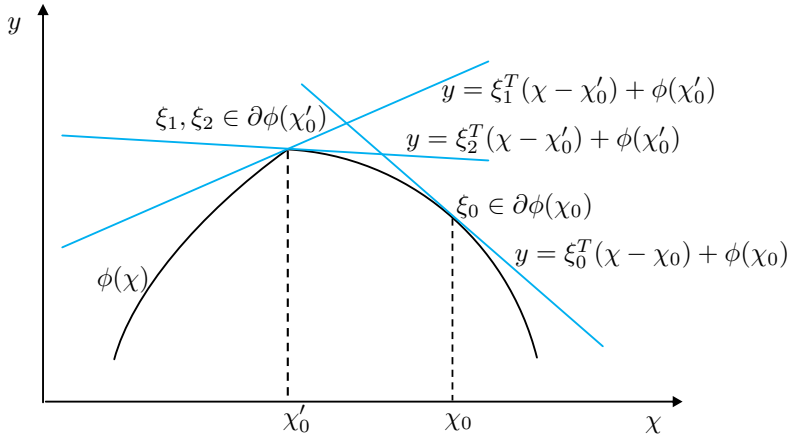


Figure 3.1: Geometric interpretation of subgradients for differentiable (χ_0) and non-differentiable (χ_0') points (from [YWW+23]).

In the subgradient method for distributed optimization, the primal variables $\mathbf{x}_i$ and the dual variables $\boldsymbol{\lambda}$ are updated according to

$$\forall i \in \mathcal{I},\ \mathbf{x}_i^{(t+1)} = \arg\min_{\mathbf{x}_i \in \mathcal{X}_i} \mathcal{L}_i(\mathbf{x}_i, \boldsymbol{\lambda}^{(t)}) \tag{3.2a}$$

$$\boldsymbol{\lambda}^{(t+1)} = \left[\boldsymbol{\lambda}^{(t)} + \alpha^{(t)}\mathbf{g}(\boldsymbol{\lambda}^{(t)})\right]^+ \tag{3.2b}$$

in each iteration t, where $\mathbf{g}(\boldsymbol{\lambda}^{(t)})$ is a subgradient of the dual function at $\boldsymbol{\lambda}^{(t)}$ and $[\cdot]^+$ denotes the projection onto the positive orthant. Note that the update of the primal variables (3.2a) can be performed in a distributed fashion by solving the local optimization problems for the given values of the dual variables. A subgradient of the dual function can be computed by evaluating the system-wide constraints for the primal variables $\mathbf{x}^{(t+1)}$ [YJ10], i.e.,

$$\mathbf{g}(\boldsymbol{\lambda}^{(t)}) \coloneqq \left(\sum_{i \in \mathcal{I}} \mathbf{A}_i \mathbf{x}_i^{(t+1)} - \mathbf{b} \right) \in \partial d(\boldsymbol{\lambda}^{(t)}). \tag{3.3}$$

It can be shown that (3.3) describes a subgradient of the dual function at $\boldsymbol{\lambda}^{(t)}$:

$$\begin{aligned}
&\mathbf{g}^T(\boldsymbol{\lambda}^{(t)})(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)}) + d(\boldsymbol{\lambda}^{(t)}) = \\
&\underbrace{\left(\sum_{i \in \mathcal{I}} \mathbf{A}_i \mathbf{x}_i^{(t+1)} - \mathbf{b} \right)^T}_{=\mathbf{g}^T(\boldsymbol{\lambda}^{(t)})} (\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)}) + \underbrace{\sum_{i \in \mathcal{I}} f_i(\mathbf{x}_i^{(t+1)}) + \boldsymbol{\lambda}^{(t),T} \left(\sum_{i \in \mathcal{I}} \mathbf{A}_i \mathbf{x}_i^{(t+1)} - \mathbf{b} \right)}_{=d(\boldsymbol{\lambda}^{(t)})} = \\
&\sum_{i \in \mathcal{I}} f_i(\mathbf{x}_i^{(t+1)}) + \boldsymbol{\lambda}^T \left(\sum_{i \in \mathcal{I}} \mathbf{A}_i \mathbf{x}_i^{(t+1)} - \mathbf{b} \right) \geq \min_{\mathbf{x}_i \in \mathcal{X}_i,\, \forall i \in \mathcal{I}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = d(\boldsymbol{\lambda})
\end{aligned} \tag{3.4}$$

The update step (3.2b) updates the dual variables in the direction of the subgradient of the dual. The step size parameter $\alpha^{(t)}$ plays an important role in the convergence of the algorithm. If it is chosen too large, the algorithm might diverge. If it is chosen too small, no substantial progress is made. The optimal step size can be defined utilizing the Lipschitz constant of the gradient of the dual function [Nes04]. However, this information is usually not available in a distributed optimization setting. For practical applications, the step size is adapted throughout the iterations [Ber99].

In this thesis, the condition that both the Euclidean norm (2-norm) of the primal residual $\|\mathbf{w}_p^{(t)}\|_2$ and of the dual residual $\|\mathbf{w}_d^{(t)}\|_2$ lie below pre-defined thresholds or that the maximum number of iterations is reached is used as the termination criterion, i.e.,

$$\left(\|\mathbf{w}_p^{(t)}\|_2 \leq \epsilon_p \wedge \|\mathbf{w}_d^{(t)}\|_2 \leq \epsilon_d \right) \vee (t = t_{\max}). \tag{3.5}$$

The primal residual indicates the satisfaction of the system-wide constraints. If these constraints (2.26b) are posed as inequalities, the primal residual is defined componentwise as

$$[\mathbf{w}_p^{(t)}]_l \coloneqq \max \left\{ \left[\sum_{i \in \mathcal{I}} \mathbf{A}_i \mathbf{x}_i^{(t)} - \mathbf{b} \right]_l, 0 \right\},\ l = 1, \ldots, n_{\mathbf{b}}. \tag{3.6}$$

If they are posed as equalities it is defined as

$$\mathbf{w}_p^{(t)} \coloneqq \sum_{i \in \mathcal{I}} \mathbf{A}_i \mathbf{x}_i^{(t)} - \mathbf{b}, \tag{3.7}$$

i.e., equal to the subgradient. The dual residual indicates the convergence of the dual variables to a stationary value and is defined as

$$\mathbf{w}_d^{(t)} \coloneqq \boldsymbol{\lambda}^{(t+1)} - \boldsymbol{\lambda}^{(t)}. \tag{3.8}$$

Algorithm 1 summarizes the subgradient method (SG). Note that steps 5–8 can be performed in a distributed manner, while steps 9–23 are performed by the coordinator.

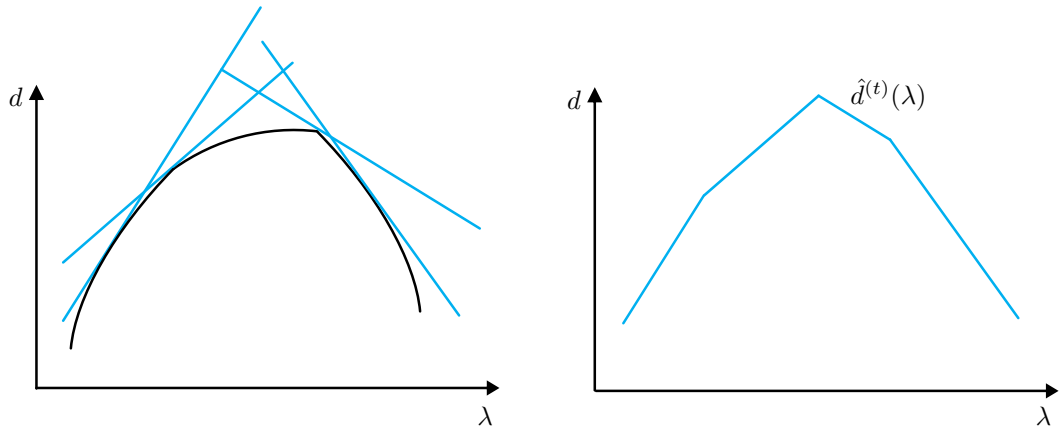
Algorithm 1 Subgradient Method (SG)

Require: $\boldsymbol{\lambda}^{(0)}$, $\alpha^{(0)}$, ϵ_p, ϵ_d, $t_{\max}$

$t \leftarrow 0$
repeat
  $t \leftarrow t + 1$
  Send $\boldsymbol{\lambda}^{(t)}$ to all subproblems
  for all $i = 1, \ldots, N_s$ **do**
    $\mathbf{x}_i^{(t+1)} \leftarrow \arg\min_{\mathbf{x}_i \in \mathcal{X}_i} \mathcal{L}_i(\mathbf{x}_i, \boldsymbol{\lambda}^{(t)})$
    Send $\mathbf{A}_i \mathbf{x}_i^{(t+1)}$ to the coordinator
  end for
  $\mathbf{g}(\boldsymbol{\lambda}^{(t)}) \leftarrow \sum_{i \in \mathcal{I}} \mathbf{A}_i \mathbf{x}_i^{(t+1)} - \mathbf{b}$
  if Constraints (2.26b) are inequalities **then**
    for all $l = 1, \ldots, n_{\mathbf{b}}$ **do**
      $[\mathbf{w}_p^{(t)}]_l \leftarrow \max\left\{ \left[\mathbf{g}(\boldsymbol{\lambda}^{(t)})\right]_l, 0 \right\}$
    end for
  else if Constraints (2.26b) are equalities **then**
    $\mathbf{w}_p^{(t)} \leftarrow \mathbf{g}(\boldsymbol{\lambda}^{(t)})$
  end if
  $\alpha^{(t)} \leftarrow \text{Update}(\alpha^{(t-1)})$
  if Constraints (2.26b) are inequalities **then**
    $\boldsymbol{\lambda}^{(t+1)} \leftarrow [\boldsymbol{\lambda}^{(t)} + \alpha^{(t)} \mathbf{g}(\boldsymbol{\lambda}^{(t)})]^+$
  else if Constraints (2.26b) are equalities **then**
    $\boldsymbol{\lambda}^{(t+1)} \leftarrow \boldsymbol{\lambda}^{(t)} + \alpha^{(t)} \mathbf{g}(\boldsymbol{\lambda}^{(t)})$
  end if
  $\mathbf{w}_d^{(t)} \leftarrow \boldsymbol{\lambda}^{(t+1)} - \boldsymbol{\lambda}^{(t)}$
until $(\|\mathbf{w}_p^{(t)}\|_2 \leq \epsilon_p \wedge \|\mathbf{w}_d^{(t)}\|_2 \leq \epsilon_d) \vee (t = t_{\max})$
return $\boldsymbol{\lambda}^{(t)}$

3.2 Bundle trust method

The subgradient method described in Sec. 3.1 only employs the subgradient of the previous iteration to update the dual variables. However, in contrast to the gradient, a subgradient does not necessarily provide an ascent direction for the dual function. This often leads to a slow rate of convergence. A generally more efficient class of algorithms is that of bundle methods [Mäk02]. Bundle methods are among the most efficient algorithms for nonsmooth optimization [BKM14]. As such they have been employed in the context of dual decomposition-based distributed optimization to solve the nonsmooth dual problem, e.g., for distributed model predictive control [PAL15] or for the coordination of energy networks [ZGG13]. Furthermore, bundle methods are also widely used in other areas where nonsmooth problems have to be solved, such as machine learning, where nonsmoothness is often encountered due to regularization terms [LSV07]. In the following, the bundle trust method (BTM) according to Bagirov et al. [BKM14], as described by Yfantis and Ruskowski in [YR22], is presented.



(a) Dual function and cutting planes defined by its subgradients.

(b) Resulting cutting plane model.

Figure 3.2: Illustration of the cutting plane model (from [YWW+23]).

The idea of bundle methods is to employ subgradient information collected from multiple previous iterations to construct a piece-wise linear over-approximator, a so-called cutting plane model, of the nonsmooth concave dual function $d(\boldsymbol{\lambda})$. To this end, the data

$$\mathcal{B}^{(t)} = \{(\boldsymbol{\lambda}^{(j)}, d(\boldsymbol{\lambda}^{(j)}), \mathbf{g}(\boldsymbol{\lambda}^{(j)})) \in \mathbb{R}^{n_\mathfrak{b}} \times \mathbb{R} \times \mathbb{R}^{n_\mathfrak{b}} |\ 1 \leq j \leq t\} \tag{3.9}$$

is stored in each iteration. $\mathcal{B}$ is referred to as a bundle. As shown in the previous section, the hyperplane defined by the subgradient is an over-approximator of its corresponding function. The cutting plane model $\hat{d}^{(t)}(\boldsymbol{\lambda})$ of the dual function in iteration t is defined as

$$\hat{d}^{(t)}(\boldsymbol{\lambda}) \coloneqq \min_{j \in \mathcal{J}^{(t)}} \{d(\boldsymbol{\lambda}^{(j)}) + \mathbf{g}^T(\boldsymbol{\lambda}^{(j)})(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(j)})\}, \tag{3.10}$$

where $\mathcal{J}^{(t)} \subseteq \{1, \ldots, t\}$ denotes the subset of the used data points. As storing the dual variables, dual values, and subgradients for all past iterations might require a significant storage memory, only the bundle information up to a certain iteration age τ is stored,

$$\mathcal{J}^{(t)} \coloneqq \{\max\{1, t - \tau + 1\}, \ldots, t\}. \tag{3.11}$$

Fig. 3.2 illustrates the cutting plane model for a nonsmooth dual function. The approximation can be written in an equivalent form as

$$\hat{d}^{(t)}(\boldsymbol{\lambda}) = \min_{j \in \mathcal{J}^{(t)}} \{d(\boldsymbol{\lambda}^{(t)}) + \mathbf{g}^T(\boldsymbol{\lambda}^{(j)})(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)}) - \beta^{(j,t)}\}, \tag{3.12}$$

with the linearization error

$$\beta^{(j,t)} = d(\boldsymbol{\lambda}^{(t)}) - d(\boldsymbol{\lambda}^{(j)}) - \mathbf{g}^T(\boldsymbol{\lambda}^{(j)})(\boldsymbol{\lambda}^{(t)} - \boldsymbol{\lambda}^{(j)}),\ \forall j \in \mathcal{J}^{(t)}. \tag{3.13}$$

The bundle trust method computes a search direction $\mathbf{s}^{(t)}$ in each iteration, by solving the direction finding problem

$$\max_{\mathbf{s}\in\mathbb{R}^{n_\mathbf{b}}} \hat{d}^{(t)}(\boldsymbol{\lambda}^{(t)}+\mathbf{s}), \tag{3.14a}$$

$$\text{s. t. } \|\mathbf{s}\|_2^2 \leq \alpha^{(t)}, \tag{3.14b}$$

$$\boldsymbol{\lambda}^{(t)}+\mathbf{s} \geq \mathbf{0}. \tag{3.14c}$$

The constraint (3.14b) represents a trust region, preventing too aggressive update steps. Throughout this thesis the step size parameter of the subgradient method is also used to define this trust region to provide a better basis for comparison. The constraint (3.14c) ensures the feasibility of the updated dual variables and can be omitted if the system-wide constraints (2.26b) are equalities. It replaces the projection onto the feasible set used in (3.2b). Problem (3.14) is still nonsmooth and can be rewritten as a smooth quadratic direction-finding problem

$$\max_{v\in\mathbb{R},\, \mathbf{s}\in\mathbb{R}^{n_\mathbf{b}}} v, \tag{3.15a}$$

$$\text{s. t. } \|\mathbf{s}\|_2^2 \leq \alpha^{(t)}, \tag{3.15b}$$

$$\mathbf{g}^T(\boldsymbol{\lambda}^{(j)})\mathbf{s} - \beta^{(j,t)} \geq v,\ \forall j \in \mathcal{J}^{(t)}, \tag{3.15c}$$

$$\boldsymbol{\lambda}^{(t)}+\mathbf{s} \geq \mathbf{0}. \tag{3.15d}$$

To summarize, the bundle trust method (BTM) updates the primal and dual variables in each iteration according to

$$\forall i \in \mathcal{I},\, \mathbf{x}_i^{(t+1)} = \arg\min_{\mathbf{x}_i\in\mathcal{X}_i} \mathcal{L}_i(\mathbf{x}_i, \boldsymbol{\lambda}^{(t)}), \tag{3.16a}$$

$$\mathbf{s}^{(t)} = \arg\max_{v\in\mathbb{R},\, \mathbf{s}\in\mathbb{R}^{n_\mathbf{b}}} v, \tag{3.16b}$$

$$\text{s.t. } (3.15\text{b}) - (3.15\text{d}),$$

$$\boldsymbol{\lambda}^{(t+1)} = \boldsymbol{\lambda}^{(t)} + \mathbf{s}^{(t)}. \tag{3.16c}$$

Bundle methods often employ a null step, i.e., $\boldsymbol{\lambda}^{(t+1)} = \boldsymbol{\lambda}^{(t)}$, to compute a new subgradient at the current iterate and to improve the approximation. In the case of dual decomposition-based distributed optimization, the same subgradient would be obtained after a null step. The dual variables are therefore updated according to (3.16c) in each iteration. Note that in the case of BTM in addition to the contributions to the subgradient $\mathbf{g}_i(\boldsymbol{\lambda}^{(t)})$ the subproblems have to also communicate their contribution to the dual function

$$d_i(\boldsymbol{\lambda}^{(t)}) = f_i(\mathbf{x}_i^{(t+1)}) + \boldsymbol{\lambda}^{(t),T}\mathbf{A}_i\mathbf{x}_i^{(t+1)} = \mathcal{L}_i(\mathbf{x}_i^{(t+1)}, \boldsymbol{\lambda}^{(t)}). \tag{3.17}$$

Algorithm 2 summarizes the bundle trust method (BTM).

3.3 Alternating direction method of multipliers

Another approach to solving the nonsmooth dual problem is to smoothen the problem by convexifying the Lagrange function. This is used in augmented Lagrangian methods

Algorithm 2 Bundle Trust Method (BTM)

Require: $\boldsymbol{\lambda}^{(0)}$, $\alpha^{(0)}$, τ, ϵ_p, ϵ_d, $t_{\max}$

$t \leftarrow 0$
$\mathcal{B}^{(0)} \leftarrow \{\}$
repeat
  $t \leftarrow t+1$
  Send $\boldsymbol{\lambda}^{(t)}$ to all subproblems
  for all $i = 1, \ldots, N_s$ **do**
    $\mathbf{x}_i^{(t+1)} \leftarrow \arg\min_{\mathbf{x}_i \in \mathcal{X}_i} \mathcal{L}_i(\mathbf{x}_i, \boldsymbol{\lambda}^{(t)})$
    Send $\mathbf{A}_i \mathbf{x}_i^{(t+1)}$ and $\mathcal{L}_i(\mathbf{x}_i^{(t+1)}, \boldsymbol{\lambda}^{(t)})$ to the coordinator
  end for
  $\mathbf{g}(\boldsymbol{\lambda}^{(t)}) \leftarrow \sum_{i \in \mathcal{I}} \mathbf{A}_i \mathbf{x}_i^{(t+1)} - \mathbf{b}$
  $d(\boldsymbol{\lambda}^{(t)}) \leftarrow \sum_{i \in \mathcal{I}} \mathcal{L}_i(\mathbf{x}_i^{(t+1)}, \boldsymbol{\lambda}^{(t)}) - \boldsymbol{\lambda}^{(t),T} \mathbf{b}$
  $\mathcal{B}^{(t)} \leftarrow \mathcal{B}^{(t)} \cup \{(\boldsymbol{\lambda}^{(t)}, d(\boldsymbol{\lambda}^{(t)}), \mathbf{g}(\boldsymbol{\lambda}^{(t)}))\}$
  if Constraints (2.26b) are inequalities **then**
    for all $l = 1, \ldots, n_{\mathbf{b}}$ **do**
      $[\mathbf{w}_p^{(t)}]_l \leftarrow \max\left\{ \left[\mathbf{g}(\boldsymbol{\lambda}^{(t)})\right]_l, 0 \right\}$
    end for
  else if Constraints (2.26b) are equalities **then**
    $\mathbf{w}_p^{(t)} \leftarrow \mathbf{g}(\boldsymbol{\lambda}^{(t)})$
  end if
  $\alpha^{(t)} \leftarrow \text{Update}(\alpha^{(t-1)})$
  $\mathcal{J}^{(t)} \leftarrow \{\max\{1, t-\tau+1\}, \ldots, t\}$
  if Constraints (2.26b) are inequalities **then**
    $\mathbf{s}^{(t)} \leftarrow \arg \max_{v \in \mathbb{R},\, \mathbf{s} \in \mathbb{R}^{n_{\mathbf{b}}}} v$, s. t. (3.15b)-(3.15d)
  else if Constraints (2.26b) are equalities **then**
    $\mathbf{s}^{(t)} \leftarrow \arg \max_{v \in \mathbb{R},\, \mathbf{s} \in \mathbb{R}^{n_{\mathbf{b}}}} v$, s. t. (3.15b), (3.15c)
  end if
  $\boldsymbol{\lambda}^{(t+1)} \leftarrow \boldsymbol{\lambda}^{(t)} + \mathbf{s}^{(t)}$
  $\mathbf{w}_d^{(t)} \leftarrow \boldsymbol{\lambda}^{(t+1)} - \boldsymbol{\lambda}^{(t)}$
until $(\|\mathbf{w}_p^{(t)}\|_2 \leq \epsilon_p \wedge \|\mathbf{w}_d^{(t)}\|_2 \leq \epsilon_d) \vee (t = t_{\max})$
return $\boldsymbol{\lambda}^{(t)}$

[AHU58]. The issue with augmented Lagrangian methods is that separability is lost. The alternating direction method of multipliers (ADMM) is an extension of the augmented Lagrangian methods, whereby separability is maintained. It was first introduced by Gabay and Mercier [GM76] and Fortin and Glowinski [FG83] where it was applied to find the solution of differential equations. It has gained significant attention in recent years since it was popularized by Boyd et al. [BPC+11]. In its standard form, the ADMM algorithm solves problems of the form

$$\min_{\mathbf{x}_1 \in \mathbb{R}^{n_{\mathbf{x}_1}}, \mathbf{x}_2 \in \mathbb{R}^{n_{\mathbf{x}_2}}} \; f_1(\mathbf{x}_1) + f_1(\mathbf{x}_2) \tag{3.18a}$$

$$\text{s. t. } \mathbf{A}_1\mathbf{x}_1 + \mathbf{A}_2\mathbf{x}_2 = \mathbf{b}, \tag{3.18b}$$

by defining an augmented Lagrange function

$$\hat{\mathcal{L}}_\rho(\mathbf{x}_1, \mathbf{x}_2, \boldsymbol{\lambda}) \coloneqq f_1(\mathbf{x}_1) + f_1(\mathbf{x}_2) + \boldsymbol{\lambda}^T(\mathbf{A}_1\mathbf{x}_1 + \mathbf{A}_2\mathbf{x}_2 - \mathbf{b}) + \frac{\rho}{2}\|\mathbf{A}_1\mathbf{x}_1 + \mathbf{A}_2\mathbf{x}_2 - \mathbf{b}\|_2. \tag{3.19}$$

The last term in the augmented Lagrange function (3.19) is a regularization term that results in a convexification of the optimization problem. The primal variables are then updated in an alternating manner according to

$$\mathbf{x}_1^{(t+1)} = \arg\min_{\mathbf{x}_1 \in \mathbb{R}^{n_{\mathbf{x}_1}}} \hat{\mathcal{L}}_\rho(\mathbf{x}_1, \mathbf{x}_2^{(t)}, \boldsymbol{\lambda}^{(t)}), \tag{3.20a}$$

$$\mathbf{x}_2^{(t+1)} = \arg\min_{\mathbf{x}_2 \in \mathbb{R}^{n_{\mathbf{x}_2}}} \hat{\mathcal{L}}_\rho(\mathbf{x}_1^{(t+1)}, \mathbf{x}_2, \boldsymbol{\lambda}^{(t)}), \tag{3.20b}$$

$$\boldsymbol{\lambda}^{(t+1)} = \boldsymbol{\lambda}^{(t)} + \rho(\mathbf{A}_1\mathbf{x}_1^{(t+1)} + \mathbf{A}_2\mathbf{x}_2^{(t+1)} - \mathbf{b}). \tag{3.20c}$$

The update of the primal variables (3.20a) and (3.20b) cannot be performed in parallel. In this thesis, problems where multiple subsystems are connected through shared limited resources are considered (2.26). This case suits the use of the optimal exchange version of ADMM, as described in [BPC+11] (Sec. 7.3.2) and [Wen20] (Sec. 3.5.4). This formulation relies on the introduction of auxiliary variables $\mathbf{z}_i \in \mathbb{R}^{n_{\mathbf{b}}}$, which can be interpreted as a feasible resource utilization for subproblem i, similar to primal decomposition (cf. Sec. 2.2.3). Using the auxiliary variables problem (2.26) can be reformulated,

$$\min_{\mathbf{x}_1, \ldots, \mathbf{x}_{N_s}} \sum_{i \in \mathcal{I}} f_i(\mathbf{x}_i), \tag{3.21a}$$

$$\text{s. t. } \mathbf{A}_i\mathbf{x}_i \leq \mathbf{z}_i, \; \forall i \in \mathcal{I}, \tag{3.21b}$$

$$\sum_{i \in \mathcal{I}} \mathbf{z}_i = \mathbf{b}, \tag{3.21c}$$

$$\mathbf{x}_i \in \mathcal{X}_i, \; \forall i \in \mathcal{I}. \tag{3.21d}$$

The individual augmented Lagrange functions for problem (3.21) are defined as,

$$\hat{\mathcal{L}}_{\rho,i}(\mathbf{x}_i, \boldsymbol{\lambda}, \mathbf{z}_i) \coloneqq f_i(\mathbf{x}_i) + \boldsymbol{\lambda}^T(\mathbf{A}_i\mathbf{x}_i - \mathbf{z}_i) + \frac{\rho}{2}\left\|\mathbf{A}_i\mathbf{x}_i - \mathbf{z}_i\right\|_2^2. \tag{3.22}$$

In each iteration t, the primal, auxiliary, and dual variables are updated according to

$$\forall i \in \mathcal{I}, \; \mathbf{x}_i^{(t+1)} = \arg\min_{\mathbf{x}_i \in \mathcal{X}_i} \hat{\mathcal{L}}_{\rho,i}(\mathbf{x}_i, \boldsymbol{\lambda}^{(t)}, \mathbf{z}_i^{(t)}) \tag{3.23a}$$

$$\forall i \in \mathcal{I}, \; \mathbf{z}_i^{(t+1)} = \mathbf{A}_i\mathbf{x}_i^{(t+1)} - \text{ave}(\mathbf{A}\mathbf{x}^{(t+1)} - \mathbf{b}) \tag{3.23b}$$

$$\boldsymbol{\lambda}^{(t+1)} = [\boldsymbol{\lambda}^{(t)} + \rho^{(t)}\text{ave}(\mathbf{A}\mathbf{x}^{(t+1)} - \mathbf{b})]^+ \tag{3.23c}$$

where

$$\text{ave}(\mathbf{A}\mathbf{x}^{(t+1)} - \mathbf{b}) \coloneqq \frac{1}{N_s}\sum_{i \in \mathcal{I}} \left(\mathbf{A}_i\mathbf{x}_i^{(t+1)} - \mathbf{b}\right) \tag{3.24}$$

denotes the average of the system-wide constraints. Note that the update of the primal variables (3.23a) can now be performed in parallel. The update step of the auxiliary variables (3.23b) ensures the satisfaction of (3.21c) [Wen20],

$$\sum_{i\in\mathcal{I}} \mathbf{z}_i^{(t+1)} = \sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i^{(t+1)} - \sum_{i\in\mathcal{I}} \text{ave}(\mathbf{A}\mathbf{x}^{(t+1)} - \mathbf{b}) \tag{3.25a}$$

$$= \sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i^{(t+1)} - N_s\text{ave}(\mathbf{A}\mathbf{x}^{(t+1)} - \mathbf{b}) \tag{3.25b}$$

$$= \sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i^{(t+1)} - \sum_{i\in\mathcal{I}} \left(\mathbf{A}_i\mathbf{x}_i^{(t+1)} - \mathbf{b}\right) \tag{3.25c}$$

$$= \mathbf{b}. \tag{3.25d}$$

ADMM can be interpreted as a proximal algorithm. These algorithms employ the proximal operator.

Definition 18: Proximal operator [PB14]

The (scaled) proximal operator is defined as

$$\mathbf{x} = \text{prox}_{\rho,f}(\boldsymbol{\chi}) \coloneqq \arg\min_{\mathbf{y}} f(\mathbf{y}) + \frac{1}{2\rho}\|\mathbf{y} - \boldsymbol{\chi}\|_2^2. \tag{3.26}$$

For a better interpretation of ADMM, [Wen20] defined a modified scaled proximal operator as

$$\mathbf{x} = \text{prox}'_{\rho,f}(\boldsymbol{\chi}) \coloneqq \arg\min_{\mathbf{y}} f(\mathbf{y}) + \frac{1}{2\rho}\|\mathbf{L}(\mathbf{y}) - \boldsymbol{\chi}\|_2^2, \tag{3.27}$$

which differs only in the first term of the regularization term, where a linear mapping $\mathbf{L}(\mathbf{y})$ is used. Using the definition (3.27) and omitting the constant term $-\boldsymbol{\lambda}^{(t),T}\mathbf{z}_i$ the update of the primal variables (3.23a) can be rewritten as

$$\forall i \in \mathcal{I},\ \mathbf{x}_i^{(t+1)} = \arg\min_{\mathbf{x}_i\in\mathcal{X}_i} f(\mathbf{x}_i) + \lambda^{(\mathbf{t}),\mathbf{T}}\mathbf{A}_i\mathbf{x}_i + \frac{\rho}{2}\|\mathbf{A}_i\mathbf{x}_i - \mathbf{z}_i^{(t)}\|_2^2 \tag{3.28a}$$

$$= \arg\min_{\mathbf{x}_i\in\mathcal{X}_i} \mathcal{L}_i(\mathbf{x}_i, \boldsymbol{\lambda}^{(t)}) + \frac{\rho}{2}\|\mathbf{A}_i\mathbf{x}_i - \mathbf{z}_i^{(t)}\|_2^2 \tag{3.28b}$$

$$= \text{prox}'_{1/\rho,\mathcal{L}_i}(\mathbf{z}_i^{(t)}). \tag{3.28c}$$

Using eq. (3.28) the update step of ADMM can be interpreted as a proximal mapping onto a feasible resource utilization.

Note that while the dual variables $\boldsymbol{\lambda}$ are still common among the subsystems, the auxiliary variables $\mathbf{z}_i$ are formulated for each subproblem individually. The update of the auxiliary variables is performed on the coordinator level.

ADMM is an efficient algorithm for dual decomposition-based distributed optimization and it outperforms other algorithms for a variety of benchmark problems [KCD15]. It converges under milder assumptions than the subgradient method for convex primal prob-

lems [BPC+11]. While convergence can be proven for constant values of the regularization parameter ρ, a variation of the parameter throughout the iterations works well in practice. In this work, the adaptation strategy reported in [HYW00] and [WL01] is employed:

$$\rho^{(t+1)} = \begin{cases} \tau^{\text{incr}}\rho^{(t)}, & \text{if } \|\mathbf{w}_p^{(t)}\|_2 > \mu\|\mathbf{w}_d^{(t)}\|_2, \\ \rho^{(t)}/\tau^{\text{decr}}, & \text{if } \|\mathbf{w}_d^{(t)}\|_2 > \mu\|\mathbf{w}_p^{(t)}\|_2, \\ \rho^{(t)}, & \text{otherwise.} \end{cases} \tag{3.29}$$

The parameters $\mu, \tau^{\text{incr}}, \tau^{\text{decr}} > 1$ are tuning parameters. In contrast to the subgradient method and the bundle trust method, the dual residual in ADMM is defined as

$$\mathbf{w}_d^{(t)} := \mathbf{z}^{(t+1)} - \mathbf{z}^{(t)}, \tag{3.30}$$

where $\mathbf{z} = [\mathbf{z}_1^T, \ldots, \mathbf{z}_{N_s}^T]^T \in \mathbb{R}^{n_\mathbf{b} \cdot N_s}$ denotes the collection of the auxiliary variables.

ADMM leads to a slightly increased communication overhead between the subproblems and the coordinator, as the auxiliary variables have to be communicated to each subproblem as well. Furthermore, the coordinator has to know the contribution to the coupling constraints of each subproblem to update the auxiliary variables (3.23b), whereas the subgradient method and BTM only require the knowledge of the aggregated value of the coupling constraints. In the context of distributed optimization of interacting autonomous units, a drawback is also that the structure of the subproblems is altered due to the addition of the regularization term, which makes the objective functions lose their original meaning as, e.g., the local profit. The ADMM algorithm is summarized in Algorithm 3. Steps 5–8 are performed in parallel by the subproblems while steps 9–25 are performed by the coordinator.

3.4 Other related dual decomposition-based methods

Several algorithms have been proposed that aim at improving the performance of the subgradient method. The only parameter that can be tuned in the subgradient method is the step size. Nedić and Bertsekas [NB01] provide several dynamic step size adjustment strategies. In contrast to the gradient, the subgradient does not always provide an ascent direction for the dual variables [Mäk02]. Bragin et al. [BLY+15] address this via the surrogate Lagrangian relaxation (SLR) method, also providing convergence proofs. Instead of updating the dual variables in the direction of the subgradient, a surrogate subgradient direction that forms an acute angle towards the optimal dual values is used. The algorithm was extended in [BLY+19] by introducing an additional absolute value penalty in the objective function of the primal problem (surrogate absolute value Lagrangian relaxation, SAVLR) and in [LLB+20] by employing ordinal optimization. These methods were used to solve mixed-integer linear programming (MILP) problems with the main concern being computational scalability. However, the recovery of a primal feasible solution is

Algorithm 3 Alternating Direction Method of Multipliers (ADMM)

Require: $\boldsymbol{\lambda}^{(0)}$, $\mathbf{z}^{(0)}$, $\rho^{(0)}$, μ, τ_{incr}, τ_{decr}, ϵ_p, ϵ_d, $t_{\max}$

$t \leftarrow 0$
repeat
  $t \leftarrow t + 1$
  Send $\boldsymbol{\lambda}^{(t)}$, $\mathbf{z}_i^{(t)}$ and $\rho^{(t)}$ to all subproblems
  for all $i = 1, \ldots, N_s$ **do**
    $\mathbf{x}_i^{(t+1)} \leftarrow \text{prox}'_{1/\rho, \mathcal{L}}(\mathbf{z}_i^{(t)})$
    Send $\mathbf{A}_i \mathbf{x}_i^{(t+1)}$ to the coordinator
  end for
  for all $i = 1, \ldots, N_s$ **do**
    $\mathbf{z}_i^{(t+1)} \leftarrow \mathbf{A}_i \mathbf{x}_i^{(t+1)} - \text{ave}(\mathbf{A}\mathbf{x}^{(t+1)} - \mathbf{b})$
  end for
  if Constraints (2.26b) are inequalities **then**
    for all $l = 1, \ldots, n_{\mathbf{b}}$ **do**
      $[\mathbf{w}_p^{(t)}]_l \leftarrow \max\left\{\left[\sum_{i \in \mathcal{I}} \mathbf{A}_i \mathbf{x}_i^{(t+1)} - \mathbf{b}\right]_l, 0\right\}$
    end for
  else if Constraints (2.26b) are equalities **then**
    $\mathbf{w}_p^{(t)} \leftarrow \sum_{i \in \mathcal{I}} \mathbf{A}_i \mathbf{x}_i^{(t+1)} - \mathbf{b}$
  end if
  if Constraints (2.26b) are inequalities **then**
    $\boldsymbol{\lambda}^{(t+1)} \leftarrow [\boldsymbol{\lambda}^{(t)} + \rho^{(t)} \text{ave}(\mathbf{A}\mathbf{x}^{(t+1)} - \mathbf{b})]^+$
  else if Constraints (2.26b) are equalities **then**
    $\boldsymbol{\lambda}^{(t+1)} \leftarrow \boldsymbol{\lambda}^{(t)} + \rho^{(t)} \text{ave}(\mathbf{A}\mathbf{x}^{(t+1)} - \mathbf{b})$
  end if
  $\mathbf{w}_d^{(t)} \leftarrow \mathbf{z}^{(t+1)} - \boldsymbol{z}^{(t)}$
  $\rho^{(t+1)} \leftarrow$ Update(3.29)
until $(\|\mathbf{w}_p^{(t)}\|_2 \leq \epsilon_p \wedge \|\mathbf{w}_d^{(t)}\|_2 \leq \epsilon_d) \vee (t = t_{\max})$
return $\boldsymbol{\lambda}^{(t)}$

not explicitly addressed. The algorithms based on SLR do not require the solution of each subproblem in each iteration. This generally results in a larger number of iterations, each of which requires less computation time. Therefore these algorithms are suitable for problems where the solution of the subproblems poses the main bottleneck. A common approach to improve the rate of convergence for gradient-based algorithms is to use acceleration methods.

Uribe et al. [ULG+20] adapt the fast gradient method (FGM) proposed by Nesterov [Nes83] and apply it to distributed optimization over networks. They study different problem classes for the subproblems, where the degree of convexity and smoothness is varied. However, the system-wide problem is a consensus problem, where no individual or system-wide constraints are considered. Similar consensus problems are considered in Eisen et al. [EMR16], where consensus constraints are introduced and subsequently

relaxed through dual decomposition. The authors propose a quasi-Newton method that relies on decentralized computations to update the approximated Hessians locally. The same algorithm is further studied in [EMR17], where it is directly applied to the primal problem without introducing dual variables. Nevertheless, no constraints are considered, except for consensus constraints, and direct neighbor communication with an exchange of gradients is necessary in each iteration.

Zargham et al. [ZRO+14] locally approximate the inverse of the Hessian matrices by allowing direct communication between the subproblems. Notarnicola and Notarstefano [NN20] allow communication of auxiliary variables between the subproblems and employ a relaxation and successive distributed decomposition (RSDD) approach. Direct communication between the subproblems is also not intended in this work.

The aim of dual decomposition-based distributed optimization algorithms usually is to find a set of primal-dual variables that satisfy the Karush-Kuhn-Tucker (KKT) conditions. The Lagrange multiplier method and the KKT conditions are generalized by Li [Li18] to a wider class of functions that still satisfy the strong duality condition. These generalizations are subsequently applied to distributed optimization.

Goldstein et al. [GOS+14] extend ADMM to improve its rate of convergence by employing a predictor-corrector–type acceleration step. However, this step is only stable for strongly convex problems. To improve the rate of convergence second-order information can be exploited. Houska et al. [HFD16] extend ADMM into the augmented Lagrangian-based alternating direction inexact Newton method (ALADIN). In this approach, the Hessians of the Lagrangians of the subproblems are approximated, which requires the communication of the constraint Jacobian matrices. This kind of second-order information is assumed to be inaccessible in this thesis.

Chatzipanagiotis et al. [CDZ15] introduce an acceleration step into the augmented Lagrangian method and propose the accelerated distributed augmented Lagrangian method (ADAL), which is used to solve distributed convex problems. The convergence of ADAL for nonconvex problems is further studied in [CZ17]. ADMM is also generalized to nonconvex problems by Li in [LFT+20], where nonconvex equality constraints are considered.

As discussed in Sec. 3.3, ADMM belongs to the class of proximal algorithms. Other proximal algorithms can also be applied to distributed optimization, e.g., the Douglas-Rachford splitting method [HH16]. These methods rely on the addition of a penalty term to the objective function. A similar idea forms the basis of interior point methods, where a barrier term is added to the objective function to handle constraints [WB06]. Necoara and Suykens [NS09] combine dual decomposition with interior point methods by adding self-concordant barrier terms to the Lagrange function.

Maxeiner and Engell [ME20] propose an approximation of the dual function by performing

subgradient update steps with a constant step size and then using the collected data to extrapolate the update steps toward the optimal dual variables. This extrapolation is based on the analytic solution of the dual problem for unconstrained quadratic programs and requires adjustments if individual constraints are added or if other problem classes are considered.

4 Algorithms Based on Smooth Approximations

Dual decomposition-based distributed optimization can be interpreted as a blackbox optimization approach since it aims to maximize the dual function whose closed analytical form is unknown. In each iteration the distributed optimization algorithms can query the dual function and obtain its value and a subgradient. One of the main concepts in derivative-free optimization, a field dealing with blackbox optimization problems, is that of a surrogate function obtained by sampling the original unknown objective function [CSV09]. The previously presented bundle methods follow this idea by constructing a piece-wise linear function as a surrogate of the dual function from previously sampled objective values and subgradients. However, the resulting cutting plane model (3.10) is still nonsmooth. This chapter presents three different algorithms that rely on the computation of a smooth surrogate function $\psi^{(t)}(\boldsymbol{\lambda})$. The parameters of the surrogate function are obtained by minimizing a loss function depending on previously collected information $\mathcal{B}^{(t)}$,

$$\psi^{(t)}(\boldsymbol{\lambda}) \coloneqq \arg \min_{\psi\colon \mathbb{R}^{n_{\mathbf{b}}} \to \mathbb{R}} \sum_{j \in \mathcal{J}^{(t)}} L(\psi(\boldsymbol{\lambda}^{(j)}), \mathcal{B}^{(t)}). \tag{4.1}$$

Once the surrogate function is obtained, the dual variables are updated by solving an optimization problem, subject to constraints on the dual variables,

$$\boldsymbol{\lambda}^{(t+1)} = \arg \min_{\boldsymbol{\lambda} \in \mathcal{M}} \psi^{(t)}(\boldsymbol{\lambda}). \tag{4.2}$$

First, two algorithms based on the solution of a regression problem are presented. These are the quadratic approximation coordination (QAC) algorithm presented in [WRE20] and the new quadratically approximated dual ascent (QADA) algorithm. The algorithms share some components, in particular, the strategy to select regression data and the constraints on the step size. After the introduction of these components, the QAC and QADA algorithms are presented. The QAC algorithm was proposed by Wenzel [WRE20, Wen20] and approximates the squared Euclidean norm of the primal residual $\|\mathbf{w}_p(\boldsymbol{\lambda})\|_2^2$ by a quadratic function. The QADA algorithm, first presented in [YGW+22], expands upon the ideas of the QAC algorithm and approximates the dual function $d(\boldsymbol{\lambda})$, which is advantageous for several reasons, as explained below. Furthermore, bundle information and cutting planes are used to handle the nonsmoothness of the dual function, which cannot be used within the QAC algorithm. Finally, this thesis also deals with nonconvex problems in the form of

mixed-integer programs. In these cases, the value of the dual function, which is not used or sampled in the QAC algorithm, is essential to assess the quality of a feasible primal solution via the duality gap (2.34).

In addition to the regression-based approximation of the dual function, an algorithm based on quasi-Newton updates is presented. The quasi-Newton dual ascent (QNDA) algorithm, first presented in [YR22], also approximates the dual function as a quadratic function. However, the approximation of the Hessian is based on Broyden-Fletcher-Goldfarb-Shanno (BFGS) updates. This addresses the issue of the quadratically growing size of the regression data set of the QAC and QADA algorithms.

A discussion of the convergence of the proposed algorithms is provided at the end of the section. The contents of this chapter have been published in [YWW+23].

4.1 Regression-based approximations

This section presents the algorithms in which the surrogate function is obtained as the solution to a regression problem. First, the underlying regression problem is introduced, followed by a description of the regression data selection strategy. As the quadratic approximations are only valid locally, a trust region constraint based on the used regression data is presented, which prevents the dual variables from moving too far away from the range of validity of the surrogate function. Afterward, the QAC algorithm is summarized briefly and the QADA algorithm is presented.

4.1.1 Fitting the parameters of a quadratic model

The regression-based algorithms follow the basic idea of derivative-free optimization according to Conn et al. [CSV09], where locally a surrogate, in this case quadratic, model is fitted to previously collected data. To this end, a set of data points,

$$\mathcal{D}^{(t)} = \{(\boldsymbol{\lambda}^{(j)}, \hat{\psi}(\boldsymbol{\lambda}^{(j)}))|\ 1 \leq j \leq t\} \tag{4.3}$$

collected from previous iterations is chosen, where $\boldsymbol{\lambda}^{(j)}$ is a value of the dual variables and $\hat{\psi}(\boldsymbol{\lambda}^{(j)})$ the corresponding observed value of the approximated function. The surrogate function considered in this thesis is a quadratic function of the form

$$\psi^{(t)}(\boldsymbol{\lambda}) := \frac{1}{2}\boldsymbol{\lambda}^T\mathbf{Q}^{(t)}\boldsymbol{\lambda} + \mathbf{q}^{(t),T}\boldsymbol{\lambda} + q_0^{(t)},\ \mathbf{Q}^{(t)} \in S\mathbb{R}^{n_\mathfrak{b}\times n_\mathfrak{b}}, \mathbf{q}^{(t)} \in \mathbb{R}^{n_\mathfrak{b}}, q_0^{(t)} \in \mathbb{R}. \tag{4.4}$$

The parameters of the quadratic model (4.4) can be computed as the solution to the regression problem

$$\mathbf{Q}^{(t)}, \mathbf{q}^{(t)}, q_0^{(t)} = \arg\min_{\mathbf{Q},\mathbf{q},q_0} \sum_{j\in\mathcal{J}^{(t)}} \|\psi^{(t)}(\boldsymbol{\lambda}^{(j)}) - \hat{\psi}(\boldsymbol{\lambda}^{(j)})\|_2^2. \tag{4.5}$$

To obtain the solution (4.5) in a closed form, eq. (4.4) can be rewritten as

$$\psi^{(t)}(\boldsymbol{\lambda}) = \sum_{l=1}^{n_\mathbf{b}} \sum_{k=1}^{n_\mathbf{b}} [\mathbf{Q}^{(t)}]_{l,k} [\boldsymbol{\lambda}]_l [\boldsymbol{\lambda}]_k + \sum_{l=1}^{n_\mathbf{b}} [\mathbf{q}^{(t)}]_l [\boldsymbol{\lambda}]_l + q_0^{(t)}. \tag{4.6}$$

The parameters of the surrogate function can be summarized in a vector $\mathbf{p}^{(t)}$,

$$\mathbf{p}^{(t),T} \coloneqq [[\mathbf{Q}^{(t)}]_{1,1}, \ldots, [\mathbf{Q}^{(t)}]_{1,n_\mathbf{b}}, [\mathbf{Q}^{(t)}]_{2,2}, \ldots, [\mathbf{Q}^{(t)}]_{n_\mathbf{b},n_\mathbf{b}}, [\mathbf{q}^{(t)}]_1, \ldots, [\mathbf{q}^{(t)}]_{n_\mathbf{b}}, q_0^{(t)}]. \tag{4.7}$$

Let $n_j^{(t)} \coloneqq |\mathcal{J}^{(t)}|$ be the number of used regression points in iteration t and

$$\hat{\boldsymbol{\psi}}^{(t)} \coloneqq [\hat{\psi}(\boldsymbol{\lambda}^{(1)}), \ldots, \hat{\psi}(\boldsymbol{\lambda}^{(n_j^{(t)})})]^T \tag{4.8}$$

the vector of observed values of the approximated function. Then, by defining the Vandermonde-matrix [Wen20]

$$\mathbf{M}^{(t)} \coloneqq \left[\mathbf{l}_1^{\circ 2}, \mathbf{l}_1 \circ \mathbf{l}_2, \ldots, \mathbf{l}_1 \circ \mathbf{l}_{n_\mathbf{b}}, \mathbf{l}_2^{\circ 2}, \ldots, \mathbf{l}_{n_\mathbf{b}}^{\circ 2}, \mathbf{l}_1 \ldots, \mathbf{l}_{n_\mathbf{b}}, \mathbf{1}\right], \tag{4.9}$$

with $\mathbf{l}_l \coloneqq [[\boldsymbol{\lambda}^{(1)}]_l, \ldots, [\boldsymbol{\lambda}^{(n_j^{(t)})}]_l]^T$ and the element-wise vector multiplication $\circ$, the parameters of the surrogate function can be obtained as

$$\mathbf{p}^{(t)} = \left(\mathbf{M}^{(t),T} \mathbf{M}^{(t)}\right)^{-1} \mathbf{M}^{(t)} \hat{\boldsymbol{\psi}}^{(t)}. \tag{4.10}$$

Note that $\boldsymbol{\lambda}^{(1)}$ in equations (4.8) and (4.9) denotes the first dual variables in the regression set (4.3), not the dual variables in the first iteration of the dual decomposition-based algorithm.

To perform a quadratic approximation at least

$$n_{\text{reg,min}} \coloneqq (n_\mathbf{b} + 1)(n_\mathbf{b} + 2)/2 \tag{4.11}$$

data points are necessary, i.e., $n_j^{(t)} \geq n_{\text{reg,min}}$, since $\mathbf{p}^{(t)} \in \mathbb{R}^{n_{\text{reg,min}}}$ [CSV09, Wen20]. This shows that the approximations cannot be used in the first iterations of regression-based algorithms. Instead, an initial sampling phase is required, e.g., using the subgradient method, until at least $n_{\text{reg,min}}$ data points have been collected. Furthermore, the choice of the data used in the regression problem, i.e., $\mathcal{J}^{(t)} \subseteq \{1, \ldots, t\}$, plays an important role in the performance of the algorithms and is discussed in the next section.

4.1.2 Regression data selection strategy

The selection of suitable points for the quadratic approximation has been studied extensively in the context of derivative-free optimization [CSV09]. The following criteria are usually considered [GWE16, WRE20]: spread, distance, number of points, and age.

The points should be spread in different directions to provide enough information on the approximated function. For a good local approximation, the majority of the data points should not lie too far away from the current iterate to keep the approximation local. A minimum number of points (4.11) has to be used for the quadratic approximation, but

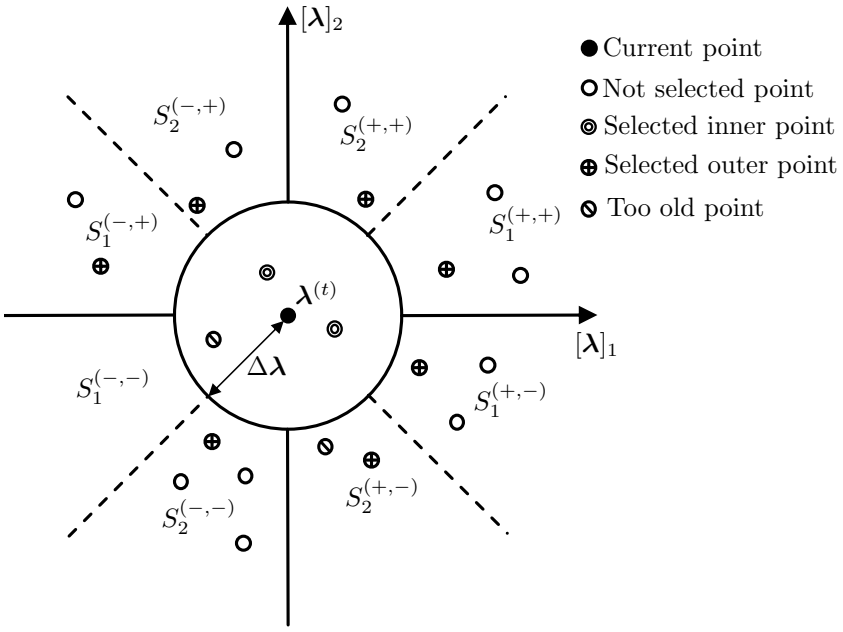


Figure 4.1: Illustration of the regression data selection using the nearest axis point separation algorithm. (from [YWW+23], adapted from [WGE15])

too many points might result in a poor quality of the approximation if the approximated function is not quadratic and the points that are considered are far away from the current iterate, e.g., in the case of a changing set of active individual constraints of the subproblems. Finally, only recent points should be used for the approximation. This is essential if the parameters of the optimization problems change over time, e.g., in the context of modifier adaptation in real-time optimization [GWE16, WGE15].

Different algorithms have been proposed for data selection in the context of quadratic approximation. Two different algorithms from Wenzel et al. [WGE15] and Gao et al. [GWE16] were compared by Wenzel et al. in [WYG17]. Throughout this work, the nearest axis point separation (NAPS) algorithm from Wenzel et al. is used as it yields comparable results to the selection algorithm proposed by Gao et al. at a lower computational cost.

The NAPS algorithm was developed in the context of modifier adaptation for real-time optimization with quadratic approximation, and later also applied to distributed optimization [WE19, Wen20, WRE20]. The algorithm aims at selecting recent points that lie close to the current iterate $\boldsymbol{\lambda}^{(t)}$ as well as evenly spread points lying further away to stabilize the approximation. The algorithm is illustrated in Fig. 4.1 for a two-dimensional example and its steps are summarized in Algorithm 4.

First, all data points that are too old are excluded from the data set, depending on a user-defined age parameter τ. The matrix containing all previously stored points is denoted by $\boldsymbol{\Lambda} \coloneqq [\boldsymbol{\lambda}^{(0)}, \ldots, \boldsymbol{\lambda}^{(t)}]$. These points are divided into inner points $\boldsymbol{\Lambda}_I$ and outer points $\boldsymbol{\Lambda}_O$. A point is classified as an inner point it if lies within a distance $\Delta\boldsymbol{\lambda}$ of the current iterate. All inner points are added to the set of regression points $\boldsymbol{\Lambda}^{(t)}$. The space of dual variables $\mathbb{R}^{n_\mathfrak{b}}$ is then divided into segments according to their sign configuration (in reference to

$\boldsymbol{\lambda}^{(t)}$) and their nearest axis. For instance, in Fig. 4.1 the segment $\mathcal{S}_1^{(+,+)}$ contains all points $\boldsymbol{\lambda}$ where $[\boldsymbol{\lambda}]_1, [\boldsymbol{\lambda}]_2 > 0$ lying closest to the $[\boldsymbol{\lambda}]_1$ axis. The algorithm then cycles through all segments, always selecting the point closest to the current iterate $\boldsymbol{\lambda}^{(t)}$. The cycling process is repeated until at least $n_{\text{reg,min}}$ points have been added to the regression set $\boldsymbol{\Lambda}$. Finally, the values of the approximated function $\hat{\psi}(\boldsymbol{\lambda})$ corresponding to the selected dual variables are selected. The regression data used for the subsequent quadratic approximation is

$$\mathcal{D}^{(t)} = \{(\boldsymbol{\lambda}^{(j)}, \hat{\psi}(\boldsymbol{\lambda}^{(j)}))|\ j \in \mathcal{J}^{(t)}\}, \tag{4.12}$$

where $\mathcal{J}^{(t)}$ contains the indices of the selected data points. The NAPS algorithm is summarized in Algorithm 4.

Algorithm 4 Nearest Axis Point Separation (NAPS, adapted from [WRE20])

Require: $\boldsymbol{\lambda}^{(t)}$, $\boldsymbol{\Lambda}$, $\hat{\boldsymbol{\Psi}}$, τ, $\Delta\boldsymbol{\lambda}$, $n_{\text{reg,min}}$
$\boldsymbol{\Lambda}^{(t)} \leftarrow \emptyset$, $\mathcal{J}^{(t)} \leftarrow \emptyset$
$\boldsymbol{\Lambda} \leftarrow \boldsymbol{\Lambda} \backslash \{\boldsymbol{\lambda}^{(j)}|\ j < t - \tau\}$ ▷ Remove old points.
$\boldsymbol{\Lambda}_I \leftarrow \{\boldsymbol{\lambda}^{(j)}|\ \|\boldsymbol{\lambda}^{(j)} - \boldsymbol{\lambda}^{(t)}\|_2 \leq \Delta\boldsymbol{\lambda}\}$ ▷ Inner points.
$\mathcal{J}^{(t)} \leftarrow \mathcal{J}^{(t)} \cup \{j \in \{1, \dots, t\}|\ \boldsymbol{\lambda}^{(j)} \in \boldsymbol{\Lambda}_I\}$ ▷ Select all indices of inner points.
$\boldsymbol{\Lambda}^{(t)} \leftarrow \boldsymbol{\Lambda}^{(t)} \cup \boldsymbol{\Lambda}_I$ ▷ Select all inner points.
$\boldsymbol{\Lambda}_O \leftarrow \boldsymbol{\Lambda} \backslash \boldsymbol{\Lambda}_I$ ▷ Outer points.
while $|\boldsymbol{\Lambda}^{(t)}| < n_{\text{reg,min}}$ **do**
  for $\mathcal{S}_l^{(\cdot)} \in \{\mathcal{S}_1^{(\cdot)}, \dots, \mathcal{S}_{n_\mathfrak{b}}^{(\cdot)}\}$ **do** ▷ Go through all segments
    $j \leftarrow \arg \min_{j \in \{j|\ \boldsymbol{\lambda}^{(j)} \in \mathcal{S}_l^{(\cdot)} \cap \boldsymbol{\Lambda}_O\}} \|\boldsymbol{\lambda}^{(j)} - \boldsymbol{\lambda}^{(t)}\|_2$
    $\mathcal{J}^{(t)} \leftarrow \mathcal{J}^{(t)} \cup \{j\}$
    $\boldsymbol{\Lambda}^{(t)} \leftarrow \boldsymbol{\Lambda}^{(t)} \cup \{\boldsymbol{\lambda}^{(j)}\}$
    $\mathcal{S}_l^{(\cdot)} \leftarrow \mathcal{S}_l^{(\cdot)} \backslash \{\boldsymbol{\lambda}^{(j)}\}$
    $\boldsymbol{\Lambda}_O \leftarrow \boldsymbol{\Lambda}_O \backslash \{\boldsymbol{\lambda}^{(j)}\}$
  end for
end while
$\hat{\boldsymbol{\psi}}^{(t)} \leftarrow \{\hat{\psi}(\boldsymbol{\lambda}^{(j)}) \in \hat{\boldsymbol{\Psi}}|\ j \in \mathcal{J}^{(t)}\}$ ▷ Match observations to selected points
return $\boldsymbol{\Lambda}^{(t)}$, $\hat{\boldsymbol{\psi}}^{(t)}$, $\mathcal{J}^{(t)}$

4.1.3 Covariance-based step size constraint

Wenzel and Engell [WE19] proposed a covariance-based ellipsoidal step size constraint for the update of the dual variables. The constraint prevents too aggressive steps and leads to updates that are in the region where the local approximation is valid.

First, the covariance matrix of the approximation data is computed,

$$\mathbf{C}^{(t)} = \text{cov}(\boldsymbol{\Lambda}^{(t)}). \tag{4.13}$$

The orientation of the ellipsoid is determined by the eigenvectors of the covariance matrix while the corresponding eigenvalues are related to the lengths of its axes. They proposed to bound the axes of the ellipsoid so that the search space does not become too small,

hindering the progression of the algorithm, or too big, possibly leading to a numerically unbounded problem. This scaling of the axes is performed using a singular value decomposition, which preserves the original orientation. The singular value decomposition is performed for the covariance matrix,

$$\mathbf{C}^{(t)} = \mathbf{U}^{(t)}\mathbf{\Sigma}^{(t)}\mathbf{V}^{(t),T},\ \mathbf{\Sigma}^{(t)} = \operatorname{diag}(\sigma_l^{(t)}), \tag{4.14}$$

where $\sigma_l^{(t)},\ l = 1, \ldots, n_{\mathbf{b}}$ denote the singular values. Subsequently, the singular values are scaled according to

$$\hat{\sigma}_l^{(t)} := \max\{\underline{s}_l, \min\{\sigma_l^{(t)}, \overline{s}_l\}\}, \tag{4.15}$$

where $\underline{s}_l$ and $\overline{s}_l$ are user defined element-wise lower and upper bounds. Note that in this way each axis can be scaled independently, even though using the same lower and upper bounds is usually more convenient in practice. Using the scaled singular values, the scaled covariance matrix can be computed,

$$\hat{\mathbf{C}}^{(t)} = \mathbf{U}^{(t)}\hat{\mathbf{\Sigma}}^{(t)}\mathbf{V}^{(t),T},\ \hat{\mathbf{\Sigma}}^{(t)} = \operatorname{diag}(\hat{\sigma}_l^{(t)}). \tag{4.16}$$

The updated dual variables $\boldsymbol{\lambda}^{(t+1)}$ are then constrained to lie within an ellipsoid which is defined by the scaled covariance matrix,

$$\mathcal{E}(\mathbf{\Lambda}^{(t)}) := \{\boldsymbol{\lambda} \in \mathbb{R}^{n_{\mathbf{b}}} | (\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)})^T \hat{\mathbf{C}}^{(t),-1} (\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)}) \leq (\gamma^{(t)})^2\}. \tag{4.17}$$

Wenzel et al. [WRE20] propose to update $\gamma^{(t)}$ according to

$$\gamma^{(t)} = \max\{\log \|\mathbf{w}_p(\boldsymbol{\lambda}^{(t)})\|_2, \underline{\gamma}\}, \tag{4.18}$$

where $\underline{\gamma}$ is a user-defined lower bound to prevent the ellipsoid from collapsing to a single point. This choice of $\gamma^{(t)}$ allows bigger steps when the current point is far away from the optimum and reduces the step size if the point is near the optimum.

4.1.4 Quadratic approximation coordination

The quadratic approximation coordination (QAC) algorithm was first proposed by Wenzel et al. in [WPK+16]. It was motivated by the distributed optimization of quadratic programs (QPs) without local constraints

$$\min_{\mathbf{x}_1,\ldots,\mathbf{x}_{N_s}} \sum_{i\in\mathcal{I}} \frac{1}{2}\mathbf{x}_i^T\mathbf{H}_i\mathbf{x}_i + \mathbf{c}_i^T\mathbf{x}_i, \tag{4.19a}$$

$$\text{s.t.} \sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i = \mathbf{b}, \tag{4.19b}$$

with symmetric positive definite matrices $\mathbf{H}_i \in S\mathbb{R}^{n_{\mathbf{x}_i}\times n_{\mathbf{x}_i}}$, $\mathbf{c}_i \in \mathbb{R}^{n_{\mathbf{x}_i}}$, $\mathbf{A}_i \in \mathbb{R}^{n_{\mathbf{b}}\times n_{\mathbf{x}_i}}$ and $\mathbf{b} \in \mathbb{R}^{n_{\mathbf{b}}}$. This problem can be summarized as

$$\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{c}^T\mathbf{x}, \tag{4.20a}$$

$$\text{s.t. } \mathbf{A}\mathbf{x} = \mathbf{b}. \tag{4.20b}$$

Algorithm 5 Quadratic Approximation Coordination (QAC)

Require: $\boldsymbol{\lambda}^{(0)}$, $\alpha^{(0)}$, τ, $\Delta\boldsymbol{\lambda}$, $\underline{s}_i$, $\overline{s}_i$, $\underline{\gamma}$, $n_{\text{reg,start}}$, ϵ_p, ϵ_d, $t_{\max}$

$t \leftarrow 0$
repeat
  $t \leftarrow t + 1$
  Send $\boldsymbol{\lambda}^{(t)}$ to all subsystems
  for all $i = 1, \ldots, N_s$ **do**
    $\mathbf{x}_i^{(t+1)} \leftarrow \arg\min_{\mathbf{x}_i \in \mathcal{X}_i} \mathcal{L}_i(\mathbf{x}_i, \boldsymbol{\lambda}^{(t)})$
    Send $\mathbf{A}_i \mathbf{x}_i^{(t+1)}$ to the coordinator
  end for
  $\mathbf{g}(\boldsymbol{\lambda}^{(t)}) \leftarrow \sum_{\forall i \in \mathcal{I}} \mathbf{A}_i \mathbf{x}_i^{(t+1)} - \mathbf{b}$
  if Constraints (2.26b) are inequalities **then**
    for all $l = 1, \ldots, n_{\mathbf{b}}$ **do**
      $[\mathbf{w}_p^{(t)}]_l \leftarrow \max\left\{ \left[\mathbf{g}(\boldsymbol{\lambda}^{(t)})\right]_l, 0 \right\}$
    end for
  else if Constraints (2.26b) are equalities **then**
    $\mathbf{w}_p^{(t)} \leftarrow \mathbf{g}(\boldsymbol{\lambda}^{(t)})$
  end if
  if $j < n_{\text{reg,start}}$ **then** ▷ Perform SG updates until enough points are collected
    $\alpha^{(t)} \leftarrow \text{Update}(\alpha^{(t-1)})$
    if Constraints (2.26b) are inequalities **then**
      $\boldsymbol{\lambda}^{(t+1)} \leftarrow [\boldsymbol{\lambda}^{(t)} + \alpha^{(t)} \mathbf{g}(\boldsymbol{\lambda}^{(t)})]^+$
    else if Constraints (2.26b) are equalities **then**
      $\boldsymbol{\lambda}^{(t+1)} \leftarrow \boldsymbol{\lambda}^{(t)} + \alpha^{(t)} \mathbf{g}(\boldsymbol{\lambda}^{(t)})$
    end if
  else ▷ Perform QAC Updates
    $\mathcal{D}^{(t)} \leftarrow \text{NAPS}(\boldsymbol{\Lambda}, \|\mathbf{w}_p(\boldsymbol{\Lambda})\|_2^2, \tau, \Delta\boldsymbol{\lambda})$
    $\mathcal{E}(\boldsymbol{\Lambda}^{(t)}) \leftarrow \text{ComputeEllipsoid}(\boldsymbol{\Lambda}^{(t)}, \underline{s}_i, \overline{s}_i, \underline{\gamma})$ (4.17)
    $r^{(t)}(\boldsymbol{\lambda}) \leftarrow \text{Regression}(\mathcal{D}^{(t)})$ (4.28)
    if Constraints (2.26b) are inequalities **then**
      $\boldsymbol{\lambda}^{(t+1)} \leftarrow \arg\min_{\boldsymbol{\lambda}} r^{(t)}(\boldsymbol{\lambda}),\ \text{s.t. } \boldsymbol{\lambda} \in \mathcal{E}(\boldsymbol{\Lambda}^{(t)}),\ \boldsymbol{\lambda} \geq \mathbf{0}$
    else if Constraints (2.26b) are equalities **then**
      $\boldsymbol{\lambda}^{(t+1)} \leftarrow \arg\min_{\boldsymbol{\lambda}} r^{(t)}(\boldsymbol{\lambda}),\ \text{s.t. } \boldsymbol{\lambda} \in \mathcal{E}(\boldsymbol{\Lambda}^{(t)})$
    end if
  end if
  $\mathbf{w}_d^{(t)} \leftarrow \boldsymbol{\lambda}^{(t+1)} - \boldsymbol{\lambda}^{(t)}$
until $(\|\mathbf{w}_p^{(t)}\|_2 \leq \epsilon_p \wedge \|\mathbf{w}_d^{(t)}\|_2 \leq \epsilon_d) \vee (t = t_{\max})$
return $\boldsymbol{\lambda}^{(t)}$

where $\mathbf{H} := \text{diag}(\mathbf{H}_1, \ldots, \mathbf{H}_{N_s})$, $\mathbf{c}^T := [\mathbf{c}_1^T, \ldots, \mathbf{c}_{N_s}^T]$, $\mathbf{A} = [\mathbf{A}_1, \ldots, \mathbf{A}_{N_s}]$. It is easy to show that the squared Euclidean norm of the primal residual $\|\mathbf{w}_p\|_2^2 = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$ is a quadratic function of the dual variables. The Lagrange function of problem (4.20) is

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = \frac{1}{2}\mathbf{x}^T \mathbf{H} \mathbf{x} + \mathbf{c}^T \mathbf{x} + \boldsymbol{\lambda}^T (\mathbf{A}\mathbf{x} - \mathbf{b}) . \tag{4.21}$$

Since the matrices $\mathbf{H}_i$ are symmetric and positive definite, i.e., problem (4.20) is convex, the optimal primal solution $\mathbf{x}^*$ can be computed as a function of the dual variables by applying the Karush-Kuhn-Tucker conditions [BV04]:

$$\nabla_x \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) \overset{!}{=} \mathbf{0} \Rightarrow \mathbf{x}^*(\boldsymbol{\lambda}) = -\mathbf{H}^{-1}(\mathbf{c} + \mathbf{A}^T\boldsymbol{\lambda}). \tag{4.22}$$

Thus, the primal residual $\mathbf{w}_p$ can be formulated as a function of the dual variables,

$$\mathbf{w}_p(\boldsymbol{\lambda}) = \mathbf{A}\mathbf{x}^*(\boldsymbol{\lambda}) - \mathbf{b} = -\mathbf{A}\mathbf{H}^{-1}(\mathbf{c} + \mathbf{A}^T\boldsymbol{\lambda}) - \mathbf{b}. \tag{4.23}$$

Computing the squared Euclidean norm of the primal residual leads to

$$\|\mathbf{w}_p(\boldsymbol{\lambda})\|_2^2 = \frac{1}{2}\boldsymbol{\lambda}^T \underbrace{2\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^T\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^T}_{=:\hat{\mathbf{Q}}}\boldsymbol{\lambda} + \underbrace{2(\mathbf{c}^T\mathbf{H}^{-1}\mathbf{A}^T + \mathbf{b}^T)\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^T}_{=:\hat{\mathbf{q}}^T}\boldsymbol{\lambda} + \underbrace{(\mathbf{c}^T\mathbf{H}^{-1}\mathbf{A}^T + \mathbf{b}^T)(\mathbf{A}\mathbf{H}^{-1}\mathbf{c} + \mathbf{b})}_{=:\hat{q}_0}. \tag{4.24}$$

This shows that the squared Euclidean norm of the primal residual is a quadratic function of the dual variables [WRE20],

$$\|\mathbf{w}_p(\boldsymbol{\lambda})\|_2^2 = \frac{1}{2}\boldsymbol{\lambda}^T\hat{\mathbf{Q}}\boldsymbol{\lambda} + \hat{\mathbf{q}}^T\boldsymbol{\lambda} + \hat{q}_0. \tag{4.25}$$

The QAC algorithm is based on local quadratic approximations of the squared Euclidean norm of the primal residual, i.e., of

$$\hat{\psi}(\boldsymbol{\lambda}) = \|\mathbf{w}_p(\boldsymbol{\lambda})\|_2^2 = \left\| \sum_{i \in \mathcal{I}} \mathbf{A}_i \mathbf{x}_i^*(\boldsymbol{\lambda}) - \mathbf{b} \right\|_2^2. \tag{4.26}$$

The surrogate function is quadratic

$$r^{(t)}(\boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{\lambda}^T\mathbf{Q}^{(t)}\boldsymbol{\lambda} + \mathbf{q}^{(t),T}\boldsymbol{\lambda} + q_0^{(t)} \tag{4.27}$$

with the parameters

$$\mathbf{Q}^{(t)}, \mathbf{q}^{(t)}, q_0^{(t)} = \arg\min_{\mathbf{Q},\mathbf{q},q_0} \sum_{j \in \mathcal{J}^{(t)}} \left\| r^{(t)}(\boldsymbol{\lambda}^{(j)}) - \|\mathbf{w}_p(\boldsymbol{\lambda}^{(j)})\|_2^2 \right\|_2^2. \tag{4.28}$$

The regression data

$$\mathcal{D}^{(t)} = \{(\boldsymbol{\lambda}^{(j)}, \|\mathbf{w}_p(\boldsymbol{\lambda}^{(j)})\|_2^2) |\ j \in \mathcal{J}^{(t)}\}, \tag{4.29}$$

is selected using the NAPS algorithm. After obtaining the surrogate function, the dual variables are updated through a minimization problem, subject to the covariance-based step size constraint

$$\boldsymbol{\lambda}^{(t+1)} = \arg\min_{\boldsymbol{\lambda} \in \mathbb{R}^{n_\mathbf{b}}} r^{(t)}(\boldsymbol{\lambda}), \tag{4.30a}$$

$$\text{s.t. } \boldsymbol{\lambda} \in \mathcal{E}(\boldsymbol{\Lambda}^{(t)}), \tag{4.30b}$$

$$\boldsymbol{\lambda} \geq 0. \tag{4.30c}$$

The QAC algorithm is summarized in Algorithm 5. Again, steps 5–8 are performed by the subproblems in parallel, while steps 9–34 are performed by the coordinator.

A key feature of the QAC algorithm is that it only requires the communication of the contribution to the system-wide constraints from the individual subproblems. Therefore, no sensitive information has to be shared, preserving the privacy of the subsystems. This is essential, e.g., in the case of coupled production systems which might share limited resources while belonging to different companies. Through the quadratic approximation, the QAC algorithm can infer second-order information of the dual problem, thereby improving the rate of convergence compared to the subgradient method, which has access to the same information.

Nevertheless, the QAC algorithm also faces some drawbacks. First, the squared Euclidean norm of the primal residual, which will be also referred to as primal residual in the following for the sake of brevity, is only quadratic for the special case of distributed QPs without individual constraints (4.19). Wenzel et al. [WRE20] showed that the primal residual is a piece-wise quadratic function of the dual variables for distributed QPs with individual constraints. The quadratic approximation in this case depends on the set of active individual constraints. If the set of active individual constraints changes, the primal residual is not smooth [WRE20]. It was shown in the example in Sec. 2.2.2 that this is also the case for the dual function $d(\boldsymbol{\lambda})$. However, while the dual function always retains concavity, the same does not hold for the convexity of the primal residual. Thus, in a more general distributed optimization setting the QAC algorithm tries to approximate a nonsmooth and nonconvex function as a smooth quadratic function, which might reduce its efficiency. The issue of the changing sets of active constraints was addressed in [WRE20] by employing a fallback strategy, if an insensitivity of the primal residual to changes of the dual variables was detected. However, the numerical experiments described in Chapter 6 showed that the QAC algorithm performed better without the fallback strategy. Therefore its discussion is omitted at this point.

4.1.5 Quadratically approximated dual ascent

As discussed in the previous section, approximating the primal residual as a quadratic function suffers from some drawbacks, mainly the loss of convexity and nonsmoothness. The problem of nonconvexity of the primal residual can be circumvented by approximating the dual function, i.e.,

$$\hat{\psi}(\boldsymbol{\lambda}) = d(\boldsymbol{\lambda}) = \min_{\mathbf{x}_i \in \mathcal{X}_i,\, \forall i \in \mathcal{I}} \sum_{i \in \mathcal{I}} \mathcal{L}_i(\mathbf{x}_i, \boldsymbol{\lambda}) - \boldsymbol{\lambda}^T \mathbf{b}, \tag{4.31}$$

which is always concave. For the special case of distributed QPs without individual constraints (4.19) it is also easy to show that the dual function is quadratic. By inserting the optimal values of the primal variables $\mathbf{x}^*(\boldsymbol{\lambda})$ (4.22) into the Lagrange function (4.21) the

dual function computes to

$$d(\boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{\lambda}^T \underbrace{\left(-\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^T\right)}_{=:\tilde{\mathbf{Q}}} \boldsymbol{\lambda} + \underbrace{\left(-\mathbf{c}^T\mathbf{H}^{-1}\mathbf{A}^T - \mathbf{b}^T\right)}_{=:\tilde{\mathbf{q}}^T} \boldsymbol{\lambda} + \underbrace{\left(-\frac{1}{2}\mathbf{c}^T\mathbf{H}^{-1}\mathbf{c}\right)}_{\tilde{q}_0}. \tag{4.32}$$

While the primal residual can become nonconvex, even for convex primal problems, the dual function is always concave, regardless of whether or not the primal problem is convex. In the new proposed algorithm, the dual function is approximated by a quadratic function,

$$d_Q^{(t)}(\boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{\lambda}^T\mathbf{Q}^{(t)}\boldsymbol{\lambda} + \mathbf{q}^{(t),T}\boldsymbol{\lambda} + q_0^{(t)} \tag{4.33}$$

with the parameters

$$\mathbf{Q}^{(t)}, \mathbf{q}^{(t)}, q_0^{(t)} = \arg\min_{\mathbf{Q},\mathbf{q},q_0} \sum_{j\in\mathcal{J}^{(t)}} \|d_Q^{(t)}(\boldsymbol{\lambda}^{(j)}) - d(\boldsymbol{\lambda}^{(j)})\|_2^2. \tag{4.34}$$

The regression data

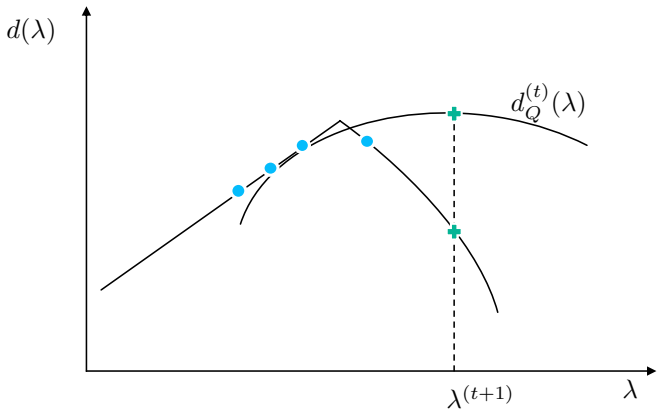
$$\mathcal{D}^{(t)} = \{(\boldsymbol{\lambda}^{(j)}, d(\boldsymbol{\lambda}^{(j)}))|\ j \in \mathcal{J}^{(t)}\}, \tag{4.35}$$

is selected using the NAPS algorithm. Once the quadratic approximation has been performed, the dual variables can be updated by maximizing the approximated dual function. The update step of the dual variables can be interpreted as an ascent step for the dual function using a quadratic approximation. Therefore the algorithm is referred to as Quadratically Approximated Dual Ascent (QADA).

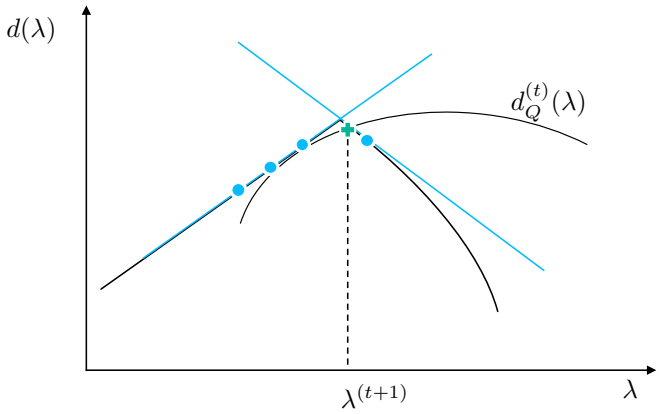
A difference between the QAC and QADA algorithms is the amount of information collected from the subproblems in each iteration. While the QAC algorithm only requires information about the violation of the system-wide constraints, the QADA algorithm additionally requires information about the contributions of the subproblems to the dual function,

$$d_i(\boldsymbol{\lambda}) = \min_{\mathbf{x}_i\in\mathcal{X}_i} \mathcal{L}_i(\mathbf{x}_i, \boldsymbol{\lambda}). \tag{4.36}$$

This essentially means that the QADA algorithm collects the same information as bundle methods (3.9), consisting of the dual variables, the corresponding values of the dual function, and the subgradients. This bundle information can be used to better handle the nonsmoothness of the dual function. Furthermore, the values of the dual function are necessary to compute the duality gap (2.34), i.e., to obtain a lower bound on the objective value of the primal optimization problem. While in the case of convex primal problems optimality can be guaranteed upon convergence of the QAC algorithm, this does not hold for nonconvex problems where the primal residual does not vanish at the dual optimum. As this thesis also deals with nonconvex problems in the form of mixed-integer programs, the value of the dual function is necessary to assess the quality of a found feasible primal solution.



(a) QADA update without bundle cuts. The approximated dual function $d_Q(\lambda)$ deviates from the actual dual function. Updating the dual variables by maximizing the approximated dual function leads to a solution that is further away from the optimum than the previous points.



(b) QADA update with bundle cuts. The constructed cutting planes prevent an update step to a point where the quadratic approximation is not valid.

Figure 4.2: Illustration of the effect of bundle cuts (from [YWW+23]).

Bundle cuts

The core idea of the QADA algorithm is that a quadratic surrogate model of the dual function is computed and optimized to update the dual variables. However, a quadratic function can exhibit a significant approximation error, especially if the optimum is at or near a point of nondifferentiability. This situation is illustrated in Fig. 4.2a. The available points (blue circles) are used to compute the quadratic approximation of the dual function $d_Q(\lambda)$. The maximum of the quadratic approximation (green cross) is far from the actual optimum of the dual function. Updating the dual variables based on this approximation results in a deterioration of the objective value.

To alleviate this issue, the collected subgradients can be used to formulate cutting planes. According to the definition of the subgradient (3.1) the following relation holds between the dual function $d(\boldsymbol{\lambda})$ and a subgradient $\mathbf{g}(\boldsymbol{\lambda}^{(j)})$ at a point $\boldsymbol{\lambda}^{(j)}$:

$$d(\boldsymbol{\lambda}) \leq d(\boldsymbol{\lambda}^{(j)}) + \mathbf{g}^T(\boldsymbol{\lambda}^{(j)})(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(j)}). \tag{4.37}$$

This implies that a quadratic approximation of the dual function is not valid if it does not satisfy condition (4.37). Therefore, the collected subgradients are used to formulate additional constraints on the updated dual variables $\boldsymbol{\lambda}^{(t+1)}$.

Definition 19: Bundle cuts

The constraints on the update of the dual variables,

$$d_A^{(t)}(\boldsymbol{\lambda}^{(t+1)}) \leq d(\boldsymbol{\lambda}^{(j)}) + \mathbf{g}^T(\boldsymbol{\lambda}^{(j)})(\boldsymbol{\lambda}^{(t+1)} - \boldsymbol{\lambda}^{(j)}),\ \forall j \in \{t-\tau+1, \ldots, t\}, \tag{4.38}$$

based on an approximation of the dual function $d_A^{(t)}(\boldsymbol{\lambda})$ and previously collected bundle data $\mathcal{B}^{(t)}$ (3.9) are referred to as bundle cuts. The cuts are summarized as

$$\mathcal{BC}_{d_A}^{(t)} := \{\boldsymbol{\lambda} \in \mathbb{R}^{n_\mathbf{b}} |\ d_A^{(t)}(\boldsymbol{\lambda}) \leq d(\boldsymbol{\lambda}^{(j)}) + \mathbf{g}^T(\boldsymbol{\lambda}^{(j)})(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(j)}),\ \forall j \in \{t-\tau+1, \ldots, t\}\} \tag{4.39}$$

in the following.

The bundle cuts are formulated by using the data points that are not older than the age parameter τ. This usually includes more points than selected by the NAPS algorithm, since the bundle cuts are constructed using subgradient information of the dual function, which is globally valid. In contrast, the approximated dual function based on the regression points selected by the NAPS algorithm is only locally valid.

Fig. 4.2b illustrates the effect of the bundle cuts on the QADA update step. Constraining the quadratic approximation of the dual function to have a value lying below the cutting planes results in an update that is closer to the optimum of the actual dual function. (4.38) constitutes a quadratic inequality constraint on the update of the dual variables, similar to the covariance-based step size constraint. Note that no additional parameters have to be defined by the user for these constraints, except for the age parameter.

QADA update

The dual variables are updated in each iteration of the QADA algorithm by maximizing the approximated dual function, subject to the covariance-based step size constraints and the bundle cuts,

$$\boldsymbol{\lambda}^{(t+1)} = \arg\max_{\boldsymbol{\lambda} \in \mathbb{R}^{n_\mathbf{b}}} d_Q^{(t)}(\boldsymbol{\lambda}), \tag{4.40a}$$

$$\text{s.t. } \boldsymbol{\lambda} \in \mathcal{E}(\boldsymbol{\Lambda}^{(t)}) \cap \mathcal{BC}_{d_Q}^{(t)}, \tag{4.40b}$$

$$\boldsymbol{\lambda} \geq \mathbf{0}. \tag{4.40c}$$

The QADA algorithm is summarized in Algorithm 6. Note that the initial sampling steps in Algorithm 6 are performed using the subgradient method, similar to the QAC algorithm

Algorithm 6 Quadratically Approximated Dual Ascent (QADA)

Require: $\boldsymbol{\lambda}^{(0)}$, $\alpha^{(0)}$, τ, $\Delta\boldsymbol{\lambda}$, $\underline{s}_i$, $\overline{s}_i$, $\underline{\gamma}$, $n_{\text{reg,start}}$, ϵ_p, ϵ_d, $t_{\max}$

1: $t \leftarrow 0$
2: **repeat**
3: $\quad t \leftarrow t+1$
4: $\quad$ Send $\boldsymbol{\lambda}^{(t)}$ to all subproblems
5: $\quad$ **for all** $i = 1, \dots, N_s$ **do**
6: $\quad\quad \mathbf{x}_i^{(t+1)} \leftarrow \arg\min_{\mathbf{x}_i \in \mathcal{X}_i} \mathcal{L}_i(\mathbf{x}_i, \boldsymbol{\lambda}^{(t)})$
7: $\quad\quad$ Send $\mathbf{A}_i \mathbf{x}_i^{(t+1)}$ and $\mathcal{L}_i(\mathbf{x}_i^{(t+1)}, \boldsymbol{\lambda}^{(t)})$ to the coordinator
8: $\quad$ **end for**
9: $\quad \mathbf{g}(\boldsymbol{\lambda}^{(t)}) \leftarrow \sum_{\forall i \in \mathcal{I}} \mathbf{A}_i \mathbf{x}_i^{(t+1)} - \mathbf{b}$
10: $\quad d(\boldsymbol{\lambda}^{(t)}) \leftarrow \sum_{\forall i \in \mathcal{I}} \mathcal{L}_i(\mathbf{x}_i^{(t+1)}, \boldsymbol{\lambda}^{(t)}) - \boldsymbol{\lambda}^{(t),T} \mathbf{b}$
11: $\quad$ **if** Constraints (2.26b) are inequalities **then**
12: $\quad\quad$ **for all** $l = 1, \dots, n_{\mathbf{b}}$ **do**
13: $\quad\quad\quad [\mathbf{w}_p^{(t)}]_l \leftarrow \max\left\{ \left[\mathbf{g}(\boldsymbol{\lambda}^{(t)})\right]_l, 0 \right\}$
14: $\quad\quad$ **end for**
15: $\quad$ **else if** Constraints (2.26b) are equalities **then**
16: $\quad\quad \mathbf{w}_p^{(t)} \leftarrow \mathbf{g}(\boldsymbol{\lambda}^{(t)})$
17: $\quad$ **end if**
18: $\quad$ **if** $j < n_{\text{reg,start}}$ **then** ▷ Perform SG updates until enough points are collected
19: $\quad\quad \alpha^{(t)} \leftarrow \text{Update}(\alpha^{(t-1)})$
20: $\quad\quad$ **if** Constraints (2.26b) are inequalities **then**
21: $\quad\quad\quad \boldsymbol{\lambda}^{(t+1)} \leftarrow [\boldsymbol{\lambda}^{(t)} + \alpha^{(t)} \mathbf{g}(\boldsymbol{\lambda}^{(t)})]^+$
22: $\quad\quad$ **else if** Constraints (2.26b) are equalities **then**
23: $\quad\quad\quad \boldsymbol{\lambda}^{(t+1)} \leftarrow \boldsymbol{\lambda}^{(t)} + \alpha^{(t)} \mathbf{g}(\boldsymbol{\lambda}^{(t)})$
24: $\quad\quad$ **end if**
25: $\quad$ **else** ▷ Perform QADA Updates
26: $\quad\quad \mathcal{D}^{(t)} \leftarrow \text{NAPS}(\boldsymbol{\Lambda}, d(\boldsymbol{\Lambda}), \tau, \Delta\boldsymbol{\lambda})$
27: $\quad\quad \mathcal{E}(\boldsymbol{\Lambda}^{(t)}) \leftarrow \text{ComputeEllipsoid}(\boldsymbol{\Lambda}^{(t)}, \underline{s}_i, \overline{s}_i, \underline{\gamma})$ (4.17)
28: $\quad\quad d_Q^{(t)}(\boldsymbol{\lambda}) \leftarrow \text{Regression}(\mathcal{D}^{(t)})$ (4.34)
29: $\quad\quad$ **if** Constraints (2.26b) are inequalities **then**
30: $\quad\quad\quad \boldsymbol{\lambda}^{(t+1)} \leftarrow \text{argmax}_{\boldsymbol{\lambda}} d_Q^{(t)}(\boldsymbol{\lambda}),\ \text{s.t. } \boldsymbol{\lambda} \in \mathcal{E}(\boldsymbol{\Lambda}^{(t)}) \cap \mathcal{BC}_{d_Q}^{(t)},\ \boldsymbol{\lambda} \geq \mathbf{0}$
31: $\quad\quad$ **else if** Constraints (2.26b) are equalities **then**
32: $\quad\quad\quad \boldsymbol{\lambda}^{(t+1)} \leftarrow \text{argmax}_{\boldsymbol{\lambda}} d_Q^{(t)}(\boldsymbol{\lambda}),\ \text{s.t. } \boldsymbol{\lambda} \in \mathcal{E}(\boldsymbol{\Lambda}^{(t)}) \cap \mathcal{BC}_{d_Q}^{(t)}$
33: $\quad\quad$ **end if**
34: $\quad$ **end if**
35: $\quad \mathbf{w}_d^{(t)} \leftarrow \boldsymbol{\lambda}^{(t+1)} - \boldsymbol{\lambda}^{(t)}$
36: **until** $(\|\mathbf{w}_p^{(t)}\|_2 \leq \epsilon_p \wedge \|\mathbf{w}_d^{(t)}\|_2 \leq \epsilon_d) \vee (t \geq t_{\max})$
37: **return** $\boldsymbol{\lambda}^{(t)}$

(Algorithm 5). However, since the QADA algorithm also uses bundle information, BTM could also be used for the initial sampling. Steps 5–8 are performed in parallel by the subproblems while steps 9–37 are performed by the coordinator.

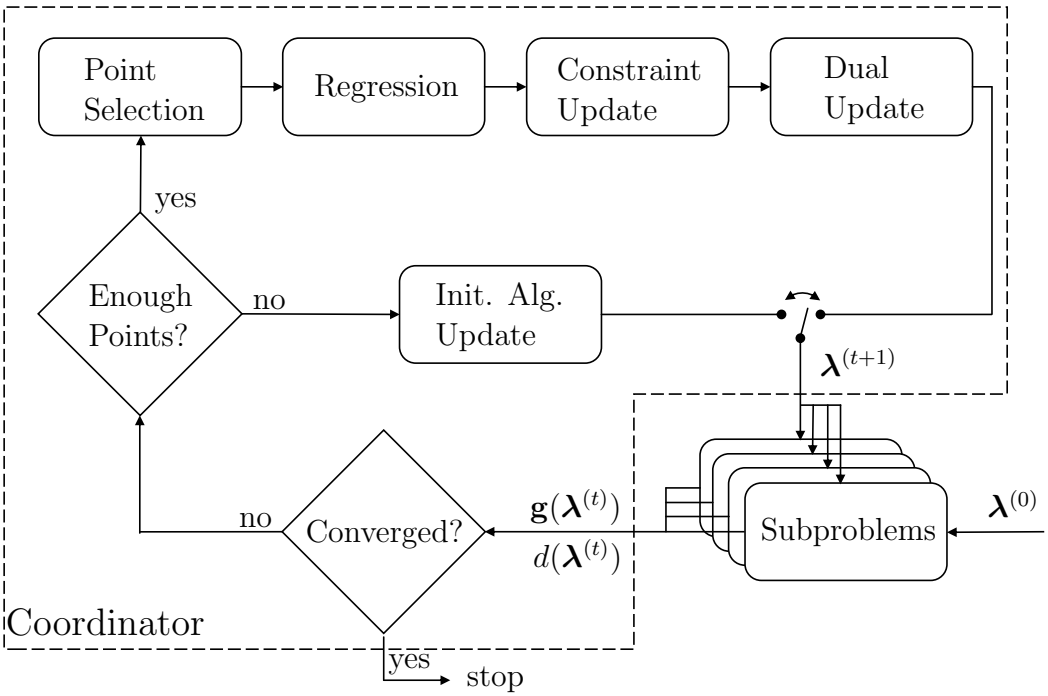


Figure 4.3: Flowchart of the regression-based coordination algorithms (from [YWW+23], adapted from [WRE20]).

4.1.6 Summary of regression-based algorithms

Fig. 4.3 shows a flowchart of the regression-based algorithms (QAC and QADA). The algorithm is initialized with the dual variables $\boldsymbol{\lambda}^{(0)}$. In each iteration, the subproblems are solved for the current values of the dual variables and the subgradient and, in the case of the QADA algorithm, the dual value are communicated to the coordinator. If not enough iterations have been performed, the dual variables are updated using the subgradient method or BTM. Otherwise, the data for the approximation is selected and the regression problem is solved. After updating the covariance-based step size constraint and, in the case of QADA, the bundle cuts, the dual variables are updated by optimizing the surrogate function.

4.2 Quasi-Newton dual ascent

The main drawback of both the QAC and the QADA algorithms is their reliance on the availability of sufficient regression data. The number of required data points for a regression $n_{\text{reg,min}}$ increases quadratically with the number of dual variabls (cf. eq. 4.11). Therefore, an algorithm that quadratically approximates the dual function without solving a regression problem is presented in the following.

Quasi-Newton methods have proven to be very efficient for smooth convex optimization problems. The idea is to approximate the Hessian of the objective function by only using first-order information, i.e., objective values and gradients. In this thesis, the same principle is applied to the dual optimization problem (2.32). If no individual constraints

Algorithm 7 Quasi-Newton Dual Ascent (QNDA)

Require: $\boldsymbol{\lambda}^{(0)}$, $\mathbf{B}^{(0)}$, $\alpha^{(0)}$, τ, ϵ_p, ϵ_d, $t_{\max}$

$t \leftarrow 0$
repeat
  $t \leftarrow t + 1$
  Send $\boldsymbol{\lambda}^{(t)}$ to all subproblems
  for all $i = 1, \ldots, N_s$ **do**
    $\mathbf{x}_i^{(t+1)} \leftarrow \arg\min_{\mathbf{x}_i \in \mathcal{X}_i} \mathcal{L}_i(\mathbf{x}_i, \boldsymbol{\lambda}^{(t)})$
    Send $\mathbf{A}_i \mathbf{x}_i^{(t+1)}$ and $\mathcal{L}_i(\mathbf{x}_i^{(t+1)}, \boldsymbol{\lambda}^{(t)})$ to the coordinator
  end for
  $\mathbf{g}(\boldsymbol{\lambda}^{(t)}) \leftarrow \sum_{\forall i \in \mathcal{I}} \mathbf{A}_i \mathbf{x}_i^{(t+1)} - \mathbf{b}$
  $d(\boldsymbol{\lambda}^{(t)}) \leftarrow \sum_{\forall i \in \mathcal{I}} \mathcal{L}_i(\mathbf{x}_i^{(t+1)}, \boldsymbol{\lambda}^{(t)}) - \boldsymbol{\lambda}^{(t),T} \mathbf{b}$
  if Constraints (2.26b) are inequalities **then**
    for all $l = 1, \ldots, n_{\mathbf{b}}$ **do**
      $[\mathbf{w}_p^{(t)}]_l \leftarrow \max\left\{ \left[\mathbf{g}(\boldsymbol{\lambda}^{(t)})\right]_l, 0 \right\}$
    end for
  else if Constraints (2.26b) are equalities **then**
    $\mathbf{w}_p^{(t)} \leftarrow \mathbf{g}(\boldsymbol{\lambda}^{(t)})$
  end if
  $\alpha^{(t)} \leftarrow \text{Update}(\alpha^{(t-1)})$
  if $j = 1$ **then** ▷ Perform SG update in first iteration.
    if Constraints (2.26b) are inequalities **then**
      $\boldsymbol{\lambda}^{(t+1)} \leftarrow [\boldsymbol{\lambda}^{(t)} + \alpha^{(t)} \mathbf{g}(\boldsymbol{\lambda}^{(t)})]^+$
    else if Constraints (2.26b) are equalities **then**
      $\boldsymbol{\lambda}^{(t+1)} \leftarrow \boldsymbol{\lambda}^{(t)} + \alpha^{(t)} \mathbf{g}(\boldsymbol{\lambda}^{(t)})$
    end if
  else ▷ Perform QNDA Updates
    $\mathbf{y}^{(t)} \leftarrow \mathbf{g}(\boldsymbol{\lambda}^{(t)}) - \mathbf{g}(\boldsymbol{\lambda}^{(t-1)})$
    $\mathbf{B}^{(t)} \leftarrow \text{BFGS}(\mathbf{B}^{(t-1)}, \mathbf{y}^{(t)}, \mathbf{s}^{(t)})$ (4.46)
    if Constraints (2.26b) are inequalities **then**
      $\boldsymbol{\lambda}^{(t+1)} \leftarrow \text{argmax}_{\boldsymbol{\lambda}} d_B^{(t)}(\boldsymbol{\lambda}),\ \text{s.t. } \|\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)}\|_2^2 \leq \alpha^{(t)},\ \boldsymbol{\lambda} \in \mathcal{BC}_{d_B}^{(t)},\ \boldsymbol{\lambda} \geq \mathbf{0}$
    else if Constraints (2.26b) are equalities **then**
      $\boldsymbol{\lambda}^{(t+1)} \leftarrow \text{argmax}_{\boldsymbol{\lambda}} d_B^{(t)}(\boldsymbol{\lambda}),\ \text{s.t. } \|\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)}\|_2^2 \leq \alpha^{(t)},\ \boldsymbol{\lambda} \in \mathcal{BC}_{d_B}^{(t)}$
    end if
  end if
  $\mathbf{s}^{(t+1)} \leftarrow \boldsymbol{\lambda}^{(t+1)} - \boldsymbol{\lambda}^{(t)}$
  $\mathbf{w}_d^{(t)} \leftarrow \mathbf{s}^{(t+1)}$
until $(\|\mathbf{w}_p^{(t)}\|_2 \leq \epsilon_p \wedge \|\mathbf{w}_d^{(t)}\|_2 \leq \epsilon_d) \vee (t = t_{\max})$
return $\boldsymbol{\lambda}^{(t)}$

(2.26c) are considered, the dual function is smooth and the subgradient is equal to the gradient. In the case of individually constrained subproblems, nondifferentiabilities of the dual function occur at the points where the set of active constraints changes. However,

quasi-Newton update steps can still be employed to compute a search direction of the dual function.

A decentralized quasi-Newton algorithm was presented by Eisen et al. in [EMR16] and [EMR17]. There the subproblems use the curvature of their objective function and estimate that of their neighbors. The proposed decentralized Broyded-Fletcher-Goldfarb-Shanno (D-BFGS) method relies on local communication between the subproblems without aggregating information through a central coordinator. However, this network topology requires the exchange of objective gradients, which is not considered in this thesis. Furthermore, no individual constraints were considered, which would result in a nonsmoothness of the dual problem. In contrast, the algorithm presented in the following section estimates the curvature of the dual function, while taking the nonsmoothness into account through the previously discussed bundle cuts.

The idea of Newton methods is to approximate the objective function $d(\boldsymbol{\lambda})$ by a quadratic function around the current iterate $\boldsymbol{\lambda}^{(t)}$ through its Taylor series,

$$d(\boldsymbol{\lambda}) \approx \frac{1}{2}(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)})^T \nabla^2 d(\boldsymbol{\lambda}^{(t)})(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)}) + \nabla^T d(\boldsymbol{\lambda}^{(t)})(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)}) + d(\boldsymbol{\lambda}^{(t)}). \tag{4.41}$$

As the dual function is nonsmooth, i.e., $\nabla d(\boldsymbol{\lambda})$ and $\nabla^2 d(\boldsymbol{\lambda})$ do not exist for every value of $\boldsymbol{\lambda}$, if the set of active constraints changes, the quadratic approximation (4.41) cannot be used in practice. Even for distributed problems without local constraints, i.e., for smooth dual functions, the Hessian $\nabla^2 d(\boldsymbol{\lambda})$ is usually not readily available in a distributed setting. Therefore, instead of using the analytical gradient and Hessian, approximations are used, resulting in the following approximation of the dual function:

$$d_B^{(t)}(\boldsymbol{\lambda}) = \frac{1}{2}(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)})^T \mathbf{B}^{(t)}(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)}) + \mathbf{g}^T(\boldsymbol{\lambda}^{(t)})(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)}) + d(\boldsymbol{\lambda}^{(t)}), \tag{4.42}$$

where the gradient is replaced by a subgradient $\mathbf{g}(\boldsymbol{\lambda}^{(t)})$ and the Hessian is approximated in each iteration by the matrix $\mathbf{B}^{(t)}$, leading to a quasi-Newton method. The update of the approximated Hessian $\mathbf{B}^{(t)}$ can also be interpreted as the solution to an optimization problem based on previous data. Thus, the surrogate function $d_B(\boldsymbol{\lambda})$ is obtained through the solution of an optimization problem (4.1). However, unlike in the case of the QAC or QADA algorithms, no regression is performed. To compute an update of the approximated Hessian, the variations of the dual variables,

$$\mathbf{s}^{(t)} = \boldsymbol{\lambda}^{(t)} - \boldsymbol{\lambda}^{(t-1)} \tag{4.43}$$

and of the subgradients

$$\mathbf{y}^{(t)} = \mathbf{g}(\boldsymbol{\lambda}^{(t)}) - \mathbf{g}(\boldsymbol{\lambda}^{(t-1)}) \tag{4.44}$$

are defined. The approximated Hessian is then updated according to [NW06],

$$\mathbf{B}^{(t)} = \arg \min_{\mathbf{B} \in \mathcal{S}\mathbb{R}^{n_\mathbf{b} \times n_\mathbf{b}}} \|\mathbf{B} - \mathbf{B}^{(t-1)}\|_F \tag{4.45a}$$

$$\text{s.t. } \mathbf{B}^{-1}\mathbf{y}^{(t)} = \mathbf{s}^{(t)}, \tag{4.45b}$$

were $\|\cdot\|_F$ denotes the (weighted) Frobenius norm. The approximated Hessian has to be symmetric since the actual Hessian is also always symmetric. Constraint (4.45b) is called the secant condition and captures the local curvature of the objective function. The solution of (4.45) can be written in a closed form as [NW06]

$$\mathbf{B}^{(t)} = \mathbf{B}^{(t-1)} + \frac{\mathbf{y}^{(t)}\mathbf{y}^{(t),T}}{\mathbf{y}^{(t),T}\mathbf{s}^{(t)}} - \frac{\mathbf{B}^{(t-1)}\mathbf{s}^{(t)}\mathbf{s}^{(t),T}\mathbf{B}^{(t-1),T}}{\mathbf{s}^{(t),T}\mathbf{B}^{(t-1)}\mathbf{s}^{(t)}}. \tag{4.46}$$

Eq. (4.46) is the well-known BFGS-update scheme. The surrogate function $d_B^{(t)}(\boldsymbol{\lambda})$ is a smooth approximation of the dual function. To perform the approximation of the dual function the same amount of information as in the BTM and QADA algorithms is collected. Therefore the bundle cut constraints can be employed to address the nonsmoothness of the actual dual function. However, the approximation is not based on multiple regression points, as in the case of the QADA algorithm. Thus, the covariance-based step size constraint should not be employed, as it is not representative of the range of validity of the approximation. Instead, the same trust region as in BTM can be used. The dual variables are updated in each iteration by solving the optimization problem

$$\boldsymbol{\lambda}^{(t+1)} = \underset{\boldsymbol{\lambda}}{\operatorname{argmax}}\ d_B^{(t)}(\boldsymbol{\lambda}), \tag{4.47a}$$

$$\text{s.t. } \|\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)}\|_2^2 \leq \alpha^{(t)}, \tag{4.47b}$$

$$\boldsymbol{\lambda} \in \mathcal{BC}_{d_B}^{(t)}, \tag{4.47c}$$

$$\boldsymbol{\lambda} \geq \mathbf{0}. \tag{4.47d}$$

The proposed algorithm performs an ascent step of the dual function using a quasi-Newton method. Hence it is referred to as Quasi-Newton Dual Ascent (QNDA). The algorithm is summarized in Algorithm 7. Steps 5–8 are performed in parallel by the subproblems while steps 9–36 are performed by the coordinator.

4.3 Convergence of the QADA and QNDA algorithms

In this section, a preliminary analysis of the convergence properties of the QADA and QNDA algorithms is provided for different cases, distributed quadratic and general convex problems without constraints, distributed quadratic and general convex problems with individual constraints, and distributed mixed-integer quadratic programs. The arguments, as usual, resort to applying sufficiently small step sizes which assures convergence but is not advantageous for the performance of the algorithms, which is demonstrated in the next chapters. In the real implementation and parameterization, the algorithms include heuristic components which can only be validated by tests for well-designed benchmark problems.

First, the case of distributed quadratic programs without individual constraints is considered,

$$\min_{\mathbf{x}_1,\dots,\mathbf{x}_{N_s}} \sum_{i\in\mathcal{I}} \frac{1}{2}\mathbf{x}_i^T\mathbf{H}_i\mathbf{x}_i + \mathbf{c}_i^T\mathbf{x}_i, \tag{4.19a}$$

$$\text{s.t.} \sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i = \mathbf{b}. \tag{4.19b}$$

As was shown in Sec. 4.1.5, the dual function of problem (4.19) is

$$d(\boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{\lambda}^T(-\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^T)\boldsymbol{\lambda} + (-\mathbf{c}^T\mathbf{H}^{-1}\mathbf{A}^T - \mathbf{b}^T)\boldsymbol{\lambda} + \left(-\frac{1}{2}\mathbf{c}^T\mathbf{H}^{-1}\mathbf{c}\right), \tag{4.32}$$

where the matrices and vectors $\mathbf{H}$, $\mathbf{c}$ and $\mathbf{A}$ contain the parameters of the subproblems. In a dual decomposition-based distributed optimization algorithm a subgradient of the dual function in iteration t can be computed as

$$\mathbf{g}(\boldsymbol{\lambda}^{(t)}) = \mathbf{A}\mathbf{x}^{(t+1)} - \mathbf{b}, \tag{4.49}$$

with

$$\mathbf{x}^{(t+1)} = \arg\min_{\mathbf{x}\in\mathbb{R}^{n_\mathbf{x}}} \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{c}^T\mathbf{x} + \boldsymbol{\lambda}^{(t),T}(\mathbf{A}\mathbf{x} - \mathbf{b}) \tag{4.50}$$

$$= -\mathbf{H}^{-1}(\mathbf{c} + \mathbf{A}^T\boldsymbol{\lambda}^{(t)}). \tag{4.22}$$

Inserting (4.22) into (4.49) yiels

$$\mathbf{g}(\boldsymbol{\lambda}^{(t)}) = -\mathbf{A}\mathbf{H}^{-1}(\mathbf{c} + \mathbf{A}^T\boldsymbol{\lambda}) - \mathbf{b} = \nabla d(\boldsymbol{\lambda}^{(t)}), \tag{4.51}$$

which shows that in the case of distributed QPs without individual constraints the subgradient is equal to the gradient of the dual function. This implies that the corresponding dual problem is an unconstrained smooth convex problem.

In this special case, the subgradient method is equivalent to the steepest ascent method with a step size $\alpha^{(t)}$ and a search direction $\mathbf{s}^{(t)} = \nabla d(\boldsymbol{\lambda}^{(t)})$,

$$\boldsymbol{\lambda}^{(t)} = \boldsymbol{\lambda}^{(t)} + \alpha^{(t)}\mathbf{s}^{(t)}. \tag{4.52}$$

Quasi-Newton methods generally provide better search directions, by accounting for the curvature of the objective function. The search direction for the BFGS method is given by

$$\mathbf{s}^{(t)} = \mathbf{B}^{(t),-1}\nabla d(\boldsymbol{\lambda}^{(t)}), \tag{4.53}$$

where $\mathbf{B}^{(t)}$ denotes the approximation of the Hessian computed by (4.46). Using this search direction is equivalent to the QNDA algorithm, if bundle cuts are omitted and a line search is used instead of a trust region. Note that since the dual problem is smooth for this special case, bundle cuts are not required. Convergence in this case can be proven by a suitable selection of the step size, e.g., by requiring the satisfaction of the Wolfe conditions [NW06],

$$d(\boldsymbol{\lambda}^{(t)} + \alpha^{(t)}\mathbf{s}^{(t)}) \geq d(\boldsymbol{\lambda}^{(t)}) + \beta_1 \alpha^{(t)} \nabla^T d(\boldsymbol{\lambda}^{(t)})\mathbf{s}^{(t)}, \tag{4.54a}$$

$$\nabla^T d(\boldsymbol{\lambda}^{(t)} + \alpha^{(t)}\mathbf{s}^{(t)})\mathbf{s}^{(t)} \geq \beta_2 \nabla^T d(\boldsymbol{\lambda}^{(t)})\mathbf{s}^{(t)}, \tag{4.54b}$$

$$\beta_1 \in (0, 1),\ \beta_2 \in (\beta_1, 1). \tag{4.54c}$$

However, the problem with finding a suitable step size through conditions (4.54) is that the closed form of the objective function $d(\boldsymbol{\lambda})$ is not known to the coordinator. Thus, the step size has to be adjusted heuristically. The same issue arises when using a trust region approach, as a certain degree of centralized information is necessary to compute optimal hyperparameters. The same convergence properties can be transferred for distributed general convex problems without local constraints, as the subgradient is equal to the gradient of the dual function.

In the case of the QADA algorithm, a quadratic surrogate function is computed by solving a regression problem. In the special case of distributed QPs without individual constraints (4.19) the dual function is quadratic but unknown to the coordinator. For enough data points, assuming a well-conditioned Vandermonde matrix $\mathbf{M}^{(t)}$ (4.9) and except for numerical errors, the surrogate function will be equal to the actual dual function. This means that the update of the dual variables in the QADA algorithm computes the solution to the dual problem. However, the employed trust region will usually prevent convergence within a single iteration.

The situation is more complicated if individual constraints of the subproblems are considered. In the case of distributed QPs with individual constraints Wenzel et al. [WRE20] showed that the primal residual for QAC is piece-wise quadratic, depending on the set of active constraints. The same holds for the dual function. The arguments made above for the case without individual constraints can then be made locally once a point sufficiently close to the optimum has been reached, employing a suitable (small) step size. As long as the step size is small, the dual function can be approximated locally by a quadratic function. If the set of active constraints changes, the quadratic form of the dual function also changes. Again, assuming small update steps, a good approximation can be obtained after a few steps. Note that since the dual function is (globally) concave, moving towards the optima of the piece-wise quadratic regions of the dual function will eventually guide the search toward the global optimum of the dual function.

Similar arguments can be made for more general problems, e.g., distributed convex problems. Many convex optimization algorithms with provable convergence employ a quadratic approximation of the objective function, e.g., sequential quadratic programming (SQP) methods, Newton methods, or quasi-Newton methods. As the dual problem is a convex optimization problem, the same principles can be applied locally. In a region where the active constraints do not change, the arguments made for distributed problems without individual constraints hold, as the subgradient is equal to the gradient. The main difference to more general convex optimization problems is that the dual function exhibits nons-

moothness when the set of active individual constraints changes. Therefore, the QADA and QNDA algorithms combine smooth convex optimization with bundle methods for nonsmooth optimization. As described above, a cutting plane model is defined as,

$$\hat{d}^{(t)}(\boldsymbol{\lambda}) \coloneqq \min_{j \in \mathcal{J}^{(t)}} \{d(\boldsymbol{\lambda}^{(j)}) + \mathbf{g}^T(\boldsymbol{\lambda}^{(j)})(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(j)})\}. \tag{3.10}$$

The cutting plane model will exactly match the concave nonsmooth dual function if all dual variables in its domain were included [BKM14],

$$d(\boldsymbol{\lambda}) = \min_{\hat{\boldsymbol{\lambda}} \in \operatorname{dom} d} \{d(\hat{\boldsymbol{\lambda}}) + \mathbf{g}^T(\hat{\boldsymbol{\lambda}})(\boldsymbol{\lambda} - \hat{\boldsymbol{\lambda}})\}. \tag{4.55}$$

This is the basis of the proof of the (theoretical) convergence of bundle methods as t goes to ∞. In the QADA and QNDA algorithms, the cutting plane model is used as an upper bound of the new objective value. For instance, the QNDA update (4.47) can be equivalently reformulated as

$$\boldsymbol{\lambda}^{(t+1)} = \arg\max_{\boldsymbol{\lambda} \in \mathbb{R}^{n_\mathbf{b}}} \; d_B^{(t)}(\boldsymbol{\lambda}), \tag{4.56a}$$

$$\text{s.t. } \|\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)}\|_2^2 \leq \alpha^{(t)}, \tag{4.56b}$$

$$d_B^{(t)}(\boldsymbol{\lambda}) \leq \hat{d}^{(t)}(\boldsymbol{\lambda}), \tag{4.56c}$$

$$\boldsymbol{\lambda} \geq \mathbf{0}. \tag{4.56d}$$

As t tends to ∞, the constraints (4.56c) prevent the algorithm from moving in the wrong direction, eventually leading to convergence. The same holds for the QADA algorithm.

From a practical point of view, it is not desirable to store all previously collected information in the bundle, as this would necessitate a possibly infinite storage memory. Therefore only recent data is stored in the bundle, both for BTM and the two proposed algorithms, with the age parameter τ being an important parameter. If sufficient data is kept in the bundle good performance can be observed in practice.

In this work distributed mixed-integer quadratic programs are also considered. In this case, the dual problem essentially does not differ from the case of distributed general convex problems. However, the convergence arguments made above apply only to the dual problem, i.e., to the convergence of the dual variables. For convex problems, the optimal primal solution can be obtained at the dual optimum, since strong duality holds (assuming a constraint qualification condition is satisfied). This is however not the case for integer problems. It can generally not be guaranteed that a feasible primal solution will be obtained, even at the dual optimum. Vujanic et al. [VEG+16] propose tightening the right-hand side of the system-wide constraints and prove that a feasible solution of the original primal problem is obtained at the dual optimum of the modified problem, additionally providing some performance guarantees. The same tightening is applied in this work and discussed in more detail in Sec. 6.2. Feasibility is proven for mixed-integer linear programming problems in [VEG+16]. The transfer of these results to mixed-integer

quadratic programming problems is an open research question.

From a practical point of view, a key issue when considering distributed integer problems (or nonconvex problems in general) in a dual decomposition-based distributed optimization setting is that all subproblems have to be solved to global optimality. Recall that the dual function is defined as the infimum of the Lagrange function for a value of the dual variables (2.28). Global optimality of mixed-integer programming problems with convex relaxations can be assessed through the obtained integrality gap. Prematurely terminating the optimization of the subproblems at a suboptimal solution, or converging to a local minimum of the subproblems for continuous nonconvex problems can lead to the loss of convexity of the sampled response surface of the dual function or the computation of wrong subgradients. Applying the proposed algorithms to nonconvex problems where global optimality of the subproblems cannot be guaranteed is also an open research question, but not one related to the proposed algorithms specifically.

Note that all discussions on convergence consider the case that the dual optimum is found for $t \to \infty$. This however does not guarantee the efficiency of the algorithms. For instance, the general ADMM algorithm for problem (3.18) provably converges to the dual optimum under certain convexity assumptions. The application of ADMM in practice shows that it converges fast to a solution with modest accuracy (in terms of the primal residual) near the optimum, but that finding a high-accuracy solution can be very time-consuming [BPC+11]. The practical efficiency of the newly proposed algorithms in comparison to the benchmark algorithms, both in terms of the number of required iterations and solution accuracy, is demonstrated in the next chapters.

5 Small Fictitious Resource Network

Constraint-coupled optimization problems can be interpreted as a collection of subsystems coupled by constraints on shared limited resources. These resources can be both produced and consumed by the different subsystems. In this context, dual decomposition-based distributed optimization is often referred to as a market-based mechanism, where a coordinator adjusts the prices for the shared resources until the supply matches the demand [Wen20]. After presenting several algorithms in Chapters 3 and 4, Chapter 6 describes extensive numerical benchmarks for these algorithms. To motivate these benchmarks, this chapter deals with the distributed optimization of a small fictitious production network as an illustrative example of the application of dual decomposition for subsystems coupled through limited resources. The study is based on the network considered by Wenzel [Wen20] and is extended to include individual constraints, leading to nonsmoothness of the dual function.

5.1 Model of the network

The considered network is depicted in Fig. 5.1. It consists of three plants that produce and/or consume two shared resources. Additionally, one resource is withdrawn from the network while the other one is procured from an external network. The amount leaving and entering the network is assumed to be constant.

Each plant i optimizes its objective function of the form

$$f_i(\mathbf{u}_i) = \underbrace{[\mathbf{c}_i]_1 \cdot [\mathbf{u}_i]_1 + [\mathbf{c}_i]_2 \cdot [\mathbf{u}_i]_2}_{\text{revenues and raw material costs}} + \underbrace{([\mathbf{u}_i]_1 - [\mathbf{u}_i^{\text{ref}}]_1)^2 + ([\mathbf{u}_i]_2 - [\mathbf{u}_i^{\text{ref}}]_2)^2}_{\text{demand tracking}}. \tag{5.1}$$

Fig. 5.1 also depicts the individual optima of the plants, which are infeasible for the system-wide problem due to the shared limited resources, i.e., they would result in an unbalanced network. The resource utilization $\mathbf{r}_i$ of each subsystem i is modeled as

$$\mathbf{r_i} = \mathbf{A}_i \mathbf{u}_i. \tag{5.2}$$

To keep the network balanced the produced and consumed amount, i.e., the total resource utilization $\mathbf{r}$ has to match the amounts of resources leaving and entering the network. The

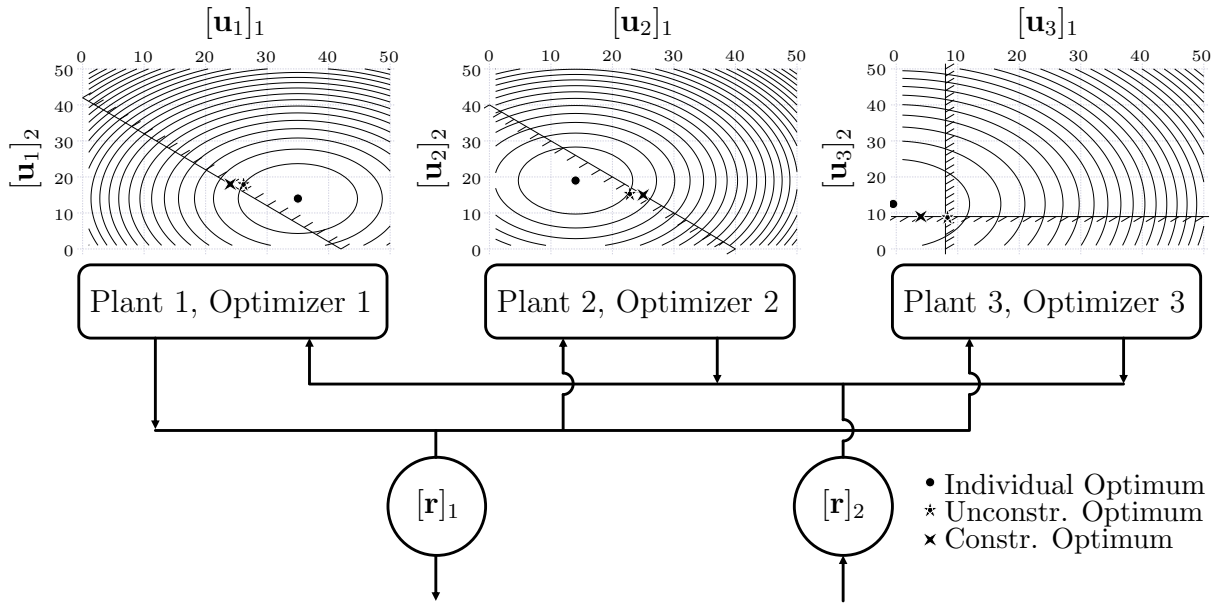


Figure 5.1: Schematic depiction of the fictitious resource network with three plants and two shared resources. The individual optima violate the constraints on the shared limited resources. The optima of the system-wide problem with (Constr.) and without (Uncostr.) individual constraints are also shown (adapted from [Wen20]).

system-wide model for the resource network has the form

$$\min_{\mathbf{u}_1,\mathbf{u}_2,\mathbf{u}_3\in\mathbb{R}^2} \sum_{i=1}^{3} f_i(\mathbf{u}_i), \tag{5.3a}$$

$$\text{s. t.} \sum_{i=1}^{3} \mathbf{A}_i\mathbf{u}_i = \mathbf{r}_e, \tag{5.3b}$$

where $\mathbf{r}_e$ denotes the external resource flow. The parameters of the model are summarized in Tab. 5.1.

The system-wide problem (5.3) can be decomposed by introducing prices $\boldsymbol{\lambda} \in \mathbb{R}^2$ for each shared resource, i.e., dual variables. The Lagrange function for problem (5.3) is

$$\mathcal{L}(\mathbf{u}, \boldsymbol{\lambda}) = \sum_{i=1}^{3} f_i(\mathbf{u}_i) + \boldsymbol{\lambda}^T \left(\sum_{i=1}^{3} \mathbf{A}_i\mathbf{u}_i - \mathbf{r}_e \right). \tag{5.4}$$

Each plant then solves its individual optimization problem

$$\min_{\mathbf{u}_i\in\mathbb{R}^2} f_i(\mathbf{u}_i) + \boldsymbol{\lambda}^T\mathbf{A}_i\mathbf{u}_i. \tag{5.5}$$

5.2 Distributed optimization without individual constraints

The prices for the shared limited resources were adapted by using the subgradient method (SG), the bundle trust method (BTM), the alternating direction method of multipliers (ADMM), the quadratic approximation coordination (QAC) algorithm, the quadratically

Table 5.1: Parameters of the plant models in the fictitious resource network [Wen20]. Negative prices denote sales prices.

	$[\mathbf{c}_i]_1$	$[\mathbf{c}_i]_2$	$[\mathbf{u}_i^{\text{ref}}]_1$	$[\mathbf{u}_i^{\text{ref}}]_2$
Plant 1	-10	8	30	18
Plant 2	12	-14	20	12
Plant 3	11	-13	5	6
Coupling	$\mathbf{A}_1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$,	$\mathbf{A}_2 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$,	$\mathbf{A}_3 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	
External	$\mathbf{r}_e = [5, -6]^T$			

approximated dual ascent (QADA) algorithm and the quasi-Newton dual ascent (QNDA) algorithm. The parameter settings for the algorithms are summarized in Tab. 5.2.

Fig. 5.1 shows the individual optima of the system-wide problem without individual constraints (Unconst. optima). Each plant adjusts its resource utilization according to the prices of the shared resources to satisfy the system-wide constraints. The results of the coordination for the used algorithms are shown in Fig. 5.2. The system-wide problem (5.3) is a quadratic program without individual constraints, thus both the squared Euclidean norm of the primal residual $\|\mathbf{w}_p(\boldsymbol{\lambda})\|_2^2$ and the dual function $d(\boldsymbol{\lambda})$ are quadratic functions. This can be seen in Fig. 5.2a, which depicts the contour plots of the two functions. Fig. 5.2b shows the evolution of both the primal and dual residuals for the considered algorithms. All algorithms manage to converge within the allowed number of iterations. BTM requires the most iterations as it moves slowly toward the optimum and then exhibits oscillations. This issue could probably be alleviated by adaptively decreasing the size of the trust region. The subgradient method, ADMM, and QNDA require a similar number of iterations. Most notably the QAC and QADA algorithms converge very quickly after the initial sampling phase. Finally, Fig. 5.2c depicts the evolution of the dual variables, i.e., the prices. All algorithms converge to the same optimal prices. It can be seen that the systems react sensibly to price changes, as the prices of BTM are almost constant at the optimum for many iterations without finding the optimum.

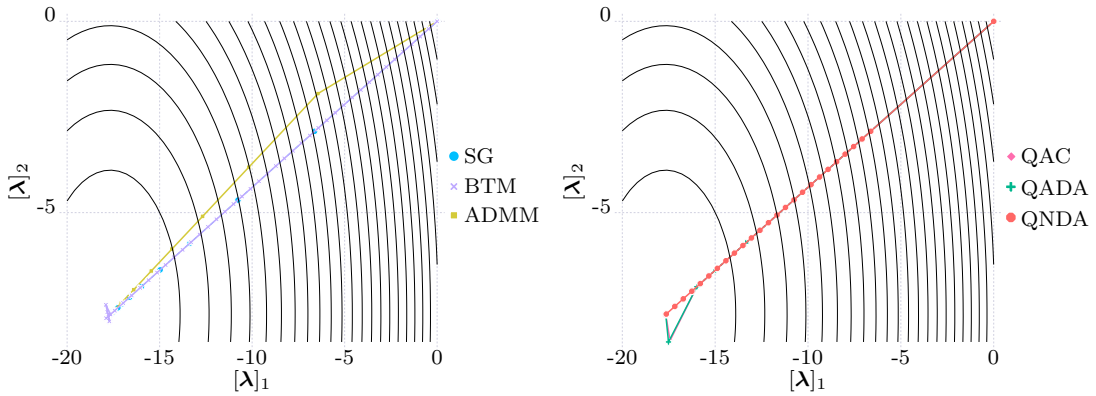
5.3 Distributed optimization with individual constraints

To further assess the performance of the distributed optimization algorithms, individual constraints are added to the optimization problem of each plant:

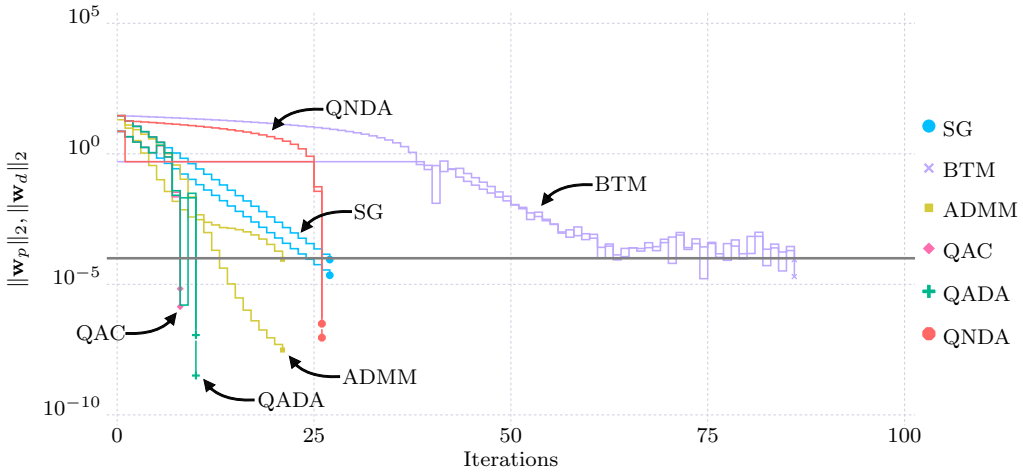
$$\text{Plant 1: } [\mathbf{u}_1]_1 + [\mathbf{u}_1]_2 \leq 42, \tag{5.6a}$$

$$\text{Plant 2: } [\mathbf{u}_2]_1 + [\mathbf{u}_2]_2 \geq 40, \tag{5.6b}$$

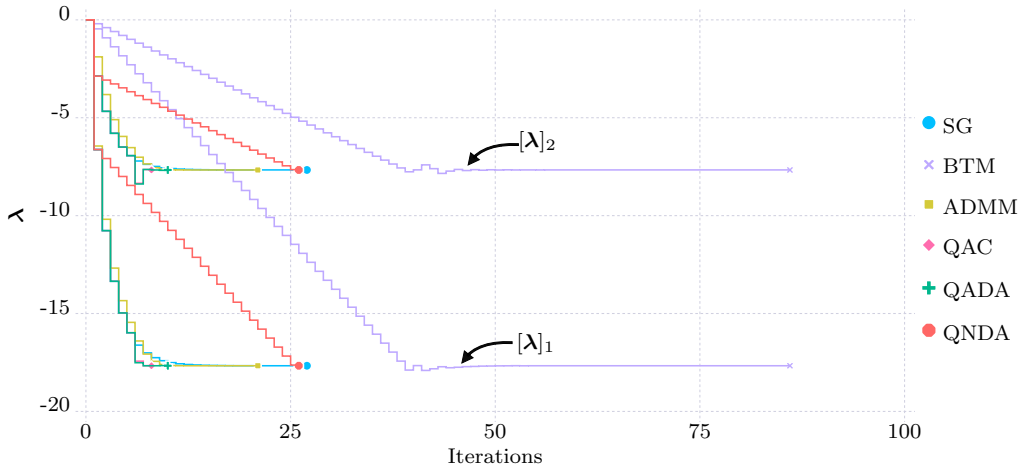
$$\text{Plant 3: } [\mathbf{u}_3]_1 \leq 8,\ [\mathbf{u}_3]_2 \geq 9. \tag{5.6c}$$



(a) Surface plots of the squared primal residual $\|\mathbf{w}_p(\boldsymbol{\lambda})\|_2^2$ (left) and the dual function $d(\boldsymbol{\lambda})$ (right).



(b) Evolution of the primal and dual residuals.



(c) Evolution of the dual variables.

Figure 5.2: Results of the distributed optimization of the fictitious resource network without individual constraints.

The constraints are depicted in Fig. 5.1. The previously computed solutions are now infeasible, thus the plants have to adjust their resource utilization. Fig. 5.3 shows the results for the distributed optimization of the resource network with individual constraints.

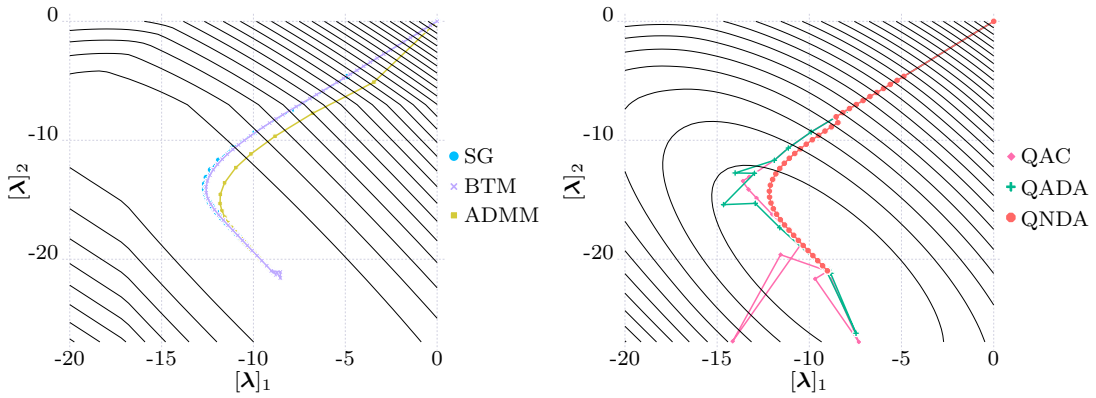
Table 5.2: Parameter settings of the distributed optimization algorithms for the coordination of the fictitious resource network.

	Value	**Description**	**Algorithms**
$\boldsymbol{\lambda}^{(0)}$	$\mathbf{0}$	initial dual variables	All
$\alpha^{(t)}$	0.25	step size/trust region parameter (constant)	SG, BTM, QAC, QADA, QNDA
$t_{\max}$	100	maximum number of iterations	All
ϵ_p,	10^{-4}	primal residual convergence tolerance	All
ϵ_d	10^{-4}	dual residual convergence tolerance	All
$\rho^{(0)}$	1	initial regularization parameter	ADMM
τ_{incr}	1.5	see (3.29)	ADMM
τ_{decr}	1.25	see (3.29)	ADMM
μ	10	see (3.29)	ADMM
$\mathbf{z}^{(0)}$	$\mathbf{0}$	initial auxiliary variables	ADMM
$n_{\text{reg,start}}$	$n_{\text{reg,min}}$	collected points before QAC/QADA are initialized	QAC, QADA
τ	$10 \times n_{\text{reg,min}}$	allowed age of data points	QAC, QADA
$\Delta\boldsymbol{\lambda}$	10^{-3}	radius of inner circle for data selection	QAC, QADA
$\underline{s}_i$	$0.03 \times n_{\mathbf{b}}$	ellipsoid parameter (4.15)	QAC, QADA
$\overline{s}_i$	$4.5 \times n_{\mathbf{b}}$	ellipsoid parameter (4.15)	QAC, QADA
$\gamma^{(t)}$	3.0	ellipsoid parameter (constant) 4.18)	QAC, QADA
$\mathbf{B}^{(0)}$	$-\mathbf{I}$	initial approximated Hessian	QNDA

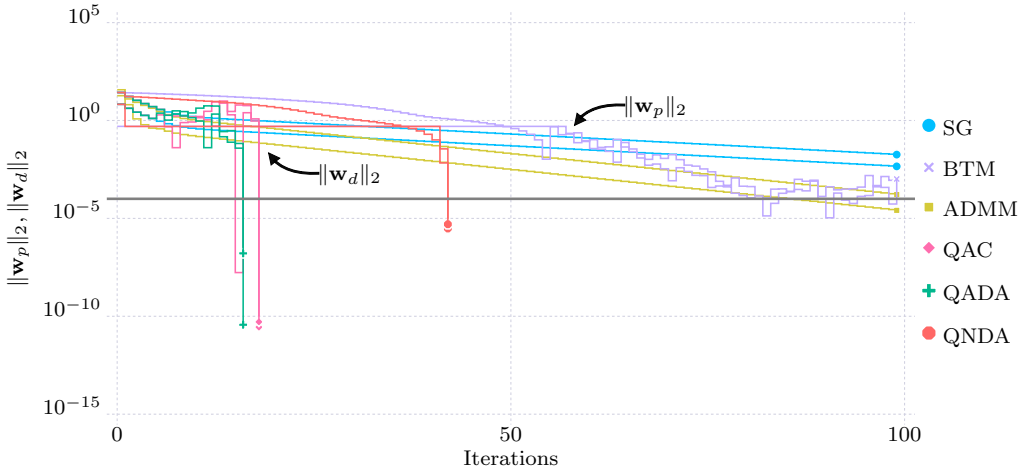
The responses of the plants change due to the presence of the constraints, which can be seen in the different orientations of the contour plots in Fig. 5.3a compared to Fig. 5.2a. Furthermore, the squared Euclidean norm of the primal residual becomes nonconvex. Fig. 5.3b shows that only the algorithms based on smooth approximations manage to converge within the allowed number of iterations. The QAC and QADA algorithms show the best performance out of the examined algorithms. Fig. 5.3c also shows that the algorithms converge to significantly different prices compared to the unconstrained case. Note that the other algorithms eventually also converge to the optimal dual variables, but require more iterations than the maximum number set in this comparison.

5.4 Conclusion

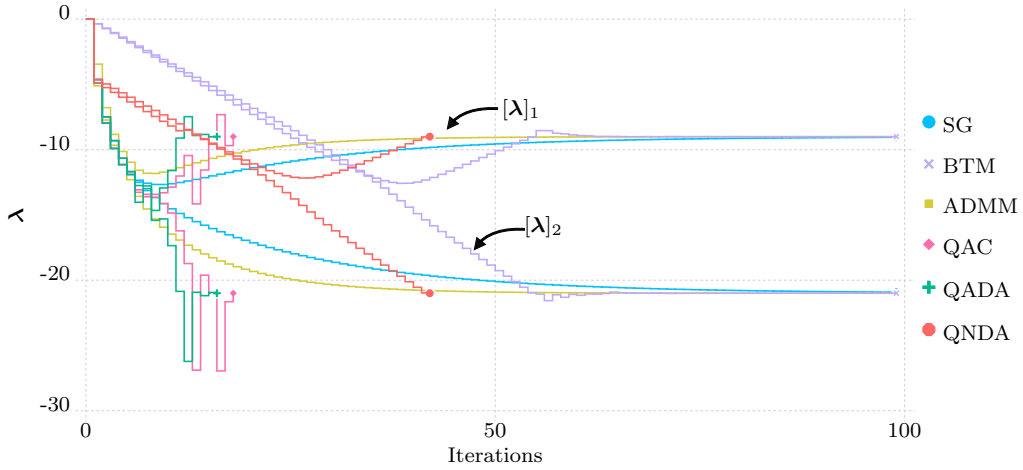
This chapter illustrated the concept of dual decomposition-based distributed optimization on an example of a small fictitious resource network. Each shared limited resource is assigned a price and the prices are communicated to the optimizers of the individual plants. The plants compute their resource utilization and the coordination algorithm



(a) Surface plots of the squared primal residual $\|\mathbf{w}_p(\boldsymbol{\lambda})\|_2^2$ (left) and the dual function $d(\boldsymbol{\lambda})$ (right).



(b) Evolution of the primal and dual residuals.



(c) Evolution of the dual variables.

Figure 5.3: Results of the distributed optimization of the fictitious resource network with individual constraints.

adjusts the prices until the network is balanced. Adding individual constraints to the plants' optimization problems significantly affects their responses to the price signals and in turn, influences the performance of the distributed optimization algorithms. The resulting

response surfaces are nonsmooth and in the case of the primal residual also nonconvex. The algorithms based on smooth approximations exhibited the best convergence behavior in the case of individually constrained subproblems. A large number of similarly structured benchmark problems are examined in the next chapter.

6 Numerical Analysis for General Optimization Problems

In this chapter, the performance of the proposed new QADA and QNDA algorithms is compared to the subgradient method, BTM, ADMM, and the QAC algorithm for different benchmark problems. Three different problem classes are considered, distributed quadratic programs (QP), distributed mixed-integer quadratic programs (MIQP), and distributed convex programs (Conv). The contents of the chapter have been partially published in [YWW+23].

All algorithms and subproblems were implemented in the programming language Julia [BEK+17] using the optimization toolbox JuMP [DHL17]. All affine, QP, and MIQP subproblems were solved using the commercial solver Gurobi [Gur23], while general convex problems were solved using the interior point solver IPOPT [WB06]. The update problem of BTM is guaranteed to be a linear program with affine and convex quadratic constraints, therefore it was solved using Gurobi. The update problems of QAC, QADA, and QNDA were solved using IPOPT. All computations throughout this thesis were performed on a standard Laptop PC (Intel(R) Core(TM) i5-6200U CPU @ 2.30 GHz, 8 GB RAM).

To assess the efficiency of the different algorithms the computation time required for the solution of a distributed optimization problem was computed as [PAL15]

$$T_{\text{comp}} = N_{\text{iter}} \cdot T_{\text{comm}} + \sum_{t=1}^{N_{\text{iter}}} (T_{\text{update}}^{(t)} + \max_{i \in \mathcal{I}} T_{\text{sub},i}^{(t)}), \tag{6.1}$$

where N_{iter} is the number of required iterations, T_{comm} is the required communication time between the coordinator and the subproblems, which is assumed to be constant, $T_{\text{update}}^{(t)}$ is the time required by the coordinator to update the dual variables in iteration t and $T_{\text{sub},i}^{(t)}$ is the solution time of subproblem i in itereation t. In a distributed optimization setting the subproblems can be solved in parallel. Since the coordinator needs to collect the responses of all subproblems the time for updating the primal variables in each iteration is dictated by the slowest subproblem. The communication time is set to $T_{\text{comm}} = 800\ ms$ in the following. All algorithms were terminated if the Euclidean norms of the primal and dual residuals lied below a threshold ϵ_p and ϵ_d respectively, or when the maximum number of iterations $t_{\max}$ was reached.

6.1 Distributed QPs

A large number of distributed QP benchmark problems were defined in [WE19] and [WRE20] with the following structure:

$$\min_{\mathbf{x}_1,\dots,\mathbf{x}_{N_s}} \sum_{i\in\mathcal{I}} \frac{1}{2}\mathbf{x}_i^T\mathbf{H}_i\mathbf{x}_i + \mathbf{c}_i^T\mathbf{x}_i, \tag{6.2a}$$

$$\text{s.t.} \sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i = \mathbf{0}, \tag{6.2b}$$

$$\mathbf{x}_i^{\text{LB}} \leq \mathbf{x}_i \leq \mathbf{x}_i^{\text{UB}},\ \forall i \in \mathcal{I}, \tag{6.2c}$$

with $\mathbf{x}_i \in \mathbb{R}^{n_{\mathbf{x}_i}}$. The system-wide constraints (6.2b) can be interpreted as a resource network balance, where N_s subsystems share resources. The goal is to optimize the overall system in a distributed manner while ensuring that the network is balanced, i.e., that the resource production and consumption match.

The matrices $\mathbf{H}_i$ were generated randomly as symmetric positive definite matrices,

$$\mathbf{H}_i = \mathbf{N}_i^T\mathbf{N}_i,\ \mathbf{N}_i \in \mathbb{R}^{n_{\mathbf{x}_i}\times n_{\mathbf{x}_i}}, \tag{6.3}$$

where the elements of $\mathbf{N}_i$ were drawn from a normal distribution $[\mathbf{N}_i]_{l,j} \in \mathcal{N}(\mu = 0, \sigma = 1)$. The elements of the vectors $\mathbf{c}_i$ were drawn from the same normal distribution. The elements of the matrices of the coupling constraints $\mathbf{A}_i$ were first drawn from a uniform continuous distribution $\mathcal{U}_c(1, 2)$. Afterward, they were altered such that their sign was flipped or they were set to zero through the uniform discrete distribution $\mathcal{U}_d[-1, 0, 1]$,

$$\mathbf{A}_i = \mathbf{B}_i \circ \mathbf{C}_i \in \mathbb{R}^{n_{\mathbf{b}}\times n_{\mathbf{x}_i}},\ [\mathbf{B}_i]_{l,j} \in \mathcal{U}_c(1, 2), [\mathbf{C}_i]_{l,j} \in \mathcal{U}_d[-1, 0, 1], \tag{6.4}$$

where $\circ$ denotes element-wise multiplication. This is done so that not every subsystem contributes to every coupling constraint, similar to the setting of production systems coupled by shared limited resources. In this interpretation, a positive sign of the resource utilization indicates consumption while a positive sign indicates production. If a row in $\mathbf{A}_i$ only contained zeros, a correction step was performed that creates at least one nonzero entry. Box constraints (6.2c) were used as individual constraints for each subproblem. In all cases $[\mathbf{x}_i^{\text{LB}}]_l = -10$ and $[\mathbf{x}_i^{\text{UB}}]_l = 10$ holds.

The number and size of the subproblems were varied as follows:

$$\text{Number of subproblems: } N_s = 2^m, m \in \{2, 3, \dots, 8\},$$
$$\text{Number of variables: } n_{\mathbf{x}_i} \in \{2, 3, \dots, 10\},\ N_s \geq n_{\mathbf{x}_i}.$$

All subproblems contain the same number of primal variables ($n_{\mathbf{x}_i} = n_{\mathbf{x}},\ \forall i \in \mathcal{I}$) and the number of coupling constraints was set equal to the number of variables, i.e., $n_{\mathbf{b}} = n_{\mathbf{x}}$. Fifty problem instances were generated for each pair of number of subproblems and number of coupling constraints/variables $(N_s, n_{\mathbf{b}})$. In the following, the notation $\text{QP}^{(R)}_{(N_s,n_{\mathbf{b}})}$ is used, where R indicates the number of the problem instance. For example, $\text{QP}^{(7)}_{(256,2)}$ refers

to the seventh problem instance containing 256 quadratic programs, each with 2 variables, connected through 2 coupling constraints. In [WE19] and [WRE20] $n_{\mathbf{x}} \in \{2, \ldots, 5\}$ was considered, resulting in a total of 1400 problem instances. By increasing the number of primal variables/ system-wide constraints an additional 1400 problems were generated, resulting in a total of 2800 distributed QPs. It should be noted that all benchmark problems are strongly convex and satisfy Slater's constraint qualification, since $\mathbf{x}_i = \mathbf{0},\ \forall i \in \mathcal{I}$ is a strictly feasible solution. Therefore, strong duality holds, and solving the dual problem is equivalent to solving the primal problem. Furthermore, since the system-wide constraints are equalities, no nonnegativity constraints have to be imposed on the dual variables.

Table 6.1: Detailed parameter settings of the distributed optimization algorithms for the solution of the benchmark problems.

	QP/Conv	**MIQP**	**Description**	**Algorithms**
$\boldsymbol{\lambda}^{(0)}$	$\mathbf{0}$	$\mathbf{0}$	initial dual variables	All
$\alpha^{(0)}$	2×10^{-3}	3×10^{-4}	initial step size/trust region parameter	SG, BTM, QAC, QADA, QNDA
$t_{\max}$	500	500	maximum number of iterations	All
ϵ_p,	10^{-2}	10^{-2}	primal residual convergence tolerance	All
ϵ_d	10^{-2}	–	dual residual convergence tolerance	All
ϵ_b	0.6	0.6	bundle cuts threshold	QADA, QNDA
$\rho^{(0)}$	$1/N_s$	$10^{-3}/N_s$	initial regularization parameter	ADMM
τ_{incr}	1.5	1.5	see (3.29)	ADMM
τ_{decr}	1.25	1.25	see (3.29)	ADMM
μ	10	10	see (3.29)	ADMM
$\mathbf{z}^{(0)}$	$\mathbf{0}$	$\mathbf{0}$	initial auxiliary variables	ADMM
$n_{\text{reg,start}}$	$n_{\text{reg,min}}$	$n_{\text{reg,min}}$	collected points before QAC/QADA are initialized	QAC, QADA
τ	$2 \times n_{\text{reg,min}}$	$1.5 \times n_{\text{reg,min}}$	allowed age of data points	QAC, QADA
$\Delta\boldsymbol{\lambda}$	5×10^{-5}	5×10^{-5}	radius of inner circle for data selection	QAC, QADA
$\underline{s}_i$	$n_{\mathbf{b}} \times 10^{-6}$	$n_{\mathbf{b}} \times 10^{-8}$	ellipsoid parameter (4.15)	QAC, QADA
$\overline{s}_i$	$n_{\mathbf{b}} \times 10^{-3}$	$n_{\mathbf{b}} \times 10^{-4}$	ellipsoid parameter (4.15)	QAC, QADA
$\underline{\gamma}$	1	0.1	ellipsoid parameter (4.18)	QAC, QADA
$\mathbf{B}^{(0)}$	$-\mathbf{I}$	$-\mathbf{I}$	initial approximated Hessian	QNDA

6.1.1 Parameter settings for distributed QPs

For the subgradient method (SG) the initial step size parameter was set to $\alpha^{(0)} = 2 \times 10^{-3}$ and then varied according to

$$\alpha^{(t)} = \frac{\alpha^{(0)}}{\max\{\|\mathbf{w}_p^{(0)}\|_2, \ldots, \|\mathbf{w}_p^{(t)}\|_2\}}, \tag{6.5}$$

as proposed by Wenzel et al. [WRE20]. The same parameter was used for the trust region of BTM. Furthermore, for BTM only recent points were used to construct the cutting plane model, with an age parameter $\tau = 2 \times n_{\text{reg,min}}$. For ADMM the initial regularization parameter was set to $\rho^{(0)} = 1/N_s$ and varied according to (3.29) with $\tau_{\text{decr}} = 1.25$, $\tau_{\text{incr}} = 1.5$ and $\mu = 10$. The parameters for the regression-based algorithms were chosen as in [WRE20]. The age parameter of NAPS was set to $\tau = 2 \times n_{\text{reg,min}}$, similar to the BTM algorithm, while the radius of the inner sphere was set to $\Delta\boldsymbol{\lambda} = 5 \times 10^{-5}$. The lower and upper bounds for the covariance-based step size constraints were set to $\underline{s}_l = n_{\mathbf{b}} \times 10^{-6}$ and $\overline{s}_l = n_{\mathbf{b}} \times 10^{-3}$ respectively. The parameter $\gamma^{(t)}$ was updated according to (4.18), with $\underline{\gamma} = 1$. The regression data was selected using the NAPS algorithm. However, all recent points according to the age parameter τ were used to construct the bundle cuts in the case of the QADA algorithm. The same age parameter was used for the bundle cuts of the QNDA algorithm, while the trust region was defined in the same way as for the BTM algorithm. The approximated Hessian was initialized with the negative identity matrix $\mathbf{B}^{(0)} = -\mathbf{I}$.

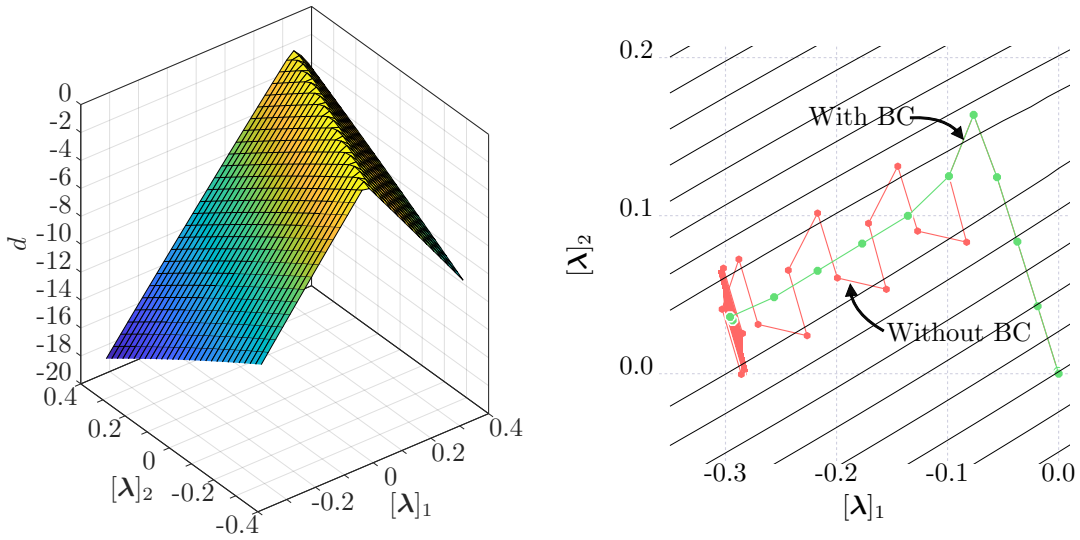
The bundle cuts usually lead to more conservative update steps, especially during the initial iterations. This issue can slow down the convergence of the QADA and QNDA algorithms. Therefore, the bundle cuts were only enforced within a certain distance to the optimum, i.e., when

$$\|\mathbf{w}_p(\boldsymbol{\lambda}^{(t)})\|_2 \leq \epsilon_b \cdot \|\mathbf{w}_p(\boldsymbol{\lambda}^{(0)})\|_2. \tag{6.6}$$

The corresponding parameter was set to $\epsilon_b = 0.6$. The dual variables, and the auxiliary variables in the case of ADMM, were initialized with $\boldsymbol{\lambda}^{(0)} = \mathbf{0}$ and $\mathbf{z}_i^{(0)} = \mathbf{0}$, $\forall i \in \mathcal{I}$. The maximum number of iterations was set to $t_{\max} = 500$ and the convergence tolerances to $\epsilon_p = \epsilon_d = 10^{-2}$. All parameters were set by trial and error, to find the parameters that result in the most converged benchmark problems. All parameters are summarized in Tab. 6.1.

6.1.2 Effect of the bundle cuts

Before the results for the distributed QPs are presented, the effect of the bundle cuts is illustrated on one of the benchmark problems. Fewer subproblems usually lead to a higher degree of nonsmoothness. This is shown in Fig. 6.1 for problem $\text{QP}_{(2,2)}^{(2)}$ using the QNDA algorithm. The optimal dual variables lie at a nondifferentiable point, which is located at the top of a nondifferentiable region. Deactivating the bundle cuts leads to



(a) Surface plot of the dual function $d(\boldsymbol{\lambda})$. (b) Contour plot of the dual function along with the evolution of the dual variables

Figure 6.1: Illustration of the effect of the bundle cuts with the QNDA algorithm for problem $\mathrm{QP}^{(2)}_{(2,2)}$ (adapted from [YR22] © 2022 IEEE).

oscillations as the dual variables jump from one side of the nonsmooth region to the other. In comparison, when bundle cuts are used, cutting planes are generated on both sides of the nonsmooth region, which enforces a trajectory along this region. Note that one jump across the nonsmooth region is performed (in the fifth iteration) which leads to the construction of the corresponding cutting plane. Afterward, no further jumps and oscillations occur, leading to fast convergence to the optimum. The example shows that the bundle cuts play an essential role in handling the nonsmoothness of the dual function in the QADA and QNDA algorithms.

6.1.3 Results for distributed QPs

The 2800 benchmark problems were solved using the subgradient method (SG), BTM, ADMM, QAC, QADA, and QNDA. The QADA algorithm requires an initial sampling phase until enough data points are available for a regression. These initial steps were performed by the SG (QADA-SG), BTM (QADA-BTM), and QNDA (QADA-QNDA) algorithms. In principle, the same algorithms could be used to initialize the QAC algorithm. However, a main feature of the algorithm is that it only requires subgradients from previous iterations. Therefore, the QAC algorithm was only initialized using SG updates.

A summary of the results is given in Tab. 6.2 and Fig. 6.2. A more extensive summary is given in Tab. C.1 in the appendix. The results show that the subgradient method performs poorly for the considered benchmarks. Only a small fraction of the problems are solved within the allowed number of iterations. Additionally, the problems that do converge require a large number of iterations and long computation times. BTM and ADMM are significantly more robust, being able to solve most of the benchmark prob-

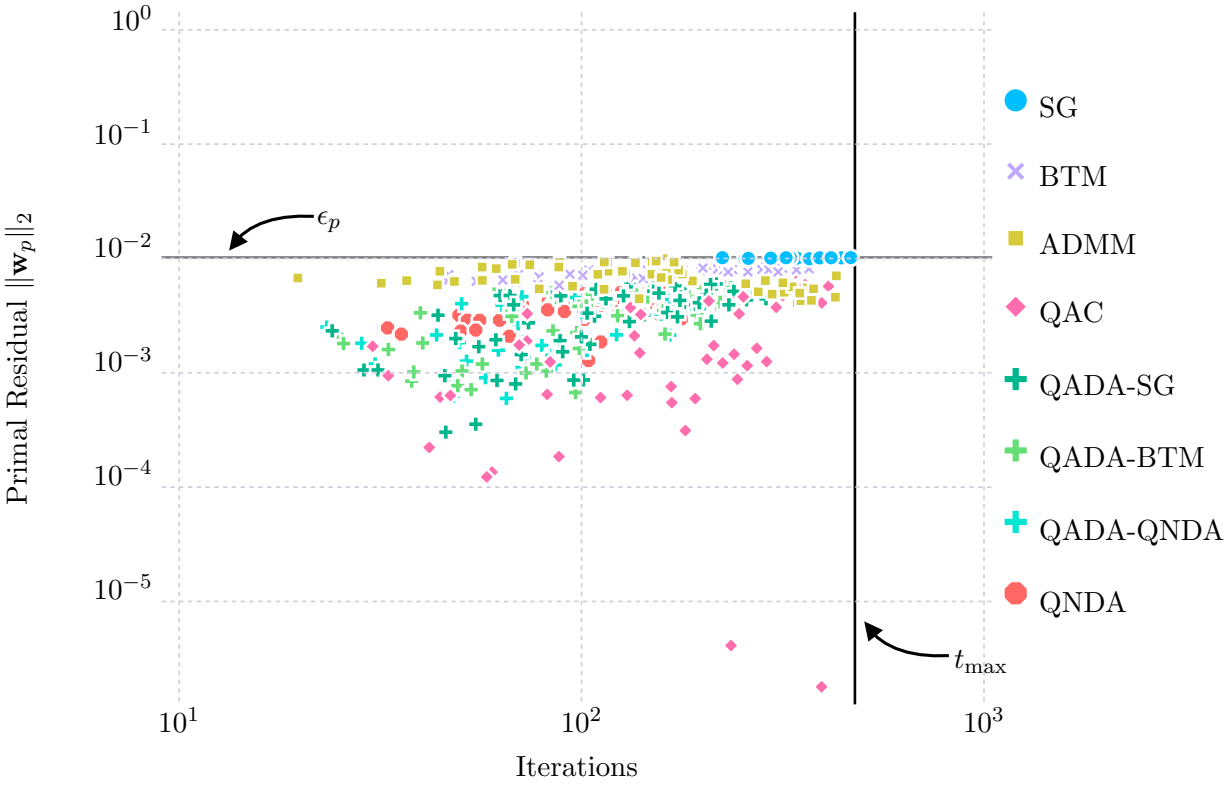
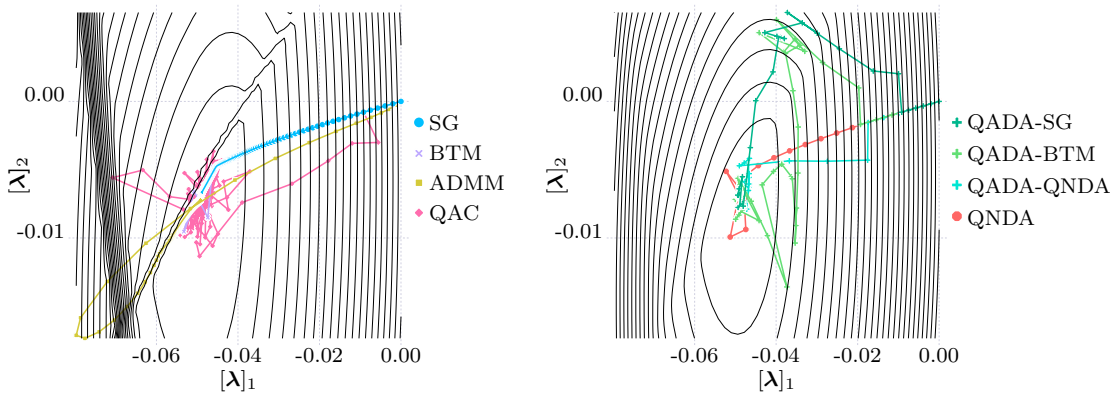


Figure 6.2: Mean values of the primal residuals upon convergence for the distributed quadratic programs. Each data point represents the mean values of the converged problem instances for a pair N_s and $n_{\mathbf{b}}$ (cf. Tab. C.1) (from [YWW+23]).

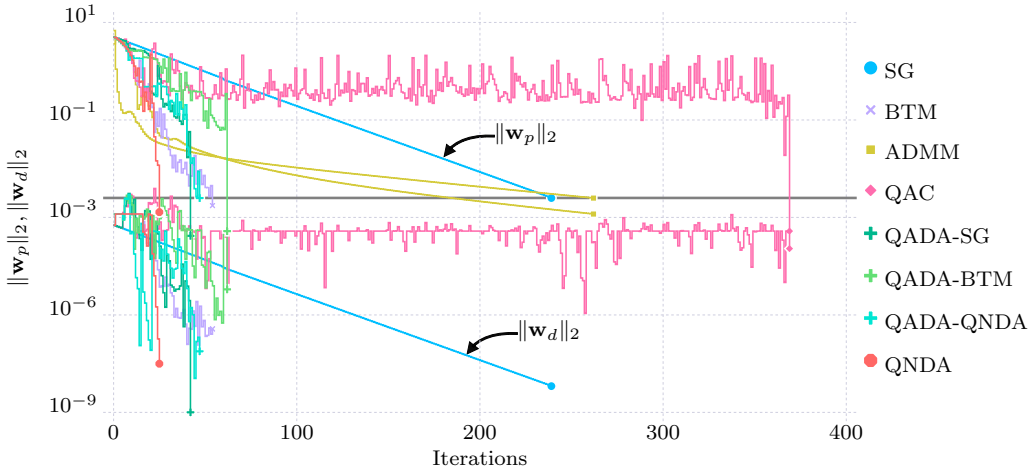
lems. While BTM solves more problems, ADMM requires fewer iterations and exhibits faster computation times for its converged problems. The number of required iterations and computation time for the QAC algorithm is comparable to BTM and ADMM. QAC converges fast near the optimum, yielding the lowest values of the primal residual upon convergence, but it is not very robust, solving only slightly more than half of the bench-

Table 6.2: Summary of the results for the distributed optimization of the QP benchmark problems (mean values of the converged instances only), $\bar{t}$: mean number of iterations until convergence, $\overline{T_{\text{comp}}}$: mean computation time of converged runs (in s), $\overline{\|\mathbf{w}_p\|_2}$: mean primal residual of converged runs ($\times 10^{-3}$), $\%_c$: percentage of converged runs within $t_{\max}$ iterations.

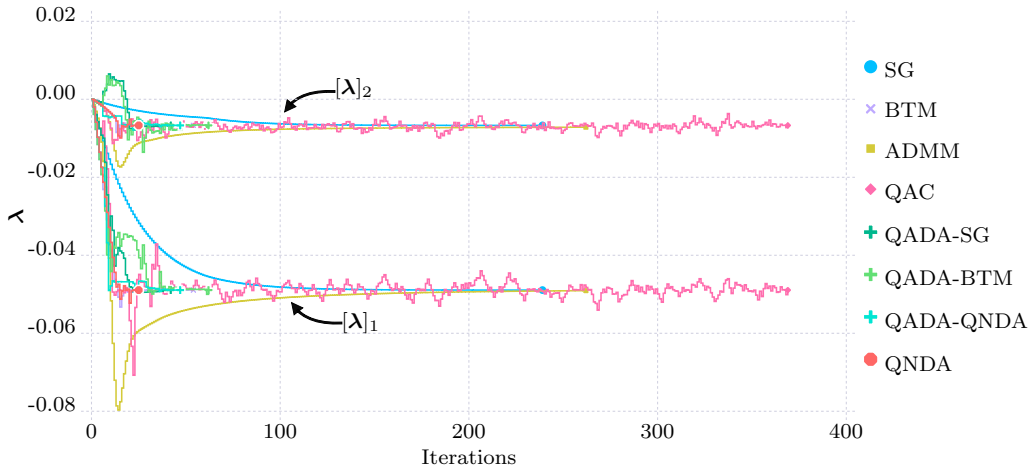
Algorithm	$\bar{t}$	$\overline{T_{\text{comp}}}$	$\overline{\|\mathbf{w}_p\|_2}$	$\%_c$
SG	384.74	308.36	9.88	16.83
BTM	196.95	160.07	7.47	95.11
ADMM	179.55	145.54	6.81	82.3
QAC	199.24	167.73	**2.15**	57.32
QADA-SG	133.49	139.5	3.31	96.46
QADA-BTM	128.10	135.22	3.26	96.96
QADA-QNDA	**127.23**	135.47	3.29	96.57
QNDA	134.31	**126.4**	3.94	**98.50**



(a) Contour plot of the squared primal residual $\|\mathbf{w}_p(\boldsymbol{\lambda})\|_2^2$ (left) and the dual function $d(\boldsymbol{\lambda})$ (right).



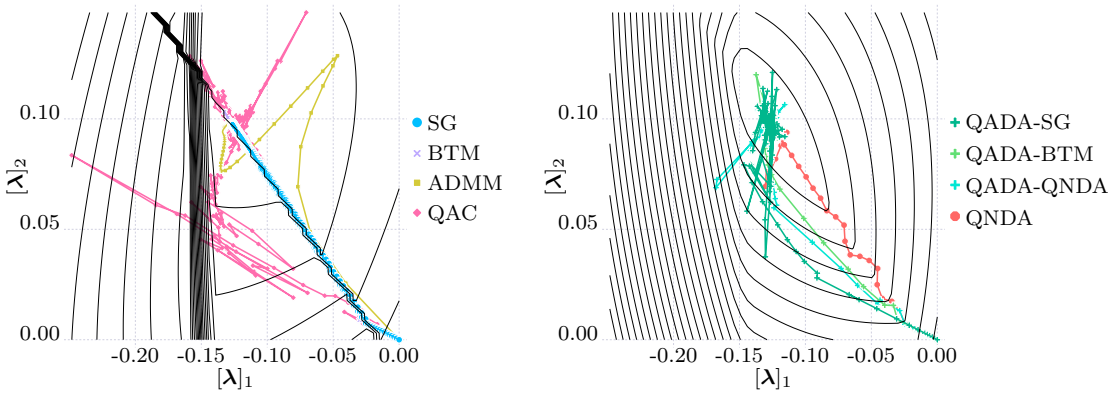
(b) Evolution of the primal and dual residuals.



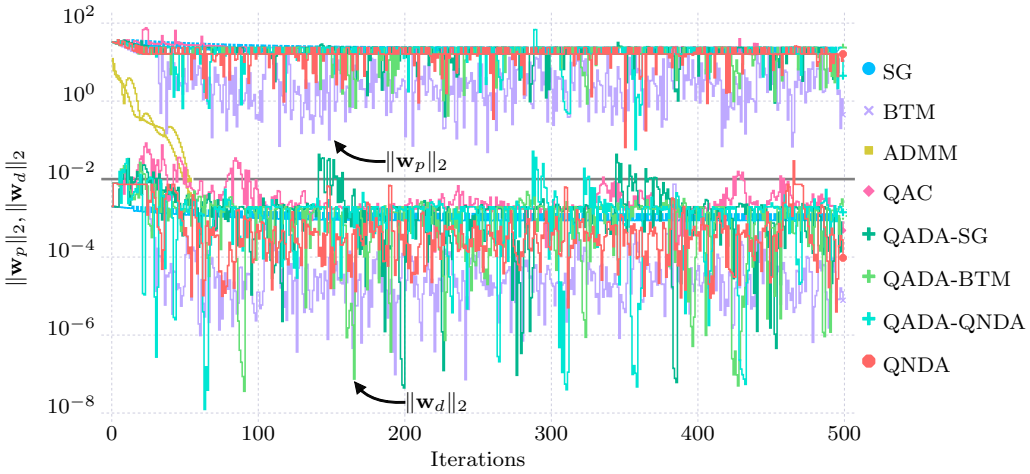
(c) Evolution of the dual varialbes.

Figure 6.3: Results of the distributed optimization of problem $\mathrm{QP}^{(7)}_{(256,2)}$ (from [YWW+23]).

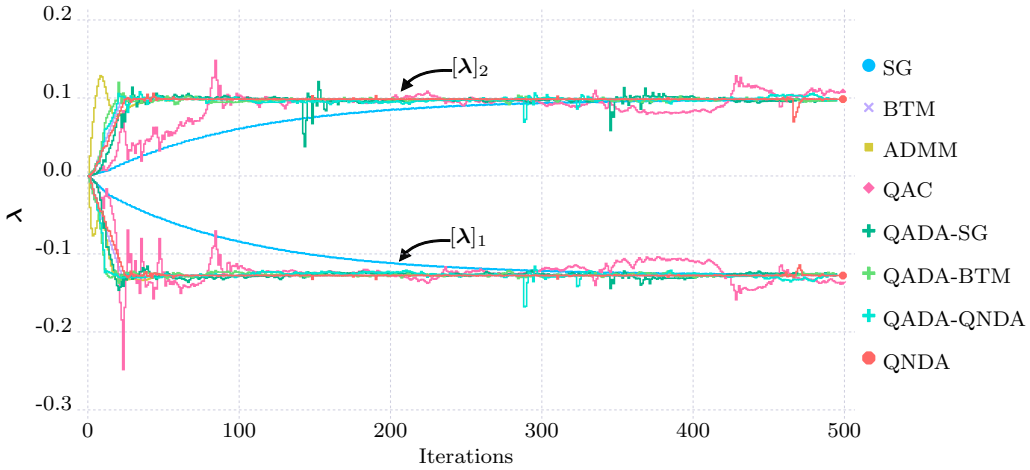
mark problems. Note however, that all other algorithms except the subgradient method use more information on the subproblems and that ADMM enforces that the subproblems are modified which may have practical disadvantages in a fully distributed setting. The QADA and QNDA algorithms show the best performance, both being able to solve



(a) Contour plot of the squared primal residual $\|\mathbf{w}_p(\boldsymbol{\lambda})\|_2^2$ (left) and the dual function $d(\boldsymbol{\lambda})$ (right).



(b) Evolution of the primal and dual residuals.



(c) Evolution of the dual varialbes.

Figure 6.4: Results of the distributed optimization of problem $\mathrm{QP}^{(17)}_{(16,2)}$ (from [YWW+23]).

almost all benchmark problems. Interestingly, while QADA requires fewer iterations to converge, QNDA requires less computation time. This is because the update steps of the dual variables are less expensive in the case of QNDA, as no regression and no singular value decomposition are required for the updates of the approximated dual and the step

size constraints respectively. However, if the required communication time is larger than assumed in this study or if the solution of individual subproblems is a bigger bottleneck, QADA might perform better. In terms of scalability, both ADMM and BTM perform well for cases with relatively few subproblems where they tend to perform better than the approximation-based algorithms. Their performance deteriorates as the problem size increases. In contrast, the approximation-based algorithms scale well with the problem size. The main influencing parameter for the performance of these algorithms is the number of dual variables/ system-wide constraints. The proposed algorithms are especially well suited for distributed optimization problems that consist of many subproblems which are coupled by relatively few constraints.

Fig. 6.3 shows the results for the distributed optimization of benchmark problem $\mathrm{QP}^{(7)}_{(256,2)}$. The contour plots in Fig. 6.3a demonstrate the advantage of the QADA and QNDA algorithms compared to the QAC algorithm. The squared primal residual $\|\mathbf{w}_p(\boldsymbol{\lambda})\|_2^2$ (left) is nonconvex and nonsmooth. In the shown problem instance, the optimum lies near a point of nondifferentiability, making it difficult for the QAC algorithm to find a suitable quadratic approximation. In contrast, the dual function $d(\boldsymbol{\lambda})$ (shown on the right) is concave. Additionally, the effect of the changing set of active constraints, which causes the nonsmoothness, is less profound in the dual function. These effects lead to faster convergence of the QADA and QNDA algorithms, even though QADA initially takes some steps away from the optimum. Among the examined algorithms the ones approximating the dual function (BTM, QADA, QNDA) exhibit the best performance. The subgradient method and ADMM also converge but require more iterations.

While the bundle cuts can handle the nonsmoothness of the dual function in most cases, this does not apply to all benchmark problems. Fig. 6.4 shows the results for benchmark problem $\mathrm{QP}^{(17)}_{(16,2)}$, where the optimum lies at a point of nondifferentiability. No algorithm manages to converge, except for ADMM, which smoothens the dual function via the regularization term in the augmented Lagrange function. All other algorithms terminate close to the optimum but are not able to reach it within the allowed number of iterations. The QADA and QNDA algorithms manage to converge for most benchmark problems, even for the ones where the optimum lies at a nondifferentiable point. From the tests, ADMM is only able to reach an optimum at a nondifferentiable point if only a few subproblems are involved.

6.2 Distributed MIQPs

In [WRE20] only convex QPs were considered. In this thesis, the computational results are extended by also considering distributed MIQPs. The benchmark problems have the following structure:

$$\min_{\mathbf{x}_1,\dots,\mathbf{x}_{N_s}} \sum_{i\in\mathcal{I}} \frac{1}{2}\mathbf{x}_i^T\mathbf{H}_i\mathbf{x}_i + \mathbf{c}_i^T\mathbf{x}_i, \tag{6.7a}$$

$$\text{s.t. } \sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i \leq \mathbf{b}, \tag{6.7b}$$

$$\mathbf{D}_i\mathbf{x}_i \leq \mathbf{d}_i,\ \forall i \in \mathcal{I}, \tag{6.7c}$$

$$\mathbf{x}_i^{\mathrm{LB}} \leq \mathbf{x}_i \leq \mathbf{x}_i^{\mathrm{UB}},\ \forall i \in \mathcal{I}, \tag{6.7d}$$

$$\mathbf{x}_i \in \mathbb{R}^{n^c_{\mathbf{x}_i}} \times \mathbb{Z}^{n^d_{\mathbf{x}_i}}, \forall i \in \mathcal{I}, \tag{6.7e}$$

with $n^c_{\mathbf{x}_i} = \lceil n_{\mathbf{x}_i}/2 \rceil$ and $n^d_{\mathbf{x}_i} = \lfloor n_{\mathbf{x}_i}/2 \rfloor$. The matrices and vectors $\mathbf{H}_i$, $\mathbf{c}_i$, $\mathbf{A}_i$, $\mathbf{x}_i^{\mathrm{LB}}$ and $\mathbf{x}_i^{\mathrm{UB}}$ were generated in the same way as for the distributed QPs. The elements for the individual constraints (6.7c) were drawn from continuous uniform distributions $[\mathbf{D}_i]_{l,j} \in \mathcal{U}_c(-5, 5)$ and $[\mathbf{d}_i]_l \in \mathcal{U}_c(-1, 1)$. The elements of the right-hand side of the system-wide constraints (6.7b) $\mathbf{b}$ were drawn from the same distribution as the elements of the matrices $\mathbf{A}_i$.

Since the system-wide constraints are inequalities a situation might occur where the solutions of the subproblems are completely decoupled, i.e., where $\boldsymbol{\lambda} = \mathbf{0}$ results in a feasible solution. This trivial solution is avoided by tightening the system-wide constraints. Once all subproblems were generated, the decoupled subproblems

$$\min_{\mathbf{x}_i} \frac{1}{2}\mathbf{x}_i^T\mathbf{H}_i\mathbf{x}_i + \mathbf{c}_i^T\mathbf{x}_i, \tag{6.8a}$$

$$\text{s.t. } \mathbf{D}_i\mathbf{x}_i \leq \mathbf{d}_i,\ \forall i \in \mathcal{I}, \tag{6.8b}$$

$$\mathbf{x}_i^{\mathrm{LB}} \leq \mathbf{x}_i \leq \mathbf{x}_i^{\mathrm{UB}},\ \forall i \in \mathcal{I}, \tag{6.8c}$$

$$\mathbf{x}_i \in \mathbb{R}^{\hat{n}_{\mathbf{x}_i}} \times \mathbb{Z}^{\tilde{n}_{\mathbf{x}_i}}, \forall i \in \mathcal{I}, \tag{6.8d}$$

were solved, obtaining the decoupled optimal primal variables $\tilde{\mathbf{x}}_i^*$. The elements of $\mathbf{b}$ were then tightened according to

$$[\mathbf{b}]_l = [\mathbf{b}]_l - (1 + [\boldsymbol{\beta}]_l) \left\| \sum_{i\in\mathcal{I}} \mathbf{A}_i\tilde{\mathbf{x}}_i^* \right\|_2, \tag{6.9}$$

with $[\boldsymbol{\beta}]_l \in \mathcal{U}_c(0.1, 0.3)$. Finally, after generating all subproblem parameters the feasibility of the central problem was evaluated. If the problem was infeasible, the benchmark problem was discarded and a new one was generated.

For the MIQPs large-scale problems with $N_s \gg n_{\mathbf{b}}$ were considered [VEG+16, CNN21]. The number and size of the subproblems were varied as follows:

$$\text{Number of subproblems: } N_s \in \{100, 200, 300, 400, 500\},$$
$$\text{Number of variables: } n_{\mathbf{x}_i} \in \{2, 3, 4, 5\}.$$

All subproblems contain the same number of variables ($n_{\mathbf{x}_i} = n_{\mathbf{x}},\ \forall i \in \mathcal{I}$) and the number of system-wide constraints is equal to the number of variables of each subproblem, i.e., $n_{\mathbf{b}} = n_{\mathbf{x}}$. Ten benchmark problems were generated for each combination $(N_s, n_{\mathbf{b}})$, resulting in a total of 200 MIQP benchmark problems.

6.2.1 Recovery of primal feasibility for distributed MIQPs

Due to the integrality constraints (6.7e) MIQPs are always nonconvex, i.e., strong duality does not hold. This means that the primal problem might not be feasible at the optimal dual solution. Vujanic et al. [VEG+16] proved that a feasible primal solution can be obtained for large-scale mixed-integer linear programs (MILP) if the system-wide constraints are tightened via a contraction. The right-hand side of the system-wide constraints is contracted as follows,

$$\sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i \leq \overline{\mathbf{b}}, \tag{6.10a}$$

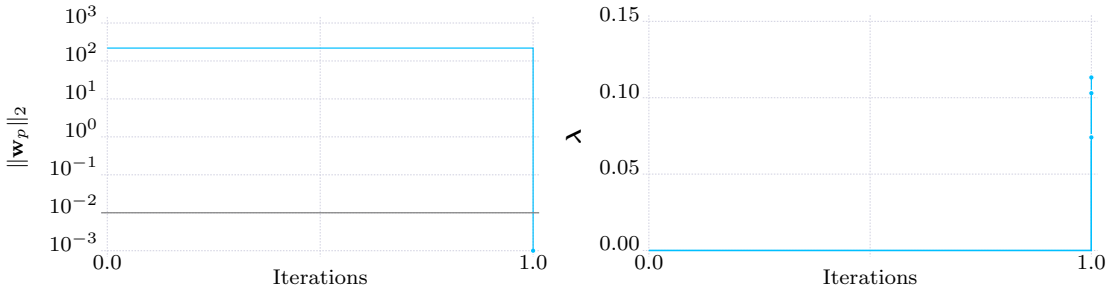
$$\overline{\mathbf{b}} = \mathbf{b} - \boldsymbol{\zeta}, \tag{6.10b}$$

$$[\boldsymbol{\zeta}]_l = n_{\mathbf{b}} \cdot \max_{i\in\mathcal{I}} \left\{ \max_{\mathbf{x}_i\in\mathcal{X}_i} [\mathbf{A}_i]_{l,:}\mathbf{x}_i - \min_{\mathbf{x}_i\in\mathcal{X}_i} [\mathbf{A}_i]_{l,:}\mathbf{x}_i \right\}, \tag{6.10c}$$

where $[\mathbf{A}_i]_{l,:}$ denotes the lth row of the matrix $\mathbf{A}_i$. The same contraction was used in this thesis. Therefore, in the first step, all subproblems have to solve the two inner optimization problems in (6.10c) and communicate the results to the coordinator. The coordinator then collects all responses and tightens the coupling constraints. It is important to note, that the contracted right-hand side $\overline{\mathbf{b}}$ is used to compute the subgradient and dual value within the distributed optimization algorithm. However, the original right-hand side $\mathbf{b}$ is used to compute the values of the primal residual for the termination criterion, as one is interested in the feasibility of the original problem. As the coupling constraints are inequalities, the primal residual is computed using eq. (3.6), and nonnegativity constraints are imposed on the dual variables.

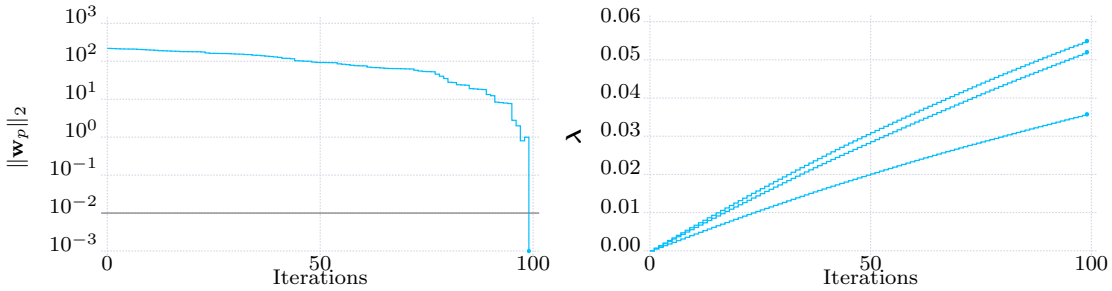
6.2.2 Parameter settings for distributed MIQPs

As noted earlier, the coupling constraints (6.7b) are inequalities for the MIQP benchmark problems. In general, it is easier to find a feasible solution for inequality-constrained problems using dual decomposition-based distributed optimization than for equality-constrained ones. By selecting larger values for the dual variables, the corresponding primal solution is "pushed" towards primal feasibility. This can be achieved by using an aggressive parametrization of the distributed optimization algorithms. However, even though an obtained primal solution might be feasible, it tends to be further away from the optimum, compared to a more conservative parametrization. This is illustrated in Fig. 6.5 for problem $\text{MIQP}^{(1)}_{(300,3)}$ using the subgradient method. By setting the initial step size parameter $\alpha^{(0)}$ to 5×10^{-2} convergence is achieved within a single iteration. In comparison, setting the parameter to 3×10^{-4} leads to convergence after 100 iterations where the step size is updated according to (6.5). However, the more aggressive parameter leads to a primal solution with a relative duality gap of 5.74 % while the conservative choice leads to a gap of 0.48 % (cf. Sec 6.2.3, eq. (6.11)). This is due to the obtained values of



(a) Evolution of the primal and dual residuals for $\alpha^{(0)} = 5 \times 10^{-2}$.

(b) Evolution of the dual varialbes for $\alpha^{(0)} = 5 \times 10^{-2}$.



(c) Evolution of the primal and dual residuals for $\alpha^{(0)} = 3 \times 10^{-4}$.

(d) Evolution of the dual varialbes for $\alpha^{(0)} = 3 \times 10^{-4}$.

Figure 6.5: Results of the distributed optimization of problem $\text{MIQP}^{(1)}_{(300,2)}$ using the subgradient method with different initial step size parameters $\alpha^{(0)}$.

the dual variables. As seen in Fig. 6.5b and 6.5d the more conservative choice converges with smaller values of the dual variables, which lie closer to the optimum. Similar effects can be observed for all distributed optimization algorithms. Therefore, all algorithms are parametrized more conservatively for the MIQP benchmark problems compared to the QP problems to obtain better primal solutions.

The step size/ trust region parameter was set to $\alpha^{(0)} = 3 \times 10^{-4}$ and the regularization parameter for ADMM was set to $\rho^{(0)} = 10^{-3}/N_s$. The age parameter of NAPS was set to $\tau = 1.5 \times n_{\text{reg,min}}$. The bounds for the covariance-based step size constraints were set to $\underline{s}_l = n_{\mathbf{b}} \times 10^{-8}$, $\overline{s}_l = n_{\mathbf{b}} \times 10^{-4}$ and $\underline{\gamma} = 0.1$. The remaining parameters remained unchanged compared to the QP benchmark problems. All parameters were set by trial and error, to find the parameters that result in the most converged benchmark problems with the best primal objective values. All parameters are summarized in Tab. 6.1.

As discussed in Sec. 6.2.1, the distributed optimization algorithms try to solve a primal problem with the contracted right-hand side $\overline{\mathbf{b}}$. However, the algorithms are terminated prematurely when the original problem is feasible with respect to the original right-hand side $\mathbf{b}$. In this case, the dual variables might not have converged to a stationary value, as they are updated based on the tightened problem. Therefore, only the primal residual is used as a convergence criterion for the MIQP benchmark problems to avoid unnecessary iterations. It should be noted that waiting for the dual variables to converge to a stationary value can also deteriorate the primal solution, similar to an aggressive parametrization.

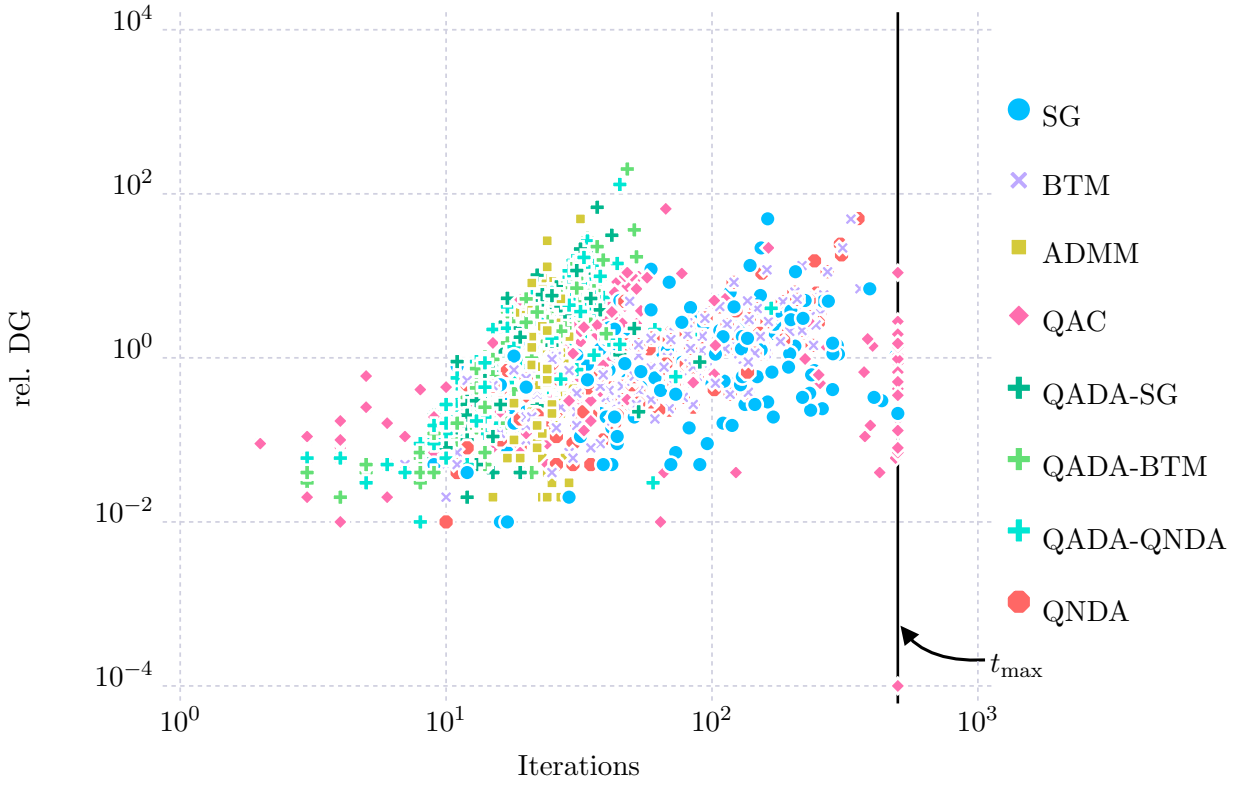


Figure 6.6: Relative duality gaps (rel. DG) of the MIQP benchmark problems upon termination for the examined algorithms. The value of the rel. DG for not converged runs has no meaning, as it corresponds to an infeasible primal solution (cf. Tab. C.2) (from [YWW+23]).

Table 6.3: Summary of the results for the distributed optimization of the MIQP benchmark problems (mean values of the converged instances only), $\bar{t}$: mean number of iterations until convergence, $\overline{T_{\text{comp}}}$: mean computation time of converged runs (in s), $\overline{\text{rel. DG}}$: mean relative duality gap of converged runs (in %), $\%_c$: percentage of converged runs within $t_{\max}$ iterations.

Algorithm	$\bar{t}$	$\overline{T_{\text{comp}}}$	$\overline{\text{rel. DG}}$	$\%_c$
SG	86.69	69.88	**1.66**	99.5
BTM	80.52	65.41	**1.66**	**100**
ADMM	25.06	21.16	2.22	**100**
QAC	59.40	52.83	2.13	86.0
QADA-SG	**19.37**	**18.37**	2.54	**100**
QADA-BTM	20.78	18.62	3.53	**100**
QADA-QNDA	22.20	22.76	2.86	**100**
QNDA	79.90	74.91	1.73	**100**

6.2.3 Results for distributed MIQPs

The MIQP benchmark problems were solved using the same algorithms as for the QP problems. The results are illustrated in Fig. 6.6 and summarized in Tab. 6.3. Instead of

depicting the primal residual (which is 0 for converged runs), Fig. 6.6 shows the relative duality gap,

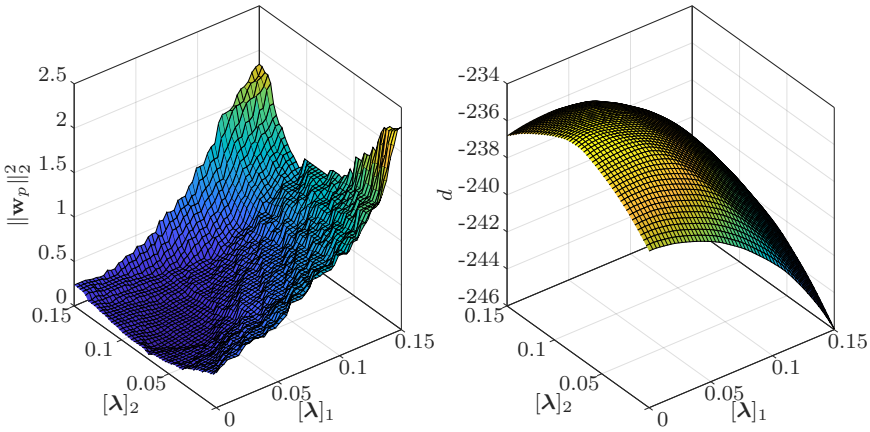
$$\text{rel. DG} = 100 \cdot \left(1 - \frac{d(\boldsymbol{\lambda})}{\sum_{i \in \mathcal{I}} f_i(\mathbf{x}_i^*(\boldsymbol{\lambda}))}\right) \tag{6.11}$$

for all benchmark problems, i.e., the relative difference between the objective value of a feasible primal solution obtained for a value of the dual variables $\boldsymbol{\lambda}$ and the corresponding value of the dual function. As weak duality still holds, the value of the dual function provides a lower bound on the global optimum of the primal problem. Thus, the relative duality gap is useful to prove a worst-case distance of a found solution to the global optimum. As can be seen, most algorithms can solve all benchmark problems, except for the subgradient method (which solves all but one) and QAC. Out of the considered algorithms, QADA exhibits the best performance, both in terms of computation time and required iterations. QADA here shows a significantly superior performance when compared to QNDA.

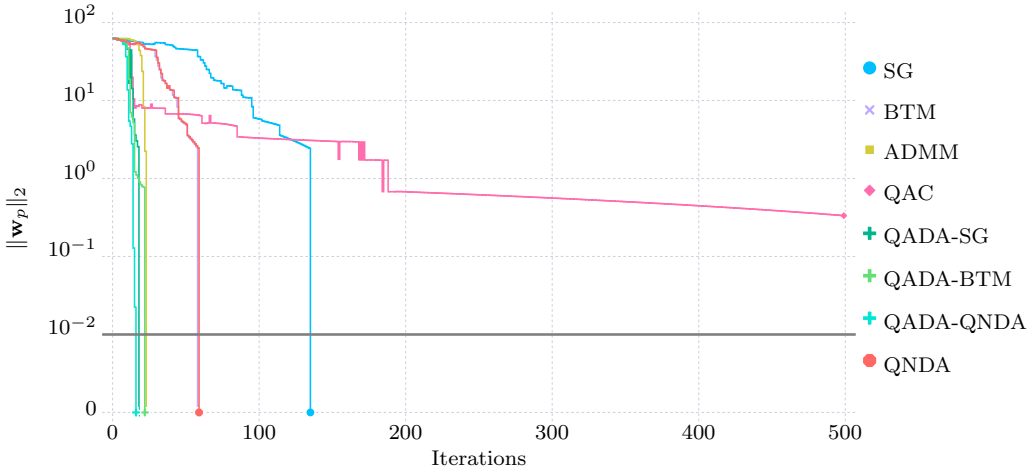
The results indicate that the primal residual cannot be approximated well as a quadratic function in all cases, leading to poor performance of the QAC algorithm. One such instance is shown in Fig. 6.7 for benchmark problem $\text{MIQP}^{(7)}_{(100,2)}$. The surface plots in Fig. 6.7a show that the primal residual $\|\mathbf{w}_p\|_2^2$, which is approximated by QAC, is nonsmooth and nonconvex. In comparison, the dual function is always concave and the nonsmoothness is less profound. Therefore, QADA and QNDA can compute a better smooth approximation and handle the nonsmoothness through the bundle cuts. This is reflected in the convergence of the algorithms. QADA converges quickly to a feasible primal solution. ADMM converges in a similar number of iterations. While QNDA does also converge, it does so in the same number of iterations as BTM and is slower than QADA and ADMM. Finally, QAC is not able to converge within the allowed number of iterations.

For integer problems, the optimal duality gap tends to decrease as the number of subproblems increases [VEG+16]. This also holds for the nonsmoothness of the response surfaces [WRE20]. Thus, the performance of the approximation-based algorithms tends to improve for larger problems. An example is shown in Fig. 6.8 for benchmark problem $\text{MIQP}^{(9)}_{(500,5)}$ where all algorithms converge to a feasible solution. The evolution of the primal residual (Fig. 6.8a) and the dual variables (Fig. 6.8b) indicate that QAC tends to oscillate, leading to a slower convergence. Fig. 6.8 also shows that BTM and QNDA converge slower than QADA and ADMM. This holds for the majority of the MIQP benchmark problems. Interestingly, the increased number of subproblems also affects the subgradient method, which tends to perform better for larger MIQP problem instances.

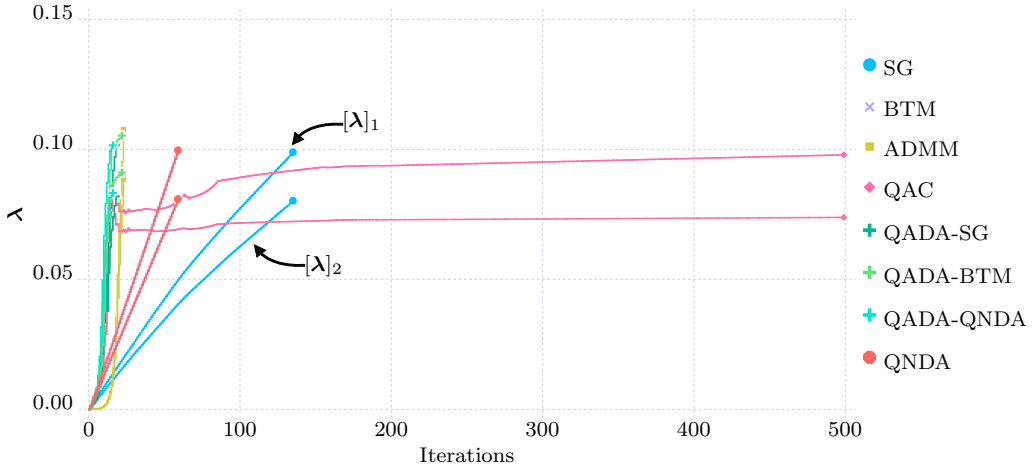
As discussed in Chapter 1, one reason for employing distributed optimization is to preserve privacy between the subproblems. However, another reason might be the computational performance of the system-wide optimization problem. This aspect is relevant for large-scale mixed-integer problems, where a centralized monolithic solution can become in-



(a) Surface plots of the squared primal residual $\|\mathbf{w}_p(\boldsymbol{\lambda})\|_2^2$ (left) and the dual function $d(\boldsymbol{\lambda})$ (right).

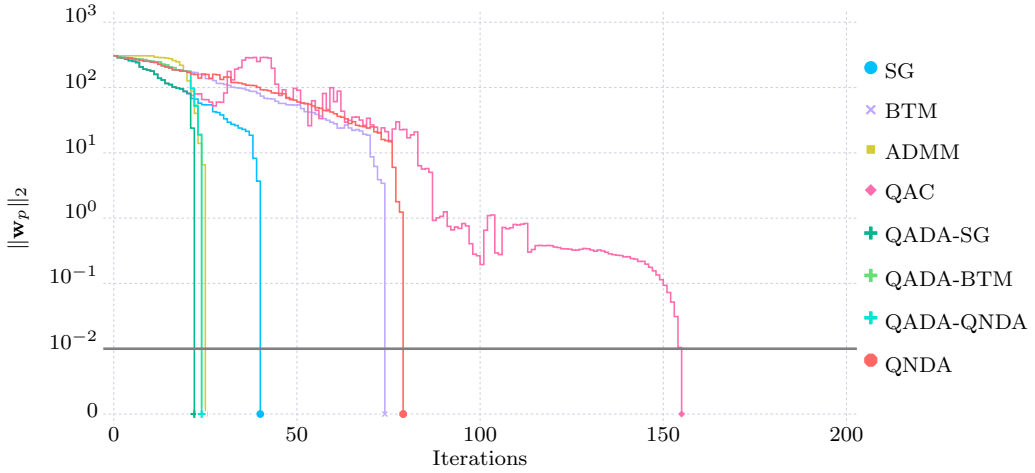


(b) Evolution of the primal and dual residuals.

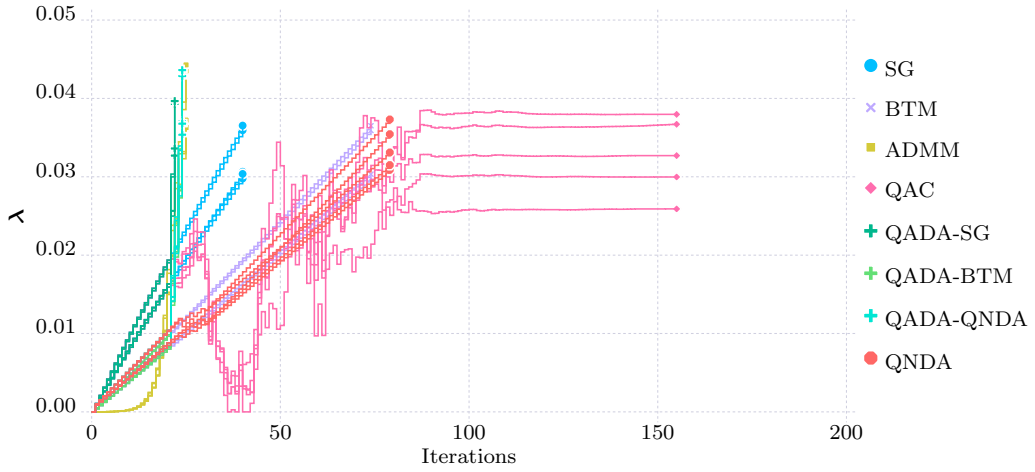


(c) Evolution of the dual variables.

Figure 6.7: Results of the distributed optimization of problem $\text{MIQP}^{(7)}_{(100,2)}$ (from [YWW+23]).



(a) Evolution of the primal residual.



(b) Evolution of the dual variables.

Figure 6.8: Results of the distributed optimization of problem $\text{MIQP}^{(9)}_{(500,5)}$ (from [YWW+23]).

tractable. Decomposing a large-scale problem into smaller subproblems and solving them in a distributed manner can lead to significant computational savings. A main appeal of state-of-the-art MIP solvers is that even when the global optimum is not found within the desired computation time, a worst-case distance to this optimum can be inferred through the relative integrality gap (rel. IG). The same holds for dual decomposition-based distributed optimization algorithms, where the distance of a found feasible primal solution to the global optimum is bounded by the duality gap. MIPs are always nonconvex due to the integrality constraints, meaning that strong duality does not hold. However, weak duality is always satisfied and provides bounds on the global optimum.

To assess the quality of the solutions obtained by the dual decomposition-based distributed optimization algorithms, they were compared to solutions obtained through a central optimization of the system-wide problem using the commercial solver Gurobi for the benchmark problems with $N_s = 500$ and $n_{\mathbf{x}} = 5$. A time limit of one hour (3600 s) was set for the central optimization. The results are shown in Tab. 6.4. Remarkably, Gurobi was

Table 6.4: Computational results for all MIQP benchmark problems with $N_s = 500$ and $n_{\mathbf{x}} = 5$.

	rel. Gap [%]	T_{comp} [s]	rel. Gap [%]	T_{comp} [s]	rel. Gap [%]	T_{comp} [s]
	Central		SG		BTM	
$\text{MIQP}^{(1)}_{(500,5)}$	3.70	3600	0.77	22.67	0.82	50.55
$\text{MIQP}^{(2)}_{(500,5)}$	2.87	3600	0.43	12.97	0.35	30.91
$\text{MIQP}^{(3)}_{(500,5)}$	3.74	3600	0.31	29.18	0.83	56.29
$\text{MIQP}^{(4)}_{(500,5)}$	3.81	3600	1.22	2.42	0.03	6.48
$\text{MIQP}^{(5)}_{(500,5)}$	3.00	3600	0.05	148.11	2.52	200.63
$\text{MIQP}^{(6)}_{(500,5)}$	3.51	3600	0.03	162.75	3.42	208.11
$\text{MIQP}^{(7)}_{(500,5)}$	3.15	3600	4.78	20.23	0.36	43.17
$\text{MIQP}^{(8)}_{(500,5)}$	3.22	3600	5.36	137.68	2.01	181.13
$\text{MIQP}^{(9)}_{(500,5)}$	3.53	3600	3.41	33.23	0.81	61.12
$\text{MIQP}^{(10)}_{(500,5)}$	3.33	3600	0.59	66.4	0.96	106.82
Mean	**3.39**		**1.21**	**63.56**	**1.21**	**94.52**
	ADMM		QAC		QADA-SG	
$\text{MIQP}^{(1)}_{(500,5)}$	0.80	22.88	–	465.69	0.92	18.72
$\text{MIQP}^{(2)}_{(500,5)}$	0.55	27.07	0.37	12.98	0.37	12.98
$\text{MIQP}^{(3)}_{(500,5)}$	1.04	21.65	1.03	33.40	1.22	19.49
$\text{MIQP}^{(4)}_{(500,5)}$	0.04	18.93	0.05	2.43	0.05	2.48
$\text{MIQP}^{(5)}_{(500,5)}$	4.76	23.46	4.28	39.72	4.78	30.65
$\text{MIQP}^{(6)}_{(500,5)}$	5.36	23.45	5.15	41.83	5.89	26.76
$\text{MIQP}^{(7)}_{(500,5)}$	0.39	21.69	0.36	19.73	0.70	18.32
$\text{MIQP}^{(8)}_{(500,5)}$	4.85	23.40	0.75	46.61	5.29	26.45
$\text{MIQP}^{(9)}_{(500,5)}$	1.21	23.43	0.67	350.01	1.07	19.30
$\text{MIQP}^{(10)}_{(500,5)}$	1.79	23.7	2,75	67.61	1.15	21.53
Mean	**2.08**	**22.97**	**1.71**	**68.26**	**2.14**	**19.67**
	QADA-BTM		QADA-QNDA		QNDA	
$\text{MIQP}^{(1)}_{(500,5)}$	0.90	21.86	0.85	21.93	0.77	54.13
$\text{MIQP}^{(2)}_{(500,5)}$	0.50	19.37	0.43	21.38	0.31	33.34
$\text{MIQP}^{(3)}_{(500,5)}$	1.22	23.06	1.15	23.20	0.83	67.61
$\text{MIQP}^{(4)}_{(500,5)}$	0.03	6.48	0.03	4.46	0.03	4.39
$\text{MIQP}^{(5)}_{(500,5)}$	3.96	31.68	5.54	31.66	2.69	245.26
$\text{MIQP}^{(6)}_{(500,5)}$	4.10	35.24	9.02	30.56	3.43	249.38
$\text{MIQP}^{(7)}_{(500,5)}$	0.66	22.54	0.59	21.74	0.36	50.78
$\text{MIQP}^{(8)}_{(500,5)}$	9.18	31.95	6.03	31.05	2.01	217.67
$\text{MIQP}^{(9)}_{(500,5)}$	1.38	21.37	1.40	23.79	0.87	77.00
$\text{MIQP}^{(10)}_{(500,5)}$	5.86	32.19	2.23	29.46	0.95	124.96
Mean	**2.78**	**24.57**	**2.73**	**23.92**	**1.22**	**112.45**

not able to solve any problem to global optimality within the time limit. In contrast, the dual decomposition-based algorithms all converge within much shorter computation times (except for problem $\text{MIQP}^{(1)}_{(500,5)}$ using QAC). Even though the found primal solutions are not provably globally optimal, the relative duality gap is usually better than the relative integrality gap provided by Gurobi. Thus, distributed optimization provides better bounds (and better primal solutions) for the examined benchmark problems. Among the

dual decomposition-based distributed optimization algorithms QADA-SG exhibits the best computation times, followed by ADMM. QADA-BTM and QADA-QNDA exhibit larger computation times, since the initial sampling steps are computationally more expensive, compared to simple subgradient updates.

6.3 General distributed convex problems

In Sec. 6.1 all of the subproblems were quadratic programs. In this section, more general convex problems of the form

$$\min_{\mathbf{x}_1,\dots,\mathbf{x}_{N_s}} \sum_{i\in\mathcal{I}} f_i(\mathbf{x}_i), \tag{6.12a}$$

$$\text{s. t. } \sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i = 0, \tag{6.12b}$$

$$\mathbf{x}_i^T\mathbf{G}_i\mathbf{x}_i \le p_i^2,\ \forall i\in\mathcal{I} \tag{6.12c}$$

$$\mathbf{x}_i^{\mathrm{LB}} \le \mathbf{x}_i \le \mathbf{x}_i^{\mathrm{UB}},\ \forall i\in\mathcal{I}. \tag{6.12d}$$

are considered. The objective functions $f_i(\mathbf{x}_i)$ are all convex functions (inside the feasible set of (6.12)) and are summarized in Tab. 6.5 alongside the bounds (6.12d) and the distributions from which their parameters were randomly drawn. The objective function for each subproblem is chosen randomly out of the considered convex functions with a uniform probability. The parameters of the system-wide constraints (6.12b) were drawn from the

Table 6.5: Used convex objective functions for the distributed convex programs.

Name	$f_i(\mathbf{x}_i)$	Bounds/Parameters
Affine	$\mathbf{c}_i^T\mathbf{x}_i + a_i$	$-10\cdot\mathbf{1} \le \mathbf{x}_i \le 10\cdot\mathbf{1}$ $a_i, [\mathbf{c}_i]_l \in \mathcal{N}(\mu=0,\sigma=1)$
Quadratic	$\frac{1}{2}\mathbf{x}_i^T\mathbf{H}_i\mathbf{x}_i + \mathbf{c}_i^T\mathbf{x}_i$	$-10\cdot\mathbf{1} \le \mathbf{x}_i \le 10\cdot\mathbf{1}$ See Sec. 6.1
Powers	$\sum_{l=1}^{n_{\mathbf{x}_i}} ([\mathbf{x}_i]_l + [\mathbf{c}_i]_l)^{[\mathbf{a}_i]_l}$	$-\frac{1}{2}\min_{l=1,\dots,n_{\mathbf{x}_i}} [\mathbf{c}_i]_l\cdot\mathbf{1} \le \mathbf{x}_i \le 10\cdot\mathbf{1}$ $[\mathbf{a}_i]_l, [\mathbf{c}_i]_l \in \mathcal{U}_c(1,5)$
Exponential	$\sum_{l=1}^{n_{\mathbf{x}_i}} \exp([\mathbf{c}_i]_l\cdot[\mathbf{x}_i]_l + [\mathbf{a}_i]_l)$	$-10\cdot\mathbf{1} \le \mathbf{x}_i \le 10\cdot\mathbf{1}$ $[\mathbf{a}_i]_l, [\mathbf{c}_i]_l \in \mathcal{N}(\mu=0,\sigma=1)$
Negative log	$-\sum_{j=1}^{n_{\mathbf{x}_i}} [\mathbf{c}_i]_j \log([\mathbf{x}_i]_j + [\mathbf{a}_i]_j)$	$-\frac{1}{2}\min_{j=l,\dots,n_{\mathbf{x}_i}} [\mathbf{c}_i]_l\cdot\mathbf{1} \le \mathbf{x}_i \le 10\cdot\mathbf{1}$ $[\mathbf{a}_i], [\mathbf{c}_i]_l \in \mathcal{U}_c(1,5)$
Negative entropy	$\sum_{j=1}^{n_{\mathbf{x}_i}} [\mathbf{a}_i]_j[\mathbf{x}_i]_j \log([\mathbf{x}_i]_j + [\mathbf{b}_i]_j)$	$-\frac{1}{2}\min_{l=1,\dots,n_{\mathbf{x}_i}} [\mathbf{c}_i]_l\cdot\mathbf{1} \le \mathbf{x}_i \le 10\cdot\mathbf{1}$ $[\mathbf{a}_i], [\mathbf{c}_i]_l \in \mathcal{U}_c(1,5)$

same distributions as for the distributed QP problems. Each subproblem is subject to individual convex constraints in the form of an ellipsoid around the origin (6.12c), with random parameters $\mathbf{G}_i = \mathbf{N}_i^T \mathbf{N}_i, [\mathbf{N}_i]_{l,j} \in \mathcal{N}(\mu = 0, \sigma = 1)$ and $p_i \in \mathcal{U}_c(1, 5)$. The number and size of the subproblems were varied as follows:

$$\text{Number of subproblems: } N_s = 2^m, m \in \{2, 3, \ldots, 8\},$$
$$\text{Number of variables: } n_{\mathbf{x}_i} \in \{2, 3, \ldots, 10\},\ N_s \geq n_{\mathbf{x}_i}.$$

All subproblems contain the same number of primal variables ($n_{\mathbf{x}_i} = n_{\mathbf{x}},\ \forall i \in \mathcal{I}$) and the number of system-wide constraints is equal to the number of variables of each subproblem, i.e., $n_{\mathbf{b}} = n_{\mathbf{x}}$. Ten benchmark problems were generated for each combination $(N_s, n_{\mathbf{b}})$, resulting in a total of 560 convex benchmark problems. Note that all benchmark problems are convex and satisfy Slater's condition, as $\mathbf{x} = \mathbf{0}$ is a strictly feasible solution. As the system-wide constraints (6.12b) are equalities, no nonnegativity constraints are imposed on the dual variables.

6.3.1 Parameter settings for distributed convex problems

The parameters for the distributed convex benchmarks were set to be equal to the ones for the distributed QPs (cf. Sec. 6.1.1 and Tab. 6.1). All parameters were set by trial and error, to find the parameters that result in the most converged benchmark problems. As the previous results showed that the performance of QADA is not heavily influenced by the algorithm used for the initial sampling phase, only the initialization with QNDA is considered in the following. For the sake of brevity, this is denoted by QADA instead of QADA-QNDA.

6.3.2 Results for distributed convex problems

The results for the distributed optimization of the convex benchmark problems are illustrated in Fig. 6.9 and summarized in Tab. 6.6. The mean computation time for a single update step of the primal and dual variables ($\overline{T_{\text{comp}}}/\bar{t}$) increases compared to the distributed QPs. This is because the solution of the general convex subproblems is computationally more expensive than that of the QPs. This is also because the problems with an affine and quadratic objective function can be solved by the commercial solver Gurobi, while the general convex problems are solved with IPOPT. This points to a benefit of distributed optimization, namely, that arbitrary solvers can be used for each subproblem [MKJ+17].

Tab. 6.6 shows that QNDA achieves the largest number of converged benchmark problems, requires the least number of iterations, and the second least computation time (after ADMM). In comparison, QADA performs rather poorly, only outperforming the subgradient method. Interestingly, QADA performs better for relatively few subproblems and

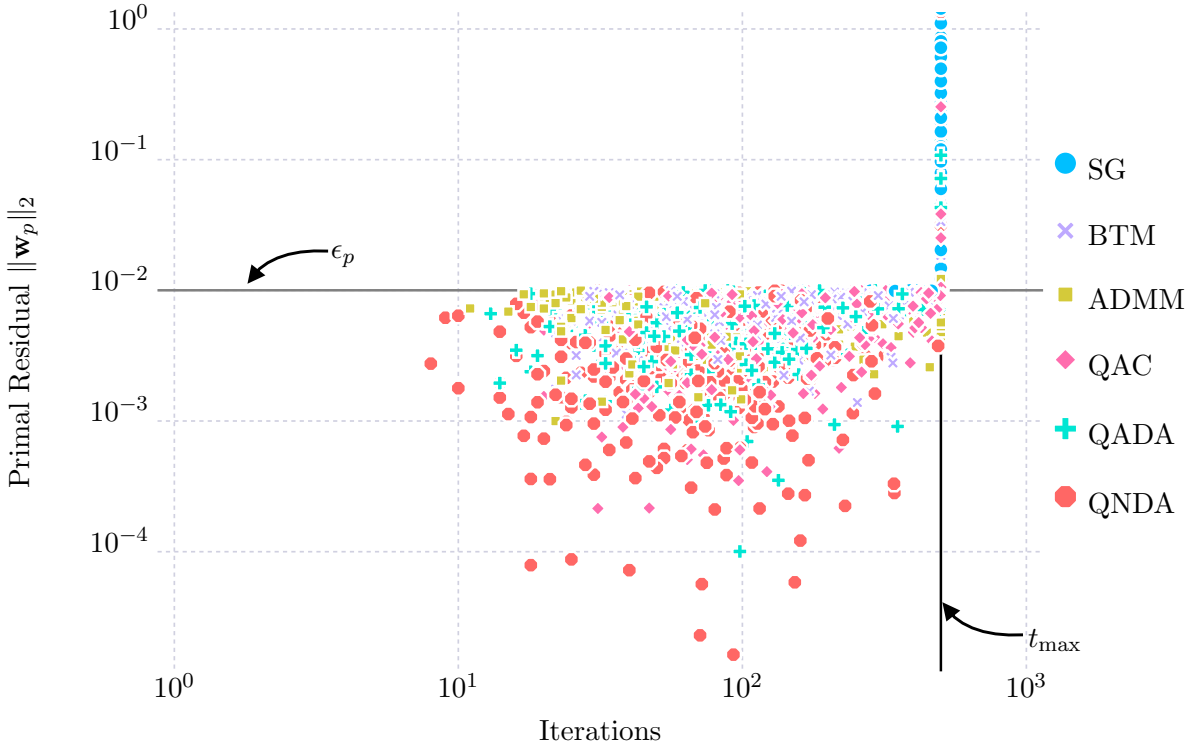


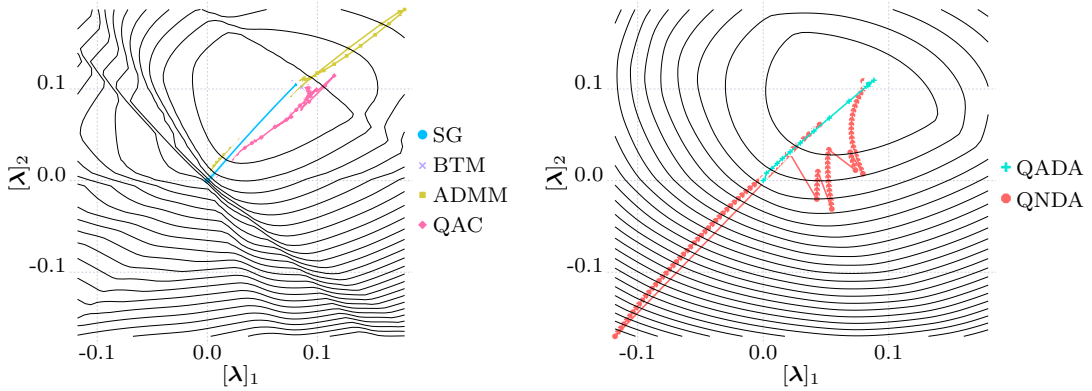
Figure 6.9: Values of the primal residuals upon termination for the distributed convex programs. Each data point represents an algorithm applied to a benchmark problem (cf. Tab. C.3) (from [YWW+23]).

system-wide constraints. One such instance is depicted in Fig. 6.10 for benchmark problem $\text{Conv}^{(6)}_{(64,2)}$. Here QADA requires the fewest iterations to converge. In contrast, QNDA takes multiple steps away from the optimum, significantly deteriorating its performance. Fig. 6.10a and 6.10c also show that ADMM overshoots in the beginning, which might indicate that a less aggressive tuning could lead to better convergence.

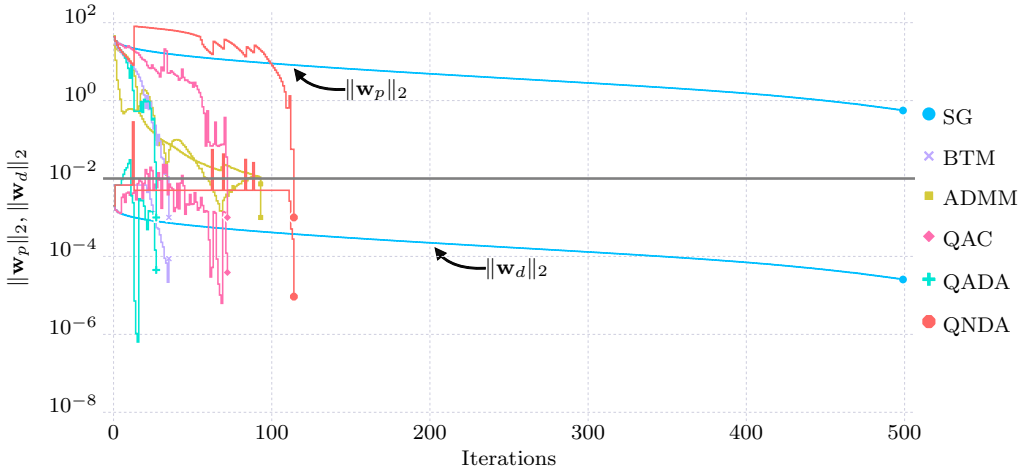
The QNDA algorithm outperforms all other algorithms for large benchmark problems. One such instance is depicted in Fig. 6.11 for problem $\text{Conv}^{(4)}_{(256,7)}$. No algorithm converges,

Table 6.6: Summary of the results for the distributed optimization of the convex benchmark problems (mean values of the converged instances only), $\bar{t}$: mean number of iterations until convergence, $\overline{T_{\text{comp}}}$: mean computation time of converged runs (in s), $\overline{\|\mathbf{w}_p\|_2}$: mean primal residual of converged runs ($\times 10^{-3}$), $\%_c$: percentage of converged runs within $t_{\max}$ iterations.

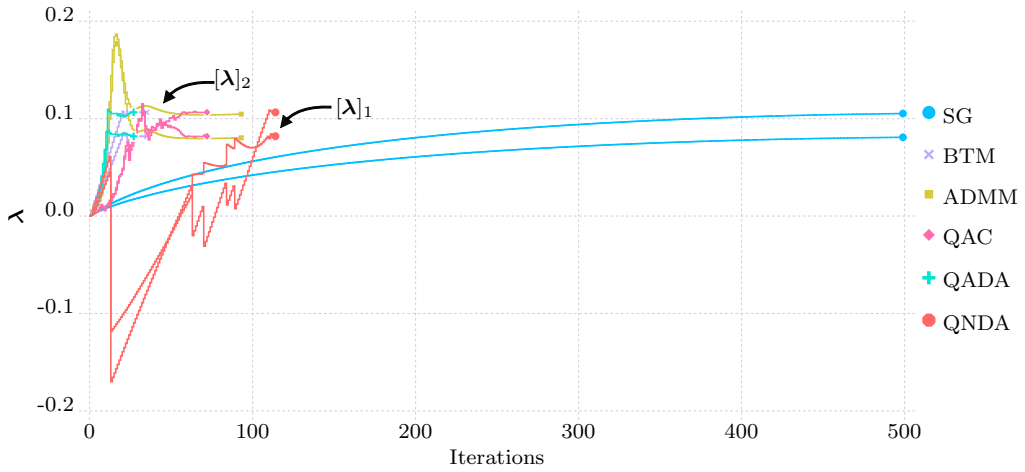
Algorithm	$\bar{t}$	$\overline{T_{\text{comp}}}$	$\overline{\|\mathbf{w}_p\|_2}$	$\%_c$
SG	342.51	310.45	9.72	46.07
BTM	160.29	139.29	7.8	78.26
ADMM	117.44	**104.97**	7.44	93
QAC	207.16	186.9	6.55	81.52
QADA	123.52	149.39	6.95	75.1
QNDA	**97.13**	114.71	**4.58**	**97.14**



(a) Contour plot of the squared primal residual $\|\mathbf{w}_p(\boldsymbol{\lambda})\|_2^2$ (left) and the dual function $d(\boldsymbol{\lambda})$ (right).



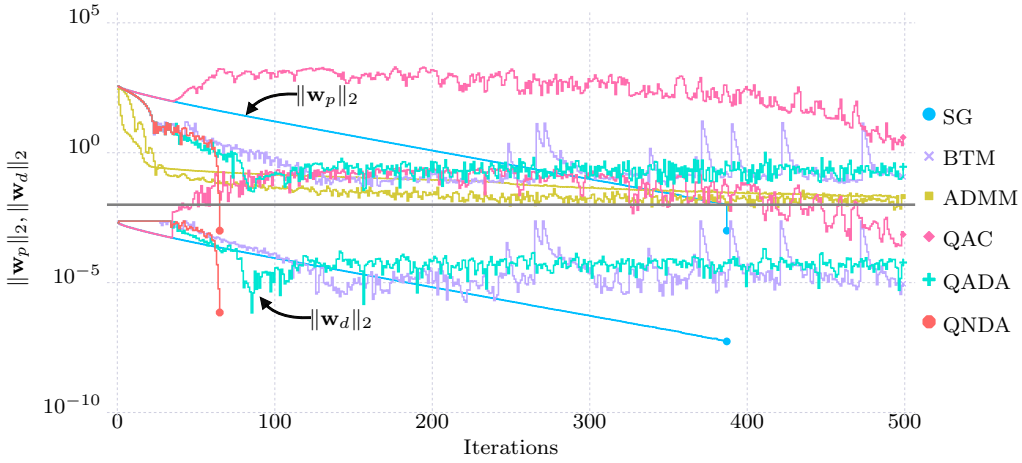
(b) Evolution of the primal and dual residuals.



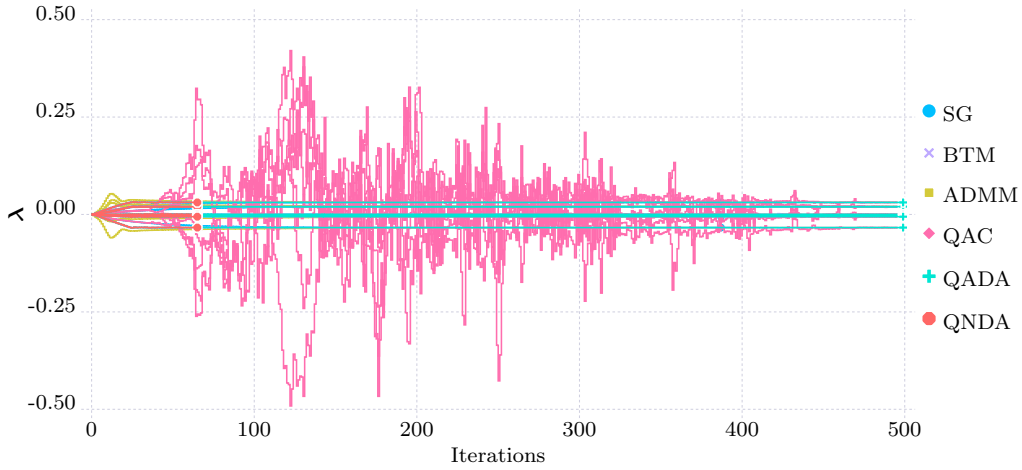
(c) Evolution of the dual varialbes.

Figure 6.10: Results of the distributed optimization of problem $\text{Conv}^{(6)}_{(64,2)}$ (from [YWW+23]).

except for the subgradient method and QNDA. QAC exhibits significant oscillations. The remaining algorithms (BTM, ADMM, and QADA) all converge to the vicinity of the optimum relatively quickly. However, they are not able to find the optimum. This is



(a) Evolution of primal and dual residuals.



(b) Evolution of the dual varialbes.

Figure 6.11: Results of the distributed optimization of problem $\text{Conv}^{(4)}_{(256,7)}$ (from [YWW+23]).

also illustrated in Fig. 6.11b, where it can be seen that all algorithms terminate close to the optimal dual variables. This behavior is indicative of an optimal solution lying close to or at a point of nondifferentiability. While the subgradient method converges after a large number of iterations, QNDA computes the solution more efficiently. Initially, the algorithm follows a similar path as BTM. However, while BTM is not able to converge, QNDA does so within a few iterations.

6.4 Conclusion

This chapter provided extensive numerical tests on a large set of benchmark problems for all previously discussed algorithms. The results show that algorithms based on smooth approximations of the dual function generally outperform the other algorithms, both in terms of the number of solved problems and the number of required iterations. While the performance of the QADA and QNDA algorithms is similar for distributed QPs, the

former shows superior performance for distributed MIQPs while the latter outperforms the other algorithms for distributed convex problems. The QADA algorithm is usually able to perform more aggressive update steps, leading to a faster convergence to a feasible primal solution in the case of the MIQP benchmark problems. The resulting relative duality gaps are larger than the ones obtained by the more conservative QNDA algorithm. However, note that the duality gap just provides a worst-case distance to the global optimum, while both algorithms converge to similar primal solutions in practice. In the case of the general convex benchmark problems the QNDA algorithm outperforms QADA due to the more local nature of its quadratic approximation. The QADA algorithm tries to compute a surrogate quadratic function via a regression problem, even though the underlying approximated dual function is not quadratic or even piece-wise quadratic. The QNDA algorithm relies more on the local curvature information of the dual function and is thus able to compute a better local approximation.

7 Application: Distributed Linear Model Predictive Control

Chapter 6 presented extensive numerical benchmarks for general distributed optimization problems. However, real-world optimization problems usually exhibit a more structured formulation. Therefore, this chapter presents how dual decomposition-based distributed optimization can be applied to linear constraint-coupled model predictive control (MPC) problems. These problems serve as an example of the application of distributed optimization to continuous convex optimization problems. Following a brief introduction to model predictive control different architectures and communication topologies for a distributed setting are presented. After decomposing the system-wide control problem into multiple subproblems by introducing dual variables a large number of randomly generated benchmark problems is solved using the subgradient method (SG), the bundle trust method (BTM), the alternating direction method of multipliers (ADMM) and the quasi-Newton dual ascent (QNDA) algorithm. The contents of this chapter have been partialy submitted for review in [YWR24].

7.1 Model predictive control

Model predictive control is an optimization-based control method whereby a model of the plant is used to predict its future behavior depending on its current state and the computed control inputs. MPC algorithms usually employ a discrete-time model to predict the evolution of the plant's states $\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}}$ depending on the current states and control inputs $\mathbf{u} \in \mathbb{R}^{n_\mathbf{u}}$. A typical linear[3] MPC optimization problem is given by eq. (7.1),

$$\min_{\mathbf{x}^{0:N_p},\mathbf{u}^{0:N_p-1}} \; J^f(\mathbf{x}^{N_p}) + \sum_{k=0}^{N_p-1} J(\mathbf{x}^k, \mathbf{u}^k), \tag{7.1a}$$

$$\text{s.t. } \mathbf{x}^{k+1} = \mathbf{A}\mathbf{x}^k + \mathbf{B}\mathbf{u}^k, \; k = 0, \dots, N_p - 1, \tag{7.1b}$$

$$\mathbf{x}^0 = \tilde{\mathbf{x}}(t_0), \tag{7.1c}$$

$$\mathbf{x}^k \in \mathcal{X} \subset \mathbb{R}^{n_\mathbf{x}}, \; k = 0, \dots, N_p, \tag{7.1d}$$

$$\mathbf{u}^k \in \mathcal{U} \subset \mathbb{R}^{n_\mathbf{u}}, \; k = 0, \dots, N_p - 1. \tag{7.1e}$$

[3]Linearity in this case refers to the dynamics of the plant, not the class of the underlying optimization problem.

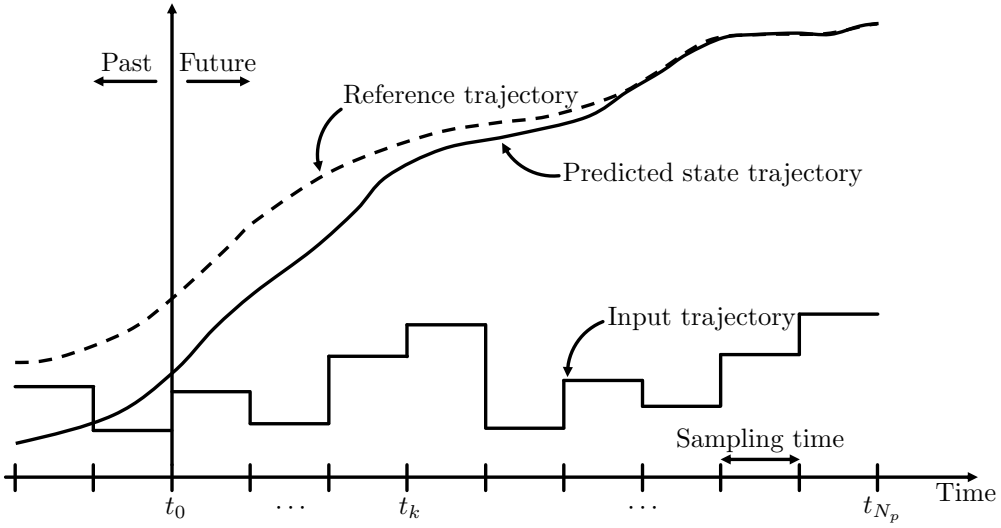


Figure 7.1: Schematic illustration of model predictive control.

In MPC time is usually discretized into equal time intervals with a sampling time T_s, i.e., into time points $t_k = t_0 + k \cdot T_s$, where t_0 is the current time. The notation $\mathbf{x}^k$ and $\mathbf{u}^k$ refers to the states and inputs of the plant at time t_k respectively. Constraint (7.1b) represents a discrete-time model used to predict the state of the plant $\mathbf{x}^{k+1}$ at the next sampling time from the current states $\mathbf{x}^k$ and the used control input $\mathbf{u}^k$ with $\mathbf{A} \in \mathbb{R}^{n_\mathbf{x} \times n_\mathbf{x}}$ and $\mathbf{B} \in \mathbb{R}^{n_\mathbf{x} \times n_\mathbf{u}}$. The future states and inputs are computed over a prediction horizon N_p. Note that the control inputs must not necessarily be computed over the entire prediction horizon, especially if solving the MPC optimization problem is computationally expensive. Instead, the control inputs can be computed over a control horizon $N_c \leq N_p$ and then be kept constant until the end of the prediction horizon, i.e., $\mathbf{u}^{N_c} = \mathbf{u}^{N_c+1} = \cdots = \mathbf{u}^{N_p-1}$. In this work $N_p = N_c$ is assumed. Constraint (7.1c) represents the initial conditions of the plant, whereby $\tilde{\mathbf{x}}(t_0)$ denotes the measured states at time t_0. Depending on the plant the actual states might not be measurable. In these cases, state estimation techniques might be used to estimate the plant's states from the measured output. However, state estimation is outside the scope of this thesis. Necoara et al. provide an overview of state estimation in a distributed setting [NND11]. A main advantage of MPC compared to other control approaches is that constraints on the states and inputs can be explicitly considered within the optimization problem [MRR+00]. These constraints are summarized by (7.1d) and (7.1e) respectively. Fig. 7.1 schematically illustrates the concept of MPC.

At each time step an input trajectory is computed over a prediction horizon N_p by solving the open-loop optimal control problem (7.1). A common objective in MPC is to track a given reference trajectory over the prediction horizon while also minimizing the control inputs, i.e.,

$$J(\mathbf{x}^k, \mathbf{u}^k) = (\mathbf{x}^k - \mathbf{x}^{\mathrm{ref},k})^T \mathbf{H}_\mathbf{x} (\mathbf{x}^k - \mathbf{x}^{\mathrm{ref},k}) + \mathbf{u}^{k,T} \mathbf{H}_\mathbf{u} \mathbf{u}^k, \tag{7.2}$$

where $\mathbf{H}_\mathbf{x} \in \mathbb{R}^{n_\mathbf{x} \times n_\mathbf{x}}$ and $\mathbf{H}_\mathbf{u} \in \mathbb{R}^{n_\mathbf{u} \times n_\mathbf{u}}$ are symmetric positive (semi-) definite weighting matrices. Eq. (7.2) is often referred to as stage costs, while $J^f(\mathbf{x}^{N_p})$ is referred to as

terminal cost. A common terminal cost is the deviation of the states from their reference at the end of the prediction horizon, i.e.,

$$J^f(\mathbf{x}^{N_p}) = (\mathbf{x}^{N_p} - \mathbf{x}^{\mathrm{ref},N_p})^T \mathbf{H}_{\mathbf{x}}^f (\mathbf{x}^{N_p} - \mathbf{x}^{\mathrm{ref},N_p}). \tag{7.3}$$

The design of the terminal cost plays an important role regarding the control performance and stability [MRR+00]. These aspects are also outside the scope of this thesis. Another main appeal of MPC is that control objectives beyond classical reference tracking can be incorporated into the optimization problem [Eng07]. For instance, the economic performance of the process can be directly optimized, giving rise to economic MPC (EMPC). An extensive review of EMPC is provided by Ellis et al. [EDC14].

After solving the open-loop optimal control problem the first input $\mathbf{u}^0$ is applied to the system. At the next sampling point, the new state is measured (or estimated) and the procedure is repeated.

7.2 Distributed linear model predictive control

MPC requires the solution of an optimization problem at each sampling point. However, this might be impractical if the underlying optimization problem becomes too large and therefore computationally too expensive to be solved within the available sampling time, or if the overall system consists of multiple subsystems and not all required information is centrally available. These issues can be addressed by decomposing a system-wide MPC problem and solving it in a distributed manner.

Two types of distributed linear MPC (DMPC) problems are mainly examined in the literature. In the first case, the system-wide MPC problem consists of $\mathcal{I} = \{1, \ldots, N_s\}$ subsystems coupled in their dynamics. These MPC problems can be expressed as

$$\min_{\mathbf{x}^{0:N_p}, \mathbf{u}^{0:N_p-1}} \sum_{i \in \mathcal{I}} \left[J_i^f(\mathbf{x}_i^{N_p}) + \sum_{k=0}^{N_p-1} J_i(\mathbf{x}_i^k, \mathbf{u}_i^k) \right], \tag{7.4a}$$

$$\text{s.t. } \mathbf{x}_i^{k+1} = \sum_{j \in \mathcal{I}} \mathbf{A}_{ij} \mathbf{x}_j^k + \mathbf{B}_{ij} \mathbf{u}_j^k, \ \forall i \in \mathcal{I}, k = 0, \ldots, N_p - 1, \tag{7.4b}$$

$$\mathbf{x}_i^0 = \tilde{\mathbf{x}}(t_0), \ \forall i \in \mathcal{I}, \tag{7.4c}$$

$$\mathbf{x}_i^k \in \mathcal{X}_i \subset \mathbb{R}^{n_{\mathbf{x}_i}}, \ \forall i \in \mathcal{I}, k = 0, \ldots, N_p, \tag{7.4d}$$

$$\mathbf{u}_i^k \in \mathcal{U}_i \subset \mathbb{R}^{n_{\mathbf{u}_i}}, \ \forall i \in \mathcal{I}, k = 0, \ldots, N_p - 1, \tag{7.4e}$$

where $\mathbf{x}_i$ and $\mathbf{u}_i$ denote the states and inputs of subsystem i respectively. The states and inputs of all subsystems are summarized in $\mathbf{x} \coloneqq [\mathbf{x}_1^T, \ldots, \mathbf{x}_{N_s}^T]^T$ and $\mathbf{u} \coloneqq [\mathbf{u}_1^T, \ldots, \mathbf{u}_{N_s}^T]^T$. Each subsystem possesses its individual terminal and stage cost functions $J_i^f(\mathbf{x}_i^{N_p})$ and $J_i(\mathbf{x}_i^k, \mathbf{u}_i^k)$ respectively. The matrices $\mathbf{A}_{ij} \in \mathbb{R}^{n_{\mathbf{x}_i} \times n_{\mathbf{x}_j}}$ and $\mathbf{B}_{ij} \in \mathbb{R}^{n_{\mathbf{x}_i} \times n_{\mathbf{u}_j}}$ describe the influence of the states and inputs of subsystem j on the dynamics of subsystem i.

In the second case, the subsystems are coupled through their constraints

$$\min_{\mathbf{x}^{0:N_p},\mathbf{u}^{0:N_p-1}} \sum_{i\in\mathcal{I}} \left[J_i^f(\mathbf{x}_i^{N_p}) + \sum_{k=0}^{N_p-1} J_i(\mathbf{x}_i^k, \mathbf{u}_i^k) \right], \tag{7.5a}$$

$$\text{s.t. } \mathbf{x}_i^{k+1} = \mathbf{A}_i\mathbf{x}_i^k + \mathbf{B}_i\mathbf{u}_i^k,\ \forall i \in \mathcal{I}, k = 0,\dots,N_p-1, \tag{7.5b}$$

$$\mathbf{x}_i^0 = \tilde{\mathbf{x}}(t_0),\ \forall i \in \mathcal{I}, \tag{7.5c}$$

$$\mathbf{x}_i^k \in \mathcal{X}_i \subset \mathbb{R}^{n_{\mathbf{x}_i}},\ \forall i \in \mathcal{I}, k = 0,\dots,N_p, \tag{7.5d}$$

$$\mathbf{u}_i^k \in \mathcal{U}_i \subset \mathbb{R}^{n_{\mathbf{u}_i}},\ \forall i \in \mathcal{I}, k = 0,\dots,N_p-1, \tag{7.5e}$$

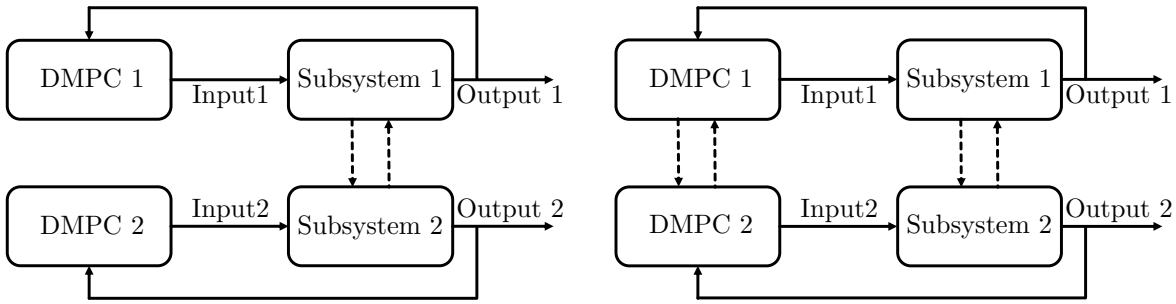
$$\sum_{i\in\mathcal{I}} \mathbf{R}_i\mathbf{u}_i^k \leq \mathbf{r}_{\max}^k,\ k = 0,\dots,N_p-1. \tag{7.5f}$$

The dynamics of each subsystem (7.5b) are decoupled. The coupling of the subsystem is represented by the system-wide constraints (7.5f), which can be interpreted as constraints on the availability of shared limited resources. The matrices $\mathbf{R}_i \in \mathbb{R}^{n_\mathbf{r}\times n_{\mathbf{u}_i}}$ map the subsystems inputs to the corresponding resource consumption or production, while $\mathbf{r}_{\max}^k \in \mathbb{R}^{n_\mathbf{r}}$ denotes the maximum availability of resources. Note that in the resource constraints, a positive sign indicates consumption while a negative sign indicates the production of a resource.

This thesis focuses on the second type of DMPC problems (7.5), which can be solved within a dual decomposition-based distributed optimization framework. Note that DMPC problems of the form (7.4) can also be solved by using dual decomposition, which will be briefly discussed in Sec. 7.2.4.

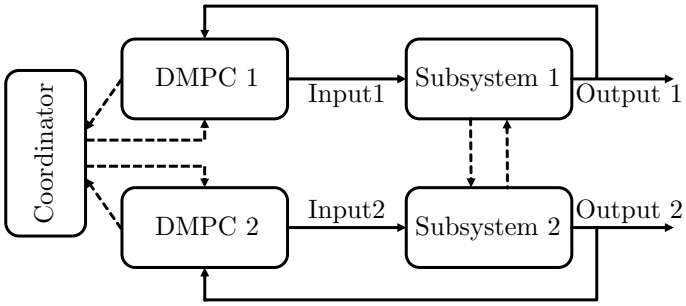
7.2.1 DMPC architectures

Apart from the type of coupling between the subsystems, the employed DMPC method strongly depends on the allowed communication. Several DMPC architectures are reviewed by, e.g., Scattolini [Sca09] and Christofides et al. [CSdlP+13], some of which are depicted in Fig. 7.2. For the sake of simplicity, the influence of the output of a DMPC controller on other subsystems is omitted at this point, i.e., $\mathbf{B}_{ij} = \mathbf{0}$ for $i \neq j$ (cf. eq. 7.4b). Fig. 7.2a depicts a setting in which the coupling of the subsystems is not explicitly considered by the DMPC controllers, i.e., there is no communication between them. This is often referred to as decentralized MPC. In this setting, the coupling between the subsystems can be regarded as an external disturbance that can be compensated by the individual controllers. If the coupling between subsystems is strong, decentralized control might exhibit poor performance. Furthermore, decentralized DMPC is not suitable for constraint-coupled problems as the satisfaction of system-wide constraints cannot be guaranteed without some amount of information exchange. Christofides et al. provide an overview of decentralized MPC [CSdlP+13].



(a) Decentralized MPC.

(b) Cooperative DMPC.



(c) Hierarchical DMPC.

Figure 7.2: Different distributed model predictive control structures for two subsystems with two controllers (adapted from [Sca09]).

If coupling between subsystems must be accounted for explicitly by the DMPC controllers some information must be exchanged. Fig. 7.2b depicts a setting in which direct communication between the DMPC controllers is allowed. In this setting, the controllers can exchange information like predicted state and input trajectories, objective values, constraints, etc. DMPC algorithms relying on this architecture can be classified according to the number of communication rounds per MPC iteration, i.e., how often information is exchanged before a control action is applied to the subsystems. Camponogara et al. provide a general overview of both types of communication [CJK+02]. The main factor determining the possible number of communication rounds between the controllers is the ratio between the computational cost of the solution of the individual DMPC problems and the sampling time of the overall MPC problem. While DMPC with multiple communication rounds, often also referred to as agent negotiation [MdlPC+11b], usually results in better control performance and under certain conditions enables the convergence to the optimal solution of the system-wide MPC problem, it is usually computationally more expensive than single communication DMPC since the subsystems have to solve multiple optimization problems in each iteration. If short sampling times must be considered, performing multiple communication rounds is usually not feasible. This is the case, e.g., in the control of robotic manipulators with collision constraints, where only the predicted state trajectory is communicated to the neighboring DMPC controllers [GKW+22].

Finally, Fig. 7.2c depicts a setting where no direct communication between the DMPC controllers is allowed. This leads to a hierarchical control structure where the individual

controllers share information with a coordinator. The coordinator aggregates the responses and communicates price signals to the DMPC controllers. This setting is considered in dual decomposition-based DMPC, where the coordinator communicates the dual variables to the subsystems. Hierarchical DMPC is especially suitable for constraint-coupled problems where direct information exchange between the controllers is not intended, e.g., due to confidentiality reasons. Furthermore, a hierarchical DMPC approach always requires multiple communication rounds between the subsystems and the coordinator. Thus, this approach is only suitable for systems with slow dynamics and large sampling times, where the underlying MPC problems can be solved multiple times within one sampling period until a feasible solution is found, e.g., for chemical plants.

The next section provides a brief overview of related work on DMPC, mainly in cooperative and hierarchical settings.

7.2.2 DMPC literature review

Distributed control for large-scale systems has been an active field of research since the 1970s [SVA75]. Extensive reviews on DMPC are provided by, e.g., Scattolini [Sca09], Necoara et al. [NND11], Chrisofides et al. [CSdlP+13], Negenborn and Maestre [NM14] and Müller and Allgöwer [MA17].

Stewart et al. present a cooperative DMPC algorithm that can be interpreted as a suboptimal system-wide MPC, in turn proving stability [SVR+10]. Maestre et al. propose a cooperative DMPC algorithm for two agents with two communication rounds per sampling period [MdlPC11a]. They also propose a cooperative DMPC scheme with multiple agents whereby in each communication round a random set of agents proposes a control action, which is accepted by their neighbors if it improves the overall control objective [MdlPC+11b]. Zheng et al. present a DMPC algorithm for a network of interacting systems with direct communication between the controllers [ZLQ12].

Dual decomposition-based DMPC has been applied to systems coupled in their dynamics and constraints. Giselsson and Rantzer employ dual decomposition for systems coupled in their dynamics and present a stopping criterion that guarantees bounded suboptimality and asymptotic stability [GR10]. Giselsson et al. also present an accelerated gradient method for dual decomposition-based DMPC where the objective function includes $\mathcal{L}_1$-norm penalty terms [GDK+13]. In [GR13] Giselsson and Rantzer present a stopping criterion based on adaptive constraint tightening that allows for an early termination of dual decomposition-based DMPC problems for dynamically coupled systems. Köhler et al. examine recursive feasibility for premature termination of the dual decomposition-based distributed optimization algorithm [KMA19]. Doan et al. employ dual decomposition and use primal averaging and constraint tightening to ensure convergence of DMPC problems with both coupled dynamics and constraints [DKD11].

Biegel et al. show how MPC can be used for congestion management in an energy grid comprised of intelligent consumers with energy storage capabilities [BSB+12]. This problem is extended into a DMPC setting in [BAS+12]. Biegel et al. also show how dual decomposition-based DMPC can be generally applied to constraint-coupled systems [BSA14]. Pflaum et al. compare primal and dual decomposition-based DMPC for the distributed control in smart districts [PAL14]. In [PAL15] they examine the scalability of dual decomposition-based DMPC using a bundle method based on the number of subsystems, the number of constraints, and the maximum size of the bundle.

Razzanelli et al. use DMPC for the management of the energy flow within a network of microgrids using a subgradient method [RCP+20]. Eser et al. use ADMM to distributedly control building energy systems [ESK+22]. Maxeiner and Engell use ADMM for DMPC of semi-batch reactors coupled in their inputs through shared limited resources [ME17]. This work is extended in [ME20], where the subgradient method, ADMM, and ALADIN are extensively compared for these types of problems. Houska and Shi provide a tutorial on how ALADIN can be employed for DMPC [HS22]. Yfantis et al. apply the QADA algorithm to DMPC problems with shared limited resources, comparing its performance to the subgradient method [YGW+22].

Conte et al. study computational aspects of dual decomposition-based DMPC using an accelerated gradient method and ADMM [CSZ+12]. More specifically they examine the influence of the coupling strength, stability, the initial state, and the network size and topology on the computational performance of the DMPC. Stomberg et al. compare the performance of six algorithms for DMPC of dynamically coupled systems, namely the subgradient method, an accelerated subgradient method, ADMM, a distributed active set algorithm, an essentially decentralized interior point method, and Jacobi iterations [SEF22].

Other proximal algorithms, other than ADMM, can also be used for DMPC. Halvgaard et al. use the Douglas-Rachford splitting method for the coordination of smart energy systems [HVP+16]. An extensive collection of DMPC algorithms and applications can be found in the textbook by Maestre and Negenborn [MN14].

7.2.3 Dual decomposition of linear constraint-coupled DMPC problems

In this section, the system-wide constraint-coupled MPC problem is decoupled using dual decomposition. By introducing dual variables $\boldsymbol{\lambda}^k \in \mathbb{R}^{n_\mathrm{r}}$ at each sampling time k the Lagrange function can be defined,

$$\mathcal{L}(\mathbf{x}^{0:N_p}, \mathbf{u}^{0:N_p-1}, \boldsymbol{\lambda}^{0:N_p-1}) = \sum_{i\in\mathcal{I}} \underbrace{\left[J_i^f(\mathbf{x}_i^{N_p}) + \sum_{k=0}^{N_p-1} [J_i(\mathbf{x}_i^k, \mathbf{u}_i^k) + \boldsymbol{\lambda}^{k,T}\mathbf{R}_i\mathbf{u}_i^k] \right]}_{=:\mathcal{L}_i(\mathbf{x}_i^{0:N_p}, \mathbf{u}_i^{0:N_p-1}, \boldsymbol{\lambda}^{0:N_p-1})} - \sum_{k=0}^{N_p-1} \boldsymbol{\lambda}^{k,T}\mathbf{r}_{\max}^k \tag{7.6}$$

The Lagrange function can then be decomposed into the individual Lagrange functions,

$$\mathcal{L}(\mathbf{x}^{0:N_p}, \mathbf{u}^{0:N_p-1}, \boldsymbol{\lambda}^{0:N_p-1}) = \sum_{i\in\mathcal{I}} \mathcal{L}_i(\mathbf{x}_i^{0:N_p}, \mathbf{u}_i^{0:N_p-1}, \boldsymbol{\lambda}^{0:N_p-1}) - \sum_{k=0}^{N_p-1} \boldsymbol{\lambda}^{k,T}\mathbf{r}_{\max}^k \tag{7.7}$$

Thus the dual function

$$d(\boldsymbol{\lambda}^{0:N_p-1}) \coloneqq \min_{\mathbf{x}^{0:N_p}, \mathbf{u}^{0:N_p-1}} \sum_{i\in\mathcal{I}} \mathcal{L}_i(\mathbf{x}_i^{0:N_p}, \mathbf{u}_i^{0:N_p-1}, \boldsymbol{\lambda}^{0:N_p-1}) - \sum_{k=0}^{N_p-1} \boldsymbol{\lambda}^{k,T}\mathbf{r}_{\max}^k, \tag{7.8a}$$

$$\text{s.t. } \mathbf{x}_i^{k+1} = \mathbf{A}_i\mathbf{x}_i^k + \mathbf{B}_i\mathbf{u}_i^k,\ \forall i \in \mathcal{I}, k = 0, \dots, N_p - 1, \tag{7.8b}$$

$$\mathbf{x}_i^0 = \tilde{\mathbf{x}}(t_0),\ \forall i \in \mathcal{I}, \tag{7.8c}$$

$$\mathbf{x}_i^k \in \mathcal{X}_i \subset \mathbb{R}^{n_{\mathbf{x}_i}},\ \forall i \in \mathcal{I}, k = 0, \dots, N_p, \tag{7.8d}$$

$$\mathbf{u}_i^k \in \mathcal{U}_i \subset \mathbb{R}^{n_{\mathbf{u}_i}},\ \forall i \in \mathcal{I}, k = 0, \dots, N_p - 1 \tag{7.8e}$$

$$\tag{7.8f}$$

can be evaluated by solving the individual DMPC problems

$$\min_{\mathbf{x}_i^{0:N_p}, \mathbf{u}_i^{0:N_p-1}} \mathcal{L}_i(\mathbf{x}_i^{0:N_p}, \mathbf{u}_i^{0:N_p-1}, \boldsymbol{\lambda}^{0:N_p-1}), \tag{7.9a}$$

$$\text{s.t. } \mathbf{x}_i^{k+1} = \mathbf{A}_i\mathbf{x}_i^k + \mathbf{B}_i\mathbf{u}_i^k,\ k = 0, \dots, N_p - 1, \tag{7.9b}$$

$$\mathbf{x}_i^0 = \tilde{\mathbf{x}}(t_0), \tag{7.9c}$$

$$\mathbf{x}_i^k \in \mathcal{X}_i \subset \mathbb{R}^{n_{\mathbf{x}_i}},\ k = 0, \dots, N_p, \tag{7.9d}$$

$$\mathbf{u}_i^k \in \mathcal{U}_i \subset \mathbb{R}^{n_{\mathbf{u}_i}},\ k = 0, \dots, N_p - 1 \tag{7.9e}$$

for each subsystem i in a distributed manner. A subgradient of the dual function (7.8) in iteration t can be computed by evaluating the constraints on the shared limited resources (7.5f), i.e.,

$$\mathbf{g}(\boldsymbol{\lambda}^{0:N_p-1,(t)}) \coloneqq \begin{pmatrix} \sum_{i\in\mathcal{I}} \mathbf{R}_i\mathbf{u}_i^{0,(t+1)} - \mathbf{r}_{\max}^0 \\ \vdots \\ \sum_{i\in\mathcal{I}} \mathbf{R}_i\mathbf{u}_i^{N_p-1,(t+1)} - \mathbf{r}_{\max}^{N_p-1} \end{pmatrix} \in \partial d(\boldsymbol{\lambda}^{0:N_p-1,(t)}). \tag{7.10}$$

The algorithms presented in Chapter 3 and 4 can thus be used to distributedly solve the system-wide constraint-coupled MPC problem (7.5). Note that in the case of ADMM auxiliary variables $\mathbf{z}_i^k$ have to be defined for each subsystem at each time point.

7.2.4 Dual decomposition of dynamically coupled DMPC problems

For the sake of completeness, this section briefly demonstrates how dual decomposition can be applied to distributedly solve linear DMPC problems with coupled dynamics (7.4). Also for the sake of brevity, it is assumed that the subsystems are only coupled through their states, i.e., $\mathbf{B}_{ij} = \mathbf{0}$ for $i \neq j$. First the subset $\mathcal{N}_i \subset \mathcal{I}$ is defined which contains all neighbors of subsystem i, i.e., $\mathbf{A}_{ij} \neq \mathbf{0},\ \forall j \in \mathcal{N}_i$. Each subsystem i can now be augmented by additional decision variables $\mathbf{v}_{ij}$ which represent a local copy of the states of a neighboring subsystem j. With this, the system-wide problem (7.4) can be reformulated as

$$\min_{\mathbf{x}^{0:N_p},\mathbf{u}^{0:N_p-1},\mathbf{v}^{0:N_p}} \sum_{i\in\mathcal{I}} \left[J_i^f(\mathbf{x}_i^{N_p}) + \sum_{k=0}^{N_p-1} J_i(\mathbf{x}_i^k, \mathbf{u}_i^k) \right], \tag{7.11a}$$

$$\text{s.t. } \mathbf{x}_i^{k+1} = \mathbf{A}_{ii}\mathbf{x}_i^k + \sum_{j\in\mathcal{N}_i} \mathbf{A}_{ij}\mathbf{v}_{ij}^k + \mathbf{B}_i\mathbf{u}_i^k,\ \forall i \in \mathcal{I}, k = 0,\dots,N_p-1, \tag{7.11b}$$

$$\mathbf{x}_i^0 = \tilde{\mathbf{x}}(t_0),\ \forall i \in \mathcal{I}, \tag{7.11c}$$

$$\mathbf{x}_i^k \in \mathcal{X}_i \subset \mathbb{R}^{n_{\mathbf{x}_i}},\ \forall i \in \mathcal{I}, k = 0,\dots,N_p, \tag{7.11d}$$

$$\mathbf{u}_i^k \in \mathcal{U}_i \subset \mathbb{R}^{n_{\mathbf{u}_i}},\ \forall i \in \mathcal{I}, k = 0,\dots,N_p-1, \tag{7.11e}$$

$$\mathbf{v}_{ij}^k - \mathbf{x}_j^k = \mathbf{0},\ \forall i \in \mathcal{I}, j \in \mathcal{N}_i, k = 0,\dots,N_p. \tag{7.11f}$$

The subsystems in problem (7.11) are now coupled through the constraints (7.11f) for which dual variables $\boldsymbol{\lambda}_{ij}^k,\ i \neq j$ can be introduced. The resulting individual Lagrange function for a subsystem i can then be defined as

$$\mathcal{L}_i(\mathbf{x}_i^{k:N_p}, \mathbf{u}_i^{0:N_p-1}, \mathbf{v}_{ij}^{0:N_p}, \boldsymbol{\lambda}_{ij}^{0:N_p}) = J_i^f(\mathbf{x}_i^{N_p}) + \sum_{k=0}^{N_p-1} J_i(\mathbf{x}_i^k, \mathbf{u}_i^k) + \sum_{k=0}^{N_p} \sum_{j\in\mathcal{N}_i} [\boldsymbol{\lambda}_{ij}^{k,T}\mathbf{v}_{ij}^k - \boldsymbol{\lambda}_{ji}^{k,T}\mathbf{x}_i^k]. \tag{7.12}$$

The system-wide MPC problem (7.11) can then be solved in a distributed manner by optimizing the individual Lagrange functions in parallel and coordinating the solution via the dual variables.

7.3 Numerical analysis for DMPC problems

In this section, several linear DMPC benchmark problems are solved using the subgradient method, BTM, ADMM, and QNDA. While the QAC and QADA algorithms could theoretically be used to solve the DMPC problems, the number of involved dual variables is prohibitive for a regression-based approach. For a DMPC problem (7.5) $\boldsymbol{\lambda} \in \mathbb{R}^{n_\mathbf{r}\cdot(N_p-1)}$ holds, hence the number of required data points for a regression rises quadratically both with the number of resources and with the prediction horizon. For instance, a problem of moderate size with $n_\mathbf{r} = 3$ resources and a prediction horizon of $N_p = 15$ would require $0.5 \cdot (n_\mathbf{r} \cdot (N_p - 1) + 1)(n_\mathbf{r} \cdot (N_p - 1) + 2) = 946$ data points for a quadratic approximation.

The generated benchmark problems have the structure of problem (7.5) with a reference tracking objective (7.2) and a terminal cost (7.3). The objective function matrices were generated randomly as symmetric positive definite matrices,

$$\mathbf{H}_{\mathbf{x}_i} = \mathbf{H}^f_{\mathbf{x}_i} = \mathbf{N}^T_{\mathbf{x}_i}\mathbf{N}_{\mathbf{x}_i} \in \mathbb{R}^{n_{\mathbf{x}_i}\times n_{\mathbf{x}_i}}, \tag{7.13}$$

$$\mathbf{H}_{\mathbf{u}_i} = \mathbf{N}^T_{\mathbf{u}_i}\mathbf{N}_{\mathbf{u}_i} \in \mathbb{R}^{n_{\mathbf{u}_i}\times n_{\mathbf{u}_i}}, \tag{7.14}$$

where the elements of $\mathbf{N}_{\mathbf{x}_i}$ and $\mathbf{N}_{\mathbf{u}_i}$ were drawn from a normal distribution $[\mathbf{N}_{\mathbf{x}_i}]_{l,j}, [\mathbf{N}_{\mathbf{u}_i}]_{l,j} \in \mathcal{N}(\mu = 0, \sigma = 1)$.

The system matrices $\mathbf{A}_i$ were generated such that the resulting subsystems are stable. A discrete-time system is stable if all eigenvalues of the matrix $\mathbf{A}_i$ lie within the unit sphere [Lun14]. The matrices were thus computed as

$$\mathbf{A}_i = \mathbf{S}_i\mathbf{D}_i\mathbf{S}_i^{-1}, \tag{7.15}$$

where $\mathbf{D}_i$ is a diagonal matrix containing the eigenvalues of the matrix $\mathbf{A}_i$ drawn from a continuous uniform distribution $[\mathbf{D}_i]_{l,l} \in \mathcal{U}_c(-1, 1)$ while the elements of the matrix $\mathbf{S}_i$ were drawn from the continuous uniform distribution $\mathcal{U}_c(-3, 3)$. The elements of the input matrix $\mathbf{B}_i$ were drawn from the continuous uniform distribution $\mathcal{U}_c(-2, 2)$.

The individual constraints were generated as intersections of ellipsoids around the origin,

$$\mathcal{X}_i = \{\mathbf{x}_i \in \mathbb{R}^{n_{\mathbf{x}_i}} |\ \mathbf{x}_i^T\mathbf{G}_{\mathbf{x}_i,l}\mathbf{x}_i \leq p^2_{\mathbf{x}_i,l},\ l = 1, \ldots, n_c\}, \tag{7.16}$$

$$\mathcal{U}_i = \{\mathbf{u}_i \in \mathbb{R}^{n_{\mathbf{u}_i}} |\ \mathbf{u}_i^T\mathbf{G}_{\mathbf{u}_i,l}\mathbf{u}_i \leq p^2_{\mathbf{u}_i,l},\ l = 1, \ldots, n_c\}. \tag{7.17}$$

The matrices $\mathbf{G}_{\mathbf{x}_i,l}$ and $\mathbf{G}_{\mathbf{u}_i,l}$ were generated in the same way as the objective matrices while the right-hand sided $p_{\mathbf{x}_i,l}$ and $p_{\mathbf{u}_i,l}$ were drawn from the continuous uniform distribution $\mathcal{U}_c(1, 5)$.

Ideally the reference trajectory $\mathbf{x}_i^{\text{ref},0:N_p}$ should be feasible. Therefore, the elements of the reference trajectory were drawn from the continuous uniform distribution $\mathcal{U}_c(-2, 2)$ at each time point until a feasible point was found, i.e., until $\mathbf{x}_i^{\text{ref},k} \in \mathcal{X}_i$. The initial state $\tilde{\mathbf{x}}(t_0)$ was generated in the same way.

The elements of the resource matrix $\mathbf{R}_i$ were first drawn from the continuous distribution $\mathcal{U}_c(1, 2)$. Afterward, they were altered such that their sign was flipped or they were set to zero through the uniform discrete distribution $\mathcal{U}_d[-1, 0, 1]$,

$$\mathbf{R}_i = \mathbf{D}_i \circ \mathbf{C}_i \in \mathbb{R}^{n_\mathbf{r}\times n_{\mathbf{u}_i}},\ [\mathbf{D}_i]_{l,j} \in \mathcal{U}_c(1, 2), [\mathbf{C}_i]_{l,j} \in \mathcal{U}_d[-1, 0, 1]. \tag{7.18}$$

The maximum resource utilization should be generated such that the decentralized solution, i.e., for $\boldsymbol{\lambda} = \mathbf{0}$, is infeasible, while the system-wide problem is feasible. To this end, the optimal input trajectory for the system-wide problem without the resource constraints was first computed,

$$\mathbf{u}^{0:N_p-1,*} = \arg\min_{\mathbf{x}^{0:N_p},\mathbf{u}^{0:N_p-1}} \sum_{i\in\mathcal{I}} \left[J_i^f(\mathbf{x}_i^{N_p}) + \sum_{k=0}^{N_p-1} J_i(\mathbf{x}_i^k, \mathbf{u}_i^k) \right], \tag{7.19a}$$

$$\text{s.t. } \mathbf{x}_i^{k+1} = \mathbf{A}_i\mathbf{x}_i^k + \mathbf{B}_i\mathbf{u}_i^k,\ \forall i \in \mathcal{I}, k = 0,\dots,N_p-1, \tag{7.19b}$$

$$\mathbf{x}_i^0 = \tilde{\mathbf{x}}(t_0),\ \forall i \in \mathcal{I}, \tag{7.19c}$$

$$\mathbf{x}_i^k \in \mathcal{X}_i \subset \mathbb{R}^{n_{\mathbf{x}_i}},\ \forall i \in \mathcal{I}, k = 0,\dots,N_p, \tag{7.19d}$$

$$\mathbf{u}_i^k \in \mathcal{U}_i \subset \mathbb{R}^{n_{\mathbf{u}_i}},\ \forall i \in \mathcal{I}, k = 0,\dots,N_p-1. \tag{7.19e}$$

Using the found optimal solution the optimal unconstrained resource utilization can be computed,

$$\mathbf{r}^{k,*} \coloneqq \sum_{i\in\mathcal{I}} \mathbf{R}_i\mathbf{u}_i^{k,*},\ k = 0,\dots,N_p-1. \tag{7.20}$$

The maximum resource utilization was then computed by tightening the optimal unconstrained resource utilization,

$$[\mathbf{r}_{\max}^k]_l = [\mathbf{r}^{k,*}]_l - \beta_l^k|[\mathbf{r}^{k,*}]_l|, k = 0,\dots,N_p-1,\ l = 1,\dots,n_{\mathbf{r}},\ \beta_l^k \in \mathcal{U}_c(0,0.2). \tag{7.21}$$

After the maximum resource utilization was generated, the feasibility of the decentralized and the system-wide problem was verified.

The number and size of the subproblems were varied as follows:

Number of subproblems: $N_s \in \{5, 10, 20, 50\}$,
Number of states: $n_{\mathbf{x}} \in \{2, 3, 4, 5\}$,
Number of inputs: $n_{\mathbf{u}} \in \{2, 3, 4, 5\}, n_{\mathbf{x}} \geq n_{\mathbf{u}}$
Number of resources: $n_{\mathbf{r}} \in \{2, 3, 4, 5\}, n_{\mathbf{u}} \geq n_{\mathbf{r}}$
Prediction horizon: $N_p \in \{10, 15, 20\}$.

Each subsystem contains the same number of states and inputs, i.e., $n_{\mathbf{x}_i} = n_{\mathbf{x}},\ n_{\mathbf{u}_i} = n_{\mathbf{u}},\ \forall i \in \mathcal{I}$. The number of state and input constraints was set equal to the number of states, i.e., $n_c = n_{\mathbf{x}}$. Ten benchmark problems were generated for each combination resulting in a total of 2400 DMPC problems. The subproblems were solved in each iteration with the commercial solver Gurobi [Gur23].

7.3.1 Modifications of distributed optimization algorithms

For the DMPC benchmark problems, some modifications were made to the ADMM and QNDA algorithms. These are presented in the following.

ADMM

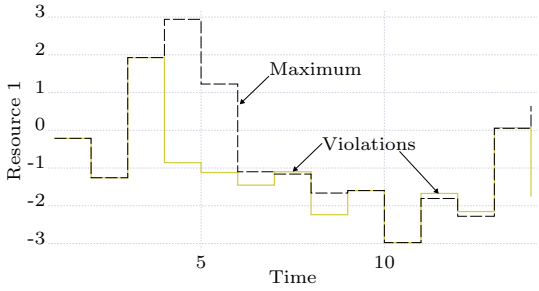
The ADMM algorithm as presented in Sec. 3.3 tends to push the subsystems towards primal feasibility due to the added regularization term. For inequality-constrained problems, this can lead to a situation where the system-wide constraints are satisfied while the regularization term does not vanish. However, the regularization term should vanish if the optimal dual variables are found, i.e., the dual variables found using ADMM should result in a primal feasible solution for the unaugmented Lagrange function. Maxeiner [Max21] proposed to replace the update of the auxiliary variables (3.23b) by an optimization problem,

$$\mathbf{z}^{(t+1)} = \underset{\mathbf{z}}{\operatorname{argmin}} \sum_{k=0}^{N_p-1} \left\| \sum_{i\in\mathcal{I}} \mathbf{R}_i \mathbf{u}_i^{k,(t)} - \mathbf{z}_i^k \right\|_2^2, \tag{7.22a}$$

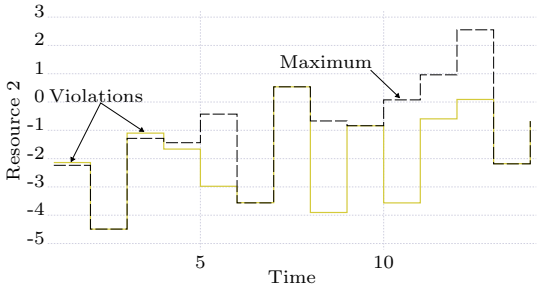
$$\text{s.t.} \sum_{i\in\mathcal{I}} \mathbf{z}_i^k \leq \mathbf{r}_{\max}^k, \; k = 0, \ldots, N_p - 1. \tag{7.22b}$$

In the following the auxiliary variables are updated using (7.22).

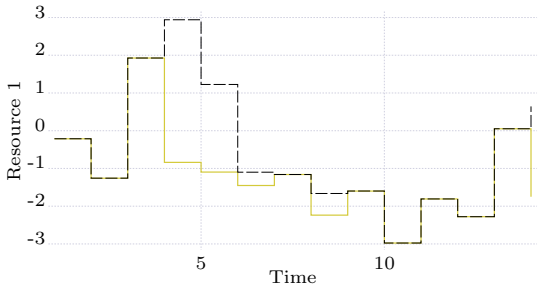
Fig. 7.3 illustrates the effect of the update of the auxiliary variables. When using the update strategy (3.23b) ADMM converges to a value of the dual variables where optimiz-



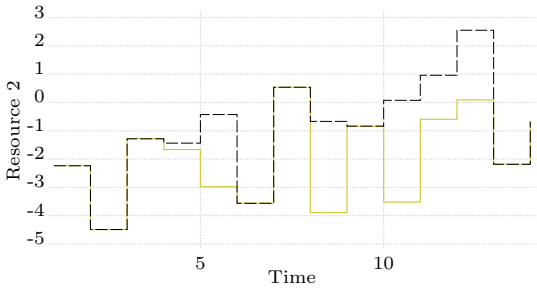
(a) Utilization of resource 1 with update strategy (3.23b).



(b) Utilization of resource 2 with update strategy (3.23b).



(c) Utilization of resource 1 with update strategy (7.22).



(d) Utilization of resource 2 with update strategy (7.22).

Figure 7.3: Comparison of the resource utilization upon convergence of the ADMM algorithm for DMPC benchmark problem with different update strategies of the auxiliary variables. The plots show the aggregated resource utilization for a DMPC problem with $N_s = 20, n_{\mathbf{x}} = 3, n_{\mathbf{u}} = 2, n_{\mathbf{u}} = 2, N_p = 15$. The constraints on the shared resources are violated when using the update strategy (3.23b).

ing the augmented Lagrange function results in a feasible resource utilization. However, when the same dual variables are used to optimize the unaugmented Lagrange function, i.e., without the regularization term, the constraints on the shared limited resources are violated. Thus, ADMM does not converge to the optimal dual variables. This case is shown for two resources in Fig. 7.3a and 7.3b. When the auxiliary variables are updated according to (7.22) ADMM converges to a dual optimal solution, i.e., a feasible solution for both the augmented and unaugmented Lagrange function. This is depicted in Fig. 7.3c and 7.3d.

QNDA

Preliminary numerical tests showed that the QNDA algorithm as presented in Sec. 4.2 tends to converge quickly to the vicinity of the optimum for the DMPC problems. However, it tends to oscillate near the optimum resulting in poor overall performance. To avoid this issue the update strategy of the dual variables is switched to a constrained line search within a certain distance to the optimum,

$$\mathbf{s}^{(t)} = \underset{\mathbf{s},\hat{\alpha}}{\operatorname{argmax}} \ \hat{\alpha}, \tag{7.23a}$$

$$\text{s.t.} \quad \boldsymbol{\lambda}^{(t)} + \mathbf{s} \in \mathcal{BC}_{d_B}^{(t)}, \tag{7.23b}$$

$$\mathbf{s} = \hat{\alpha} \nabla d_B^{(t)}(\boldsymbol{\lambda}^{(t)}), \tag{7.23c}$$

$$\hat{\alpha} \leq \alpha^{(t)}, \tag{7.23d}$$

$$\boldsymbol{\lambda}^{(t+1)} = [\boldsymbol{\lambda}^{(t)} + \mathbf{s}^{(t)}]^{+}. \tag{7.23e}$$

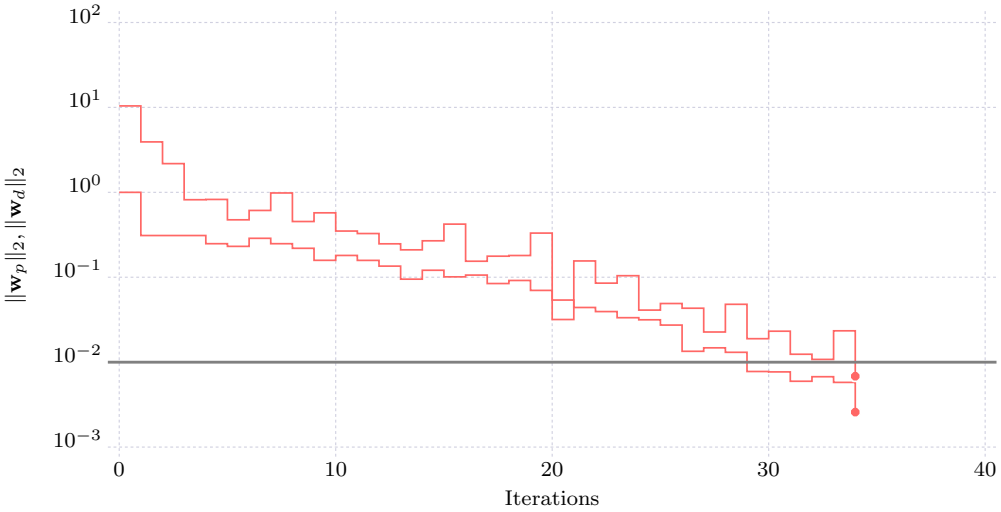
The update (7.23) essentially describes an update step in the direction of the gradient of the approximated dual function. Constraint (7.23b) ensures that the bundle cuts are satisfied while constraint (7.23c) gives the search direction. The step size is limited by constraint (7.23d), which replaces the trust region constraint. The dual variables are then updated in the computed search direction and projected onto the positive orthant due to the nonnegativity constraints on the dual variables. Note that in the case of the QNDA algorithm, the gradient of the approximated dual function is simply the subgradient of the actual dual function, i.e.,

$$\nabla d_B^{(t)}(\boldsymbol{\lambda}^{(t)}) = \mathbf{g}(\boldsymbol{\lambda}^{(t)}) \in \partial d(\boldsymbol{\lambda}^{(t)}). \tag{7.24}$$

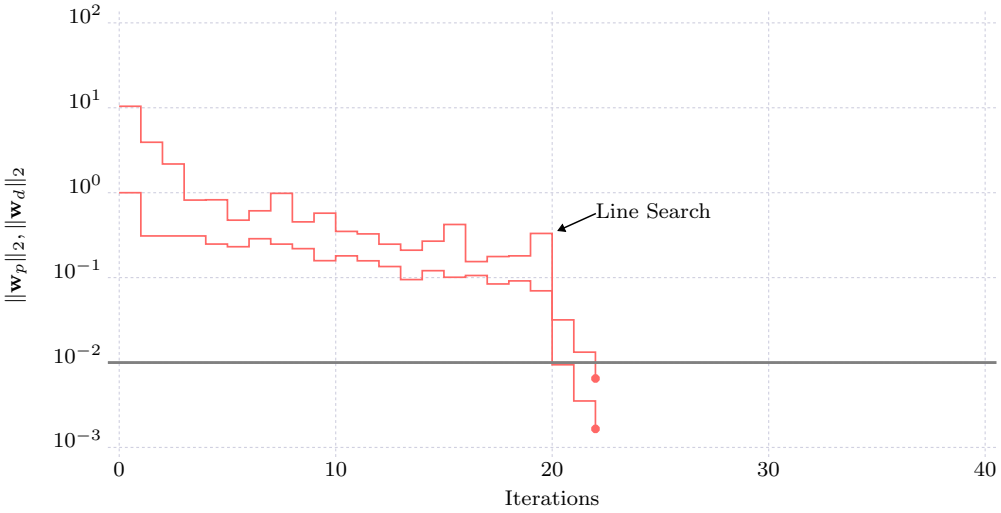
The update steps (7.23) are used if

$$\|\mathbf{w}_p(\boldsymbol{\lambda}^{(t)})\|_2 \leq \epsilon_l \cdot \|\mathbf{w}_p(\boldsymbol{\lambda}^{(0)})\|_2. \tag{7.25}$$

The effect of the line search strategy is illustrated in Fig. 7.4. The evolution of the primal and dual residuals without a line search is shown in Fig. 7.4a. In comparison, the line search strategy (7.23) is employed in Fig. 7.4b. This results in faster convergence in the vicinity of the optimum.



(a) Evolution of the primal and dual residuals without the line search strategy.



(b) Evolution of the primal and dual residuals with the line search strategy.

Figure 7.4: Comparison of the convergence of the QNDA algorithm for a DMPC benchmark problem with and without the line search update (7.23). The plots show the evolutions of the primal and dual residuals for a DMPC problem with $N_s = 20, n_{\mathbf{x}} = 3, n_{\mathbf{u}} = 2, n_{\mathbf{u}} = 2, N_p = 15$. The line search is used once the algorithm reaches the vicinity of the optimum.

7.3.2 Parameter settings for DMPC problems

The parameters for the DMPC problems were set by trial and error to achieve a good compromise between robustness and rate of convergence. The initial step size parameter (SG)/ trust region parameter (BTM, QNDA) was set to $\alpha^{(0)} = 1$ and updated according to (6.5). The ADMM algorithm exhibited divergence when tuned too aggressively. Therefore the initial regularization parameter was set $\rho^{(0)} = 10^{-3}$. The tuning parameters were set to $\tau_{\text{incr}} = 1.25$, $\tau_{\text{decr}} = 1.1$ and $\mu = 10$. The bundle cuts threshold was set to $\epsilon_b = 0.6$ (cf.

Table 7.1: Parameter settings of the distributed optimization algorithms for the DMPC benchmark problems.

	Value	Description	Algorithms
$\boldsymbol{\lambda}^{(0)}$	$\mathbf{0}$	initial dual variables	All
$\alpha^{(0)}$	1	initial step size/trust region parameter	SG, BTM, QNDA
$t_{\max}$	500	maximum number of iterations	All
ϵ_p	10^{-2}	primal residual convergence tolerance	All
ϵ_d	10^{-2}	dual residual convergence tolerance	All
ϵ_b	0.6	bundle cuts threshold	QNDA
ϵ_l	10^{-2}	line search threshold	QNDA
$\rho^{(0)}$	10^{-3}	initial regularization parameter	ADMM
τ_{incr}	1.25	see (3.29)	ADMM
τ_{decr}	1.1	see (3.29)	ADMM
μ	10	see (3.29)	ADMM
$\mathbf{z}^{(0)}$	$\mathbf{0}$	initial auxiliary variables	ADMM
τ	150	allowed age of data points	BTM, QNDA
$\mathbf{B}^{(0)}$	$-\mathbf{I}$	initial approximated Hessian	QNDA

Table 7.2: Summary of the results for the distributed optimization of the DMPC benchmark problems (mean values of the converged instances only), $\bar{t}$: mean number of iterations until convergence, $\overline{T_{\text{comp}}}$: mean computation time of converged runs (in s), $\overline{\|\mathbf{w}_p\|_2}$: mean primal residual of converged runs ($\times 10^{-3}$), $\%_c$: percentage of converged runs within $t_{\max}$ iterations.

Algorithm	$\bar{t}$	$\overline{T_{\text{comp}}}$	$\overline{\|\mathbf{w}_p\|_2}$	$\%_c$
SG	42.83	**35.24**	8.18	80.96
BTM	273.83	228.17	7.09	57.58
ADMM	62.24	51.92	7.65	**99.21**
QNDA	**42.6**	38.31	**7.07**	98.21

eq. (6.6)) and the threshold for the line search (7.23) was set to $\epsilon_l = 10^{-2}$ (cf. eq. (7.25)). The age parameter for BTM and QNDA was set to $\tau = 150$. All algorithms were initialized with $\boldsymbol{\lambda}^{(0)} = \mathbf{0}$ and $\mathbf{z}_i^{(0)} = \mathbf{0}$, $\forall i \in \mathcal{I}$ (for ADMM). The approximated Hessian for the QNDA algorithm was initialized with the negative identity matrix. All algorithms were terminated after $t_{\max} = 500$ iterations or when the primal and dual residuals reached the threshold $\epsilon_p = \epsilon_d = 10^{-2}$. The communication time was set to $T_{\text{comm}} = 800\ ms$ (cf. eq. (6.1)).

7.3.3 Results for DMPC problems

The results of the distributed optimization of the DMPC benchmark problems are depicted in Fig. 7.5 and summarized in Tab. 7.2. The results show that the ADMM and QNDA

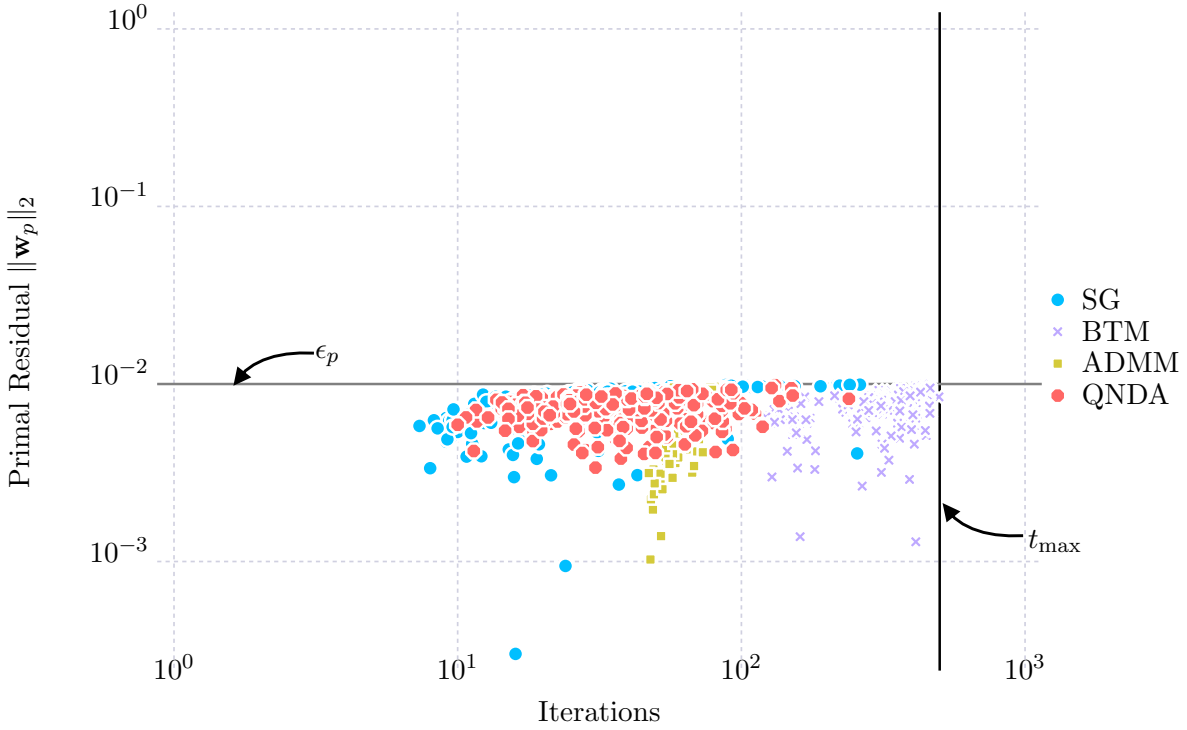


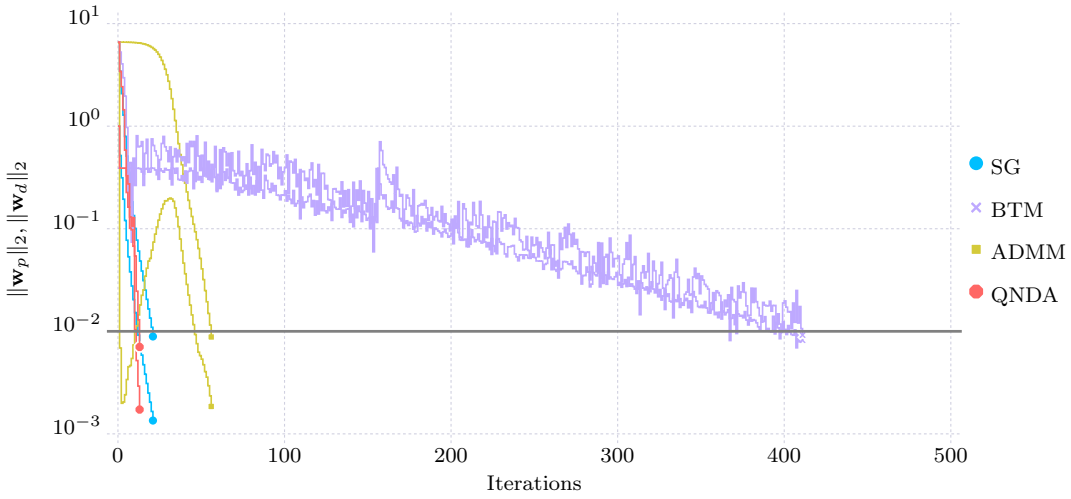
Figure 7.5: Values of the primal residuals upon termination for the distributed MPC problems. Each data point represents the mean values of the converged problem instances for a tuple $(N_s, n_{\mathbf{x}}, n_{\mathbf{u}}, n_{\mathbf{r}}, N_p)$ (cf. Tab. C.4).

algorithms can solve almost all DMPC benchmark problems. While ADMM can solve slightly more problems, the QNDA algorithm outperforms ADMM in terms of required iterations, computation time, and quality of found solutions. The subgradient method exhibits a similar number of required iterations for its converged runs as QNDA with a better computation time due to the less expensive update steps. However, the subgradient method solves far fewer problems than the ADMM and QNDA algorithms. The BTM algorithm exhibits rather poor performance for the DMPC benchmark problems. A more detailed summary of the results is given in Tab. C.4 in the appendix.

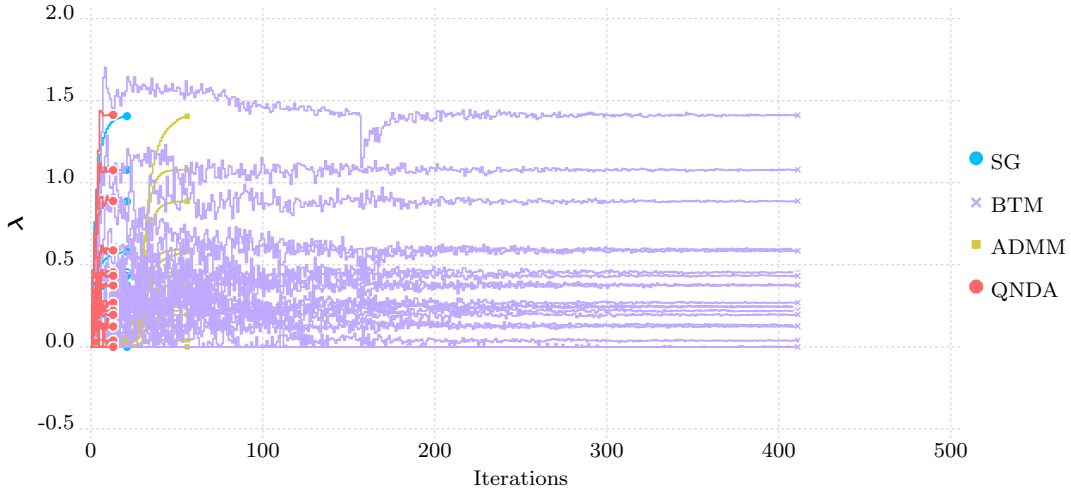
Fig. 7.6 shows the results for the distributed optimization of a benchmark problem with three shared resources and a prediction horizon of 15. i.e., $\boldsymbol{\lambda} \in \mathbb{R}^{42}$. The subgradient method and the QNDA algorithm exhibit the fastest convergence. ADMM exhibits a small oscillation but still converges quickly. The BTM algorithm exhibits significant oscillations and converges very slowly compared to the other algorithms. However, Fig. 7.6b shows that the oscillations mainly take place in the vicinity of the optimal dual variables, indicating that the resource utilization is sensitive to the change of the dual variables.

Fig. 7.7 depicts the utilization of the shared limited resources upon the convergence of the distributed optimization algorithms for the DMPC benchmark problem in Fig. 7.6. All algorithms converge to the same resource utilization, i.e., to the optimal dual variables.

The results in Tab. 7.2 show that the subgradient method exhibits similar performance to QNDA for its converged runs. However, aggressive tuning is necessary to obtain this



(a) Evolution of the primal and dual residuals.

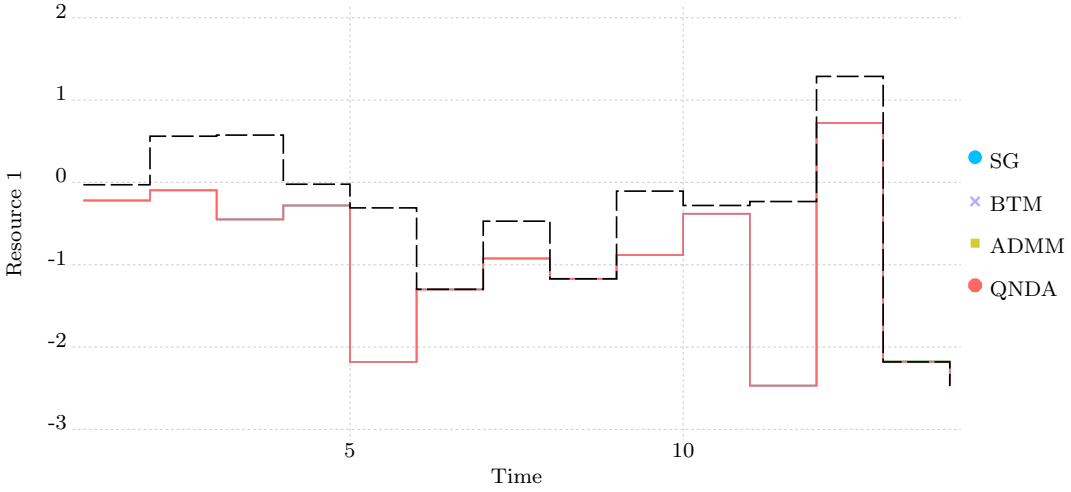


(b) Evolution of the dual variables.

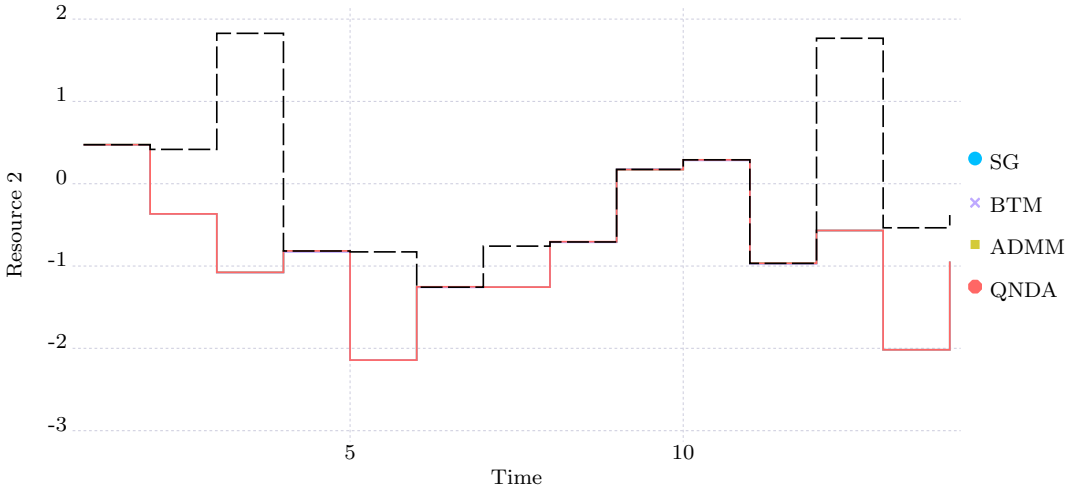
Figure 7.6: Results for a DMPC benchmark problem with $N_s = 10, n_{\mathbf{x}} = 5, n_{\mathbf{u}} = 5, n_{\mathbf{r}} = 3, N_p = 15$.

performance, which comes at the cost of robustness.

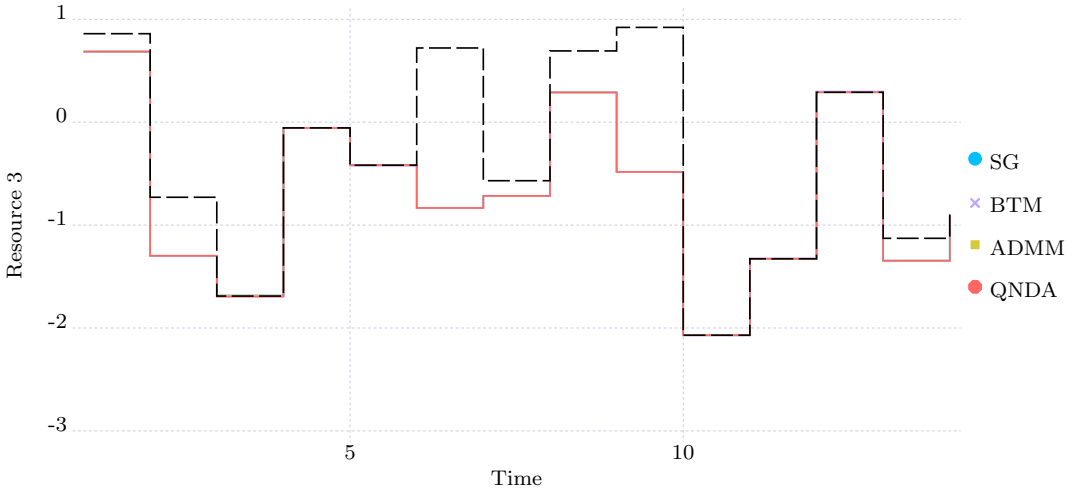
Fig. 7.8 shows the results for a DMPC benchmark problem with two shared resources and a prediction horizon of 10, i.e., $\boldsymbol{\lambda} \in \mathbb{R}^{18}$. The subgradient method is not able to converge as it exhibits extreme oscillations due to the aggressive step size parameter. Compared to that, all other algorithms converge, with the QNDA algorithm being the most efficient. Note that both the QNDA and BTM algorithms employ the same parameter as the subgradient method for their trust region and, in the case of the QNDA algorithm, for the line search updates. As discussed in Chapter 6 the degree of nonsmoothness increases as the number of subproblems decreases. The BTM and QNDA algorithms take this nonsmoothness into account through the stored bundle information. Most notably, the bundle cuts of QNDA prevent the oscillations shown by the subgradient method when the line search strategy is employed.



(a) Utilization of resource 1.

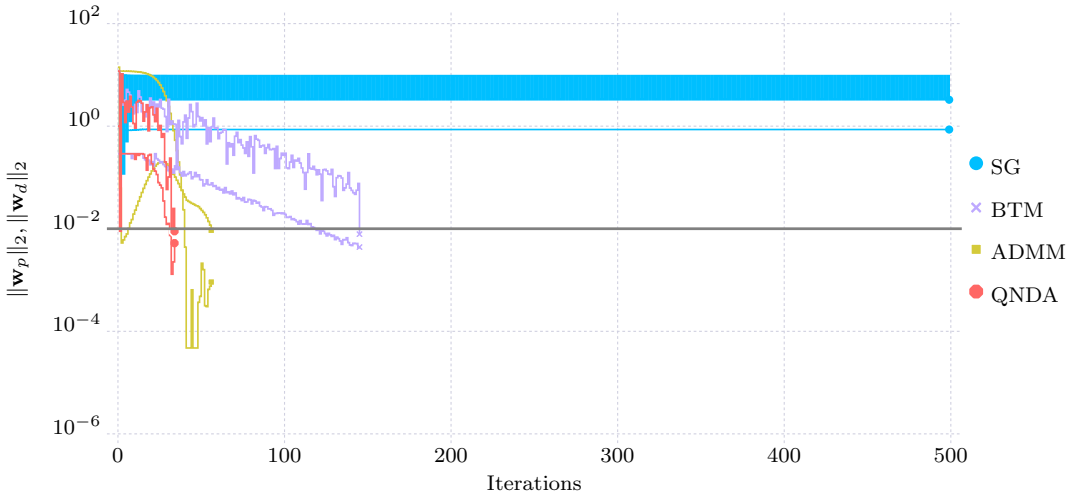


(b) Utilization of resource 2.

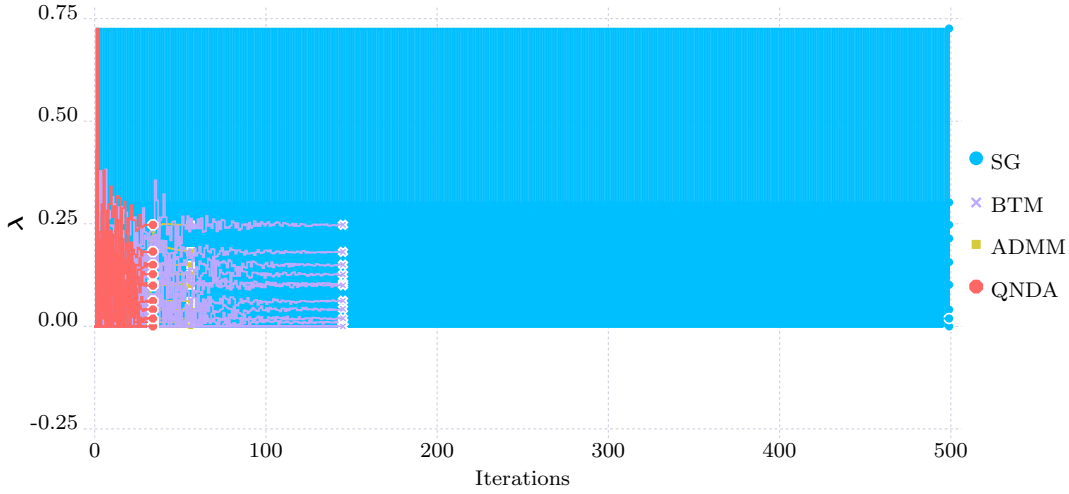


(c) Utilization of resource 3.

Figure 7.7: Resource utilization upon termination for a DMPC benchmark problem with $N_s = 10, n_{\mathbf{x}} = 5, n_{\mathbf{u}} = 5, n_{\mathbf{r}} = 3, N_p = 15$.



(a) Evolution of the primal and dual residuals.



(b) Evolution of the dual variables.

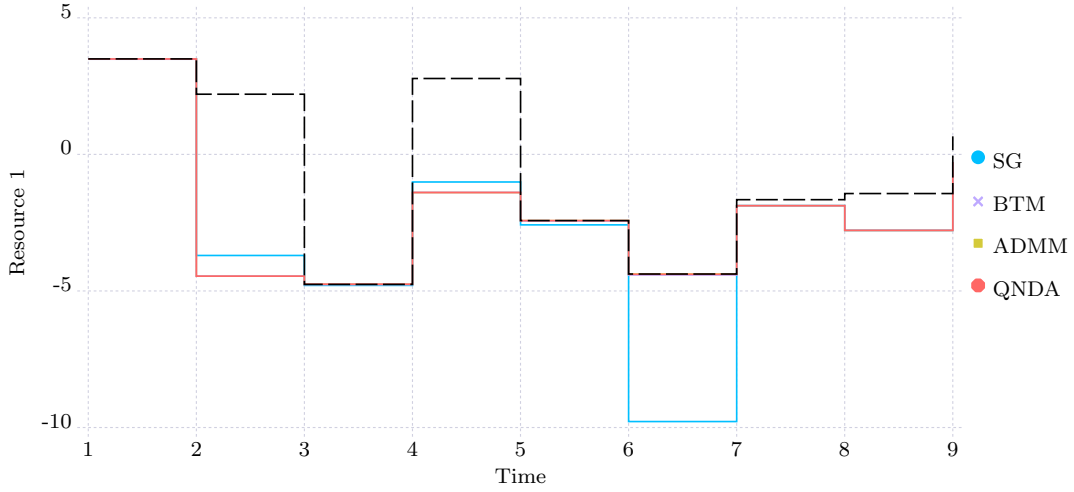
Figure 7.8: Results for a DMPC benchmark problem with $N_s = 50, n_{\mathbf{x}} = 4, n_{\mathbf{u}} = 4, n_{\mathbf{r}} = 2, N_p = 10$.

Fig. 7.9 shows the utilization of the shared limited resources for the DMPC benchmark problem in Fig. 7.8. The BTM, ADMM, and QNDA algorithms converge to the optimal resource utilization. Fig. 7.9b shows that the subgradient method terminates with an infeasible resource utilization for resource 2.

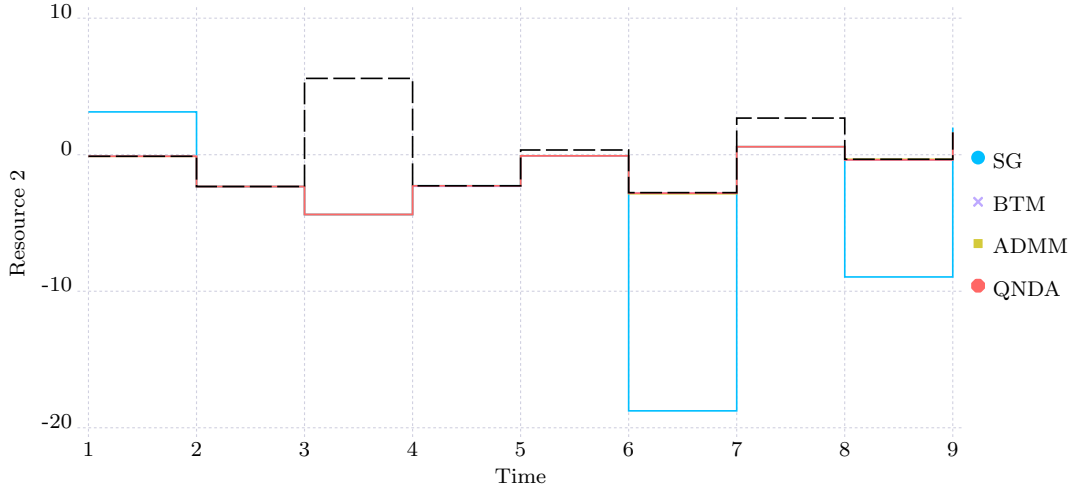
7.4 Conclusion

This chapter demonstrated how dual decomposition-based distributed optimization can be applied to solve distributed MPC problems that are coupled through shared limited resources. A large number of benchmark problems validated the efficiency of the proposed QNDA algorithm. While the ADMM algorithm was able to solve the most benchmark problems and the subgradient method showed similar performance to QNDA for

its converged runs, the QNDA algorithm exhibits the best balance in terms of performance and robustness. Future research could focus on the evaluation of the QNDA algorithm for DMPC problems with coupled dynamics. Furthermore, the application of dual decomposition-based DMPC on systems with nonlinear dynamics and constraints can be investigated. Nonlinear model predictive control (NMPC) problems are usually nonconvex, which poses challenges for dual decomposition, regardless of the used algorithm.



(a) Utilization of resource 1.



(b) Utilization of resource 2.

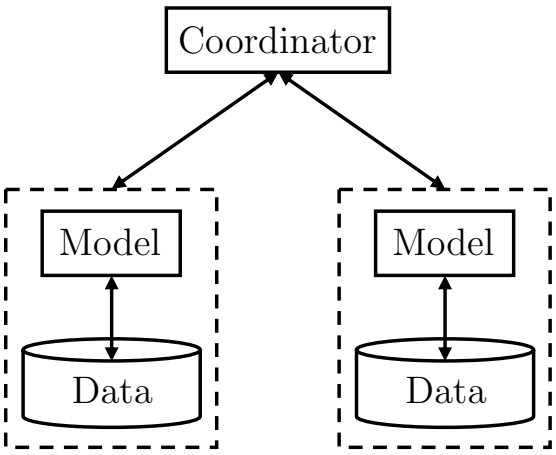
Figure 7.9: Resource utilization upon termination for a DMPC benchmark problem with $N_s = 50, n_{\mathbf{x}} = 4, n_{\mathbf{u}} = 4, n_{\mathbf{r}} = 3, N_p = 10$.

8 Application: Distributed Clustering

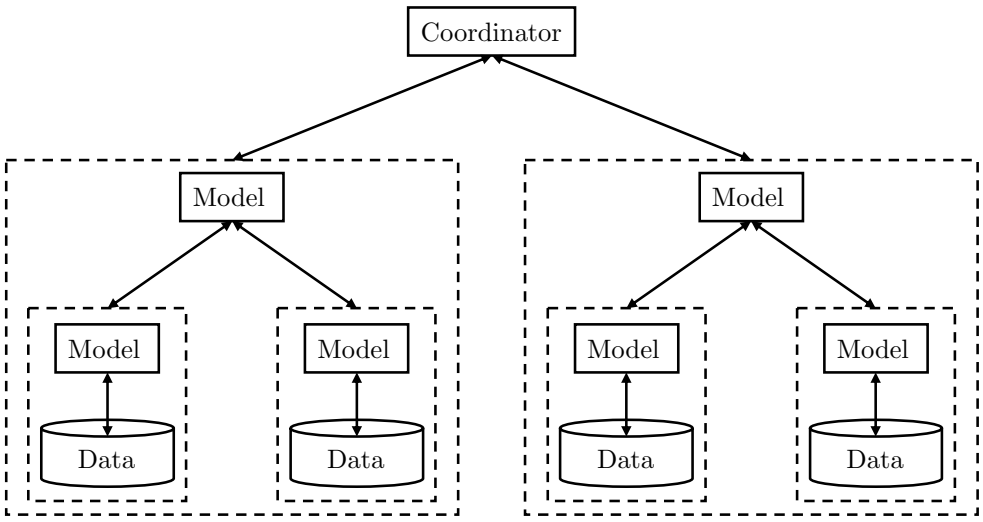
While Chapter 7 dealt with the distributed optimization of continuous convex problems, this chapter presents an application of distributed mixed-integer programming. As discussed in previous chapters, the use of distributed optimization can be motivated either by the resulting preservation of privacy or the increase in computational efficiency. Both aspects play a major role in machine learning. On the one hand, training data might be stored across multiple devices. Training a global model within a network where each node only has access to its confidential data requires the use of distributed algorithms. Even if the data is not confidential, sharing it might be prohibitive due to bandwidth limitations. On the other hand, the ever-increasing amount of available data leads to large-scale machine learning problems. By splitting the training process across multiple nodes its efficiency can be significantly increased. This chapter demonstrates the application of dual decomposition to the distributed training of K-means clustering problems. After an overview of distributed and federated machine learning, the mixed-integer quadratically constrained programming-based formulation of the K-means clustering training problem is presented. The training can be performed in a distributed manner by splitting the data across different nodes and linking these nodes through consensus constraints. Finally, the performance of the subgradient method, the bundle trust method, and the QNDA algorithm are evaluated on a set of benchmark problems. The contents of this chapter have been partially published in [YWR23].

8.1 Distributed and federated machine learning

Training a machine learning model of any kind on a large set of data usually involves the solution of a challenging optimization problem. If the underlying data set becomes too large, it might not be possible to solve the resulting optimization problem in a reasonable amount of time. Distributed optimization methods can aid in rendering the optimization problem tractable through the use of multiple computational resources. Peteiro-Barral and Guijarro-Berdiñas provide an overview of methods for distributed machine learning [PG13]. To train a global model in a distributed manner a consensus has to be established between the involved nodes and their underlying optimization problems. Forero et al. [FCG10, FCG11] and Georgopoulos and Hasler [GH14] demonstrate the distributed train-



(a) Simple architecture for federated learning.



(b) Federated learning of node clusters.

Figure 8.1: Examples of federated learning architectures.

ing of machine learning models using consensus-based distributed optimization. Tsianos et al. discuss practical issues with a consensus-based approach that arise from the difference between synchronous and asynchronous communication [TLR12]. Nedić provides an overview of distributed gradient methods for convex training problems [Ned20] while Verbraeken et al. give a general survey of distributed machine learning [VWK+20].

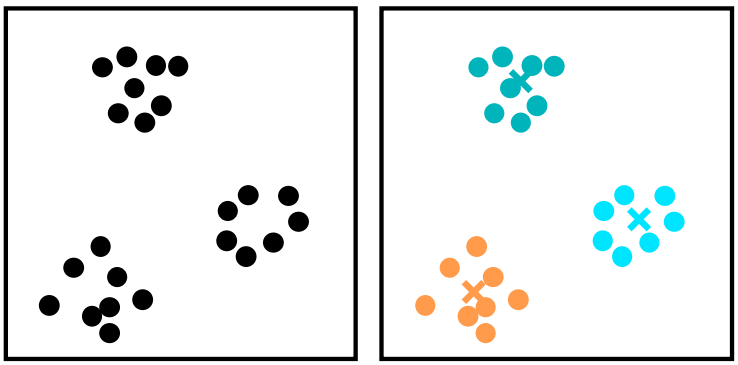
While computational performance remains an issue for many machine learning problems, the increase in computing power and the efficiency of optimization algorithms can render many challenging problems tractable. However, the inability to share data due to confidentiality reasons still necessitates the use of distributed algorithms. Fig. 8.1a shows a setting in which training data is stored across two different nodes. Each node can use its local data to train an individual machine learning model. By including a coordination layer the two training processes can be guided in a way that a global model is trained, without the need to share confidential data. If the underlying optimization problems are still hard to solve, the training process can be further divided into subproblems. Fig. 8.1b

depicts the situation in which models of different node clusters are trained in a distributed manner which in turn are again coordinated to obtain a global model. Distributed training of a global model without sharing individual training data is often referred to as federated optimization or federated learning [KMR+16]. Most algorithms for federated learning involve an averaging step of the model parameters of the individual nodes [MMR+17]. Federated learning methods have been applied in the context of manufacturing [HLR22], healthcare [AAK+22], mobile devices [LLH+20], and smart city sensing [JKO+20]. Li et al. [LFT+20] and Liu et al. [LHZ+22] provide surveys on federated learning while Chamikara et al. examine the privacy aspects related to external attacks [CBK+21]. Applying federated learning to heterogeneous data sets can lead to the deterioration of the model quality of individual nodes in regards to their training data, which might hinder their willingness to participate in such a setting. This issue is addressed through personalized federated learning [KKP20, TYC+22].

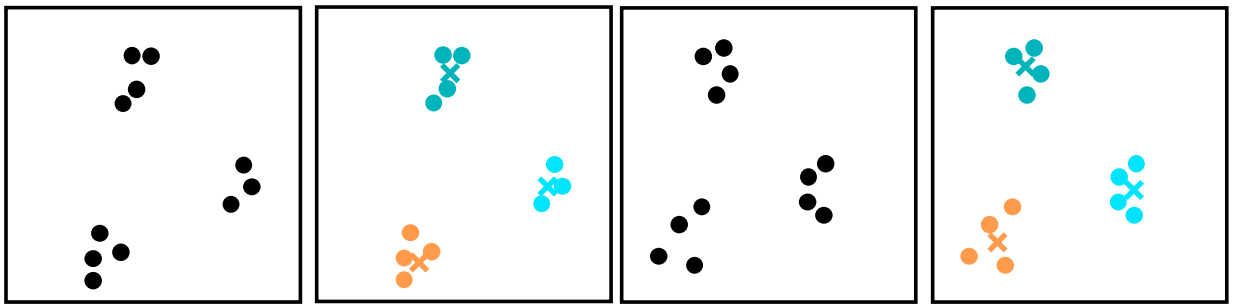
8.2 K-means clustering

K-means clustering describes an unsupervised machine learning problem in which a set of observations/data is divided into K disjoint clusters according to a similarity measure [GGN21]. Clustering problems can be found in many practical applications such as image segmentation [DMC15], customer market segmentation [KBS+18], or the identification of similar operating points in a production plant [RSB+19]. While K-means clustering is a well-studied problem, federated clustering, i.e., clustering of data across multiple nodes, has not been studied extensively in the literature yet. Dennis et al. present a one-shot federated clustering algorithm with heterogeneous data, where the clustering problem of each node is solved by Lloyd's heuristic algorithm [DLS21, Llo82]. Kumar et al. apply federated averaging to pre-trained models on separate devices and present an update strategy when new data points are added [KKN20]. Li et al. address the security aspect of federated clustering by encoding the data of each node and applying Lloyd's algorithm to the encoded data [LHB+22]. Stallmann and Wilbik extend fuzzy c-means clustering, i.e., a clustering problem where each data point can be assigned to multiple clusters, to a federated setting [SW22]. A similar approach to federated c-means clustering was previously presented by Pedrycz [Ped21]. Wang et al. use model averaging and gradient sharing for federated clustering of data in a smart grid [WJG+22].

A common feature of most federated clustering approaches described in the literature is the use of a heuristic algorithm to solve the individual clustering problems. In this thesis, mixed-integer programming is used to solve the individual clustering problems, and a dual decomposition-based distributed optimization approach is employed to coordinate the solutions of the different nodes. While an averaging step is still performed to obtain feasible primal solutions, the use of duality enables the computation of valid lower bounds



(a) Clustering of the complete dataset.



(b) Clustering of data-subset 1.

(c) Clustering of data-subset 2.

Figure 8.2: Illustration of K-means clustering both in a centralized and a decentralized setting.

on the objective of the global clustering problem. It should be noted that the mixed-integer programming problems resulting from the K-means clustering problem are very hard to solve due to their weak integer relaxations, which currently makes them intractable in practice. Thus, the approach presented in this thesis can be regarded as preliminary work that can become a viable option with the continuous improvement of mixed-integer programming solvers [KBP+22].

This section presents the mixed-integer programming-based formulation of the training problem. The formulation is subsequently extended to the case of distributedly stored data, which gives rise to a federated learning problem. Consensus constraints are used to couple the training problems of different nodes. These constraints can be dualized such that the federated learning problem can be solved via dual decomposition-based distributed optimization. Since the underlying optimization problem contains integrality constraints it is not convex and thus strong duality does not hold. However, a feasible primal solution can be computed in each iteration through an averaging heuristic.

8.2.1 Mixed-integer programming formulation

The goal of K-means clustering is to assign a set of observations $\mathbf{y}_j \in \mathbb{R}^{n_\mathbf{y}}$, $j \in \mathcal{J} = \{1, \dots, |\mathcal{J}|\}$ to a set of clusters $\mathcal{K} = \{1, \dots, K\}$ and to compute the centroids of each

cluster. The number of clusters is a hyperparameter and is set a priori or in an iterative manner. This problem can be formulated as a mixed-integer nonlinear programming (MINLP) problem [AHL12, GGN21],

$$\min_{w_{jk},\mathbf{m}_k} \sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}} w_{jk}\cdot\|\mathbf{y}_j-\mathbf{m}_k\|_2^2, \tag{8.1a}$$

$$\text{s.t.} \sum_{k\in\mathcal{K}} w_{jk}=1, \forall j\in\mathcal{J}, \tag{8.1b}$$

$$w_{jk}\in\{0,1\},\ \forall j\in\mathcal{J}, k\in\mathcal{K},\ \mathbf{m}_k\in\mathbb{R}^{n_\mathbf{y}}\ \forall k\in\mathcal{K}. \tag{8.1c}$$

The binary variables w_{jk} indicate if observation $\mathbf{y}_j$ is assigned to cluster k and $\mathbf{m}_k$ is the centroid of cluster k. Constraint (8.1b) enforces that each observation is assigned to exactly one cluster, while the objective is to minimize the sum of the squared Euclidean distances of all observations to the centroids of their assigned clusters. Problem (8.1) is a nonconvex MINLP which is hard to solve. In practice, it is more efficient to use a linearized formulation by introducing the variable d_{jk}, which describes the squared distance between an observation j and the centroid of cluster k [GGN21],

$$\min_{w_{jk},d_{jk},\mathbf{m}_k} \sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}} d_{jk}, \tag{8.2a}$$

$$\text{s.t.} \sum_{k\in\mathcal{K}} w_{jk}=1, \forall j\in\mathcal{J}, \tag{8.2b}$$

$$d_{jk}\geq\|\mathbf{y}_j-\mathbf{m}_k\|_2^2-M_j\cdot(1-w_{jk}),\ \forall j\in\mathcal{J}, k\in\mathcal{K}, \tag{8.2c}$$

$$w_{jk}\in\{0,1\}, d_{jk}\geq 0,\ \forall j\in\mathcal{J}, k\in\mathcal{K},\ \mathbf{m}_k\in\mathbb{R}^{n_\mathbf{y}}\ \forall k\in\mathcal{K}. \tag{8.2d}$$

Problem (8.2) is a mixed-integer quadratically constrained program (MIQCP) with a convex integer relaxation. Constraint (8.2c) is an epigraph formulation of the squared Euclidean distance if observation j is assigned to cluster k, i.e., when $w_{jk}=1$. Otherwise, the parameter M_j has to be large enough so that the constraint is trivially satisfied for $w_{jk}=0$. In theory, a common big-M parameter can be used for all constraints described by (8.2c). However, the parameter should be chosen as small as possible to avoid weak integer relaxations. In the following, the big-M parameter is set as

$$M_j=\max_{\boldsymbol{\chi}\in\mathcal{Y}}\|\mathbf{y}_j-\boldsymbol{\chi}\|_2^2,\ \forall j\in\mathcal{J}, \tag{8.3a}$$

$$\mathcal{Y}=\{\mathbf{y}\in\mathbb{R}^{n_\mathbf{y}}|\ \min_{j\in\mathcal{J}}[\mathbf{y}_j]_l\leq[\mathbf{y}]_l\leq\max_{j\in\mathcal{J}}[\mathbf{y}_j]_l,\ l=1.\ldots,n_\mathbf{y}\}. \tag{8.3b}$$

Different approaches have been proposed to solve the clustering optimization problem. Bagirov and Yearwood present a heuristic method based on nonsmooth optimization [BY06], Aloise et al. propose a column generation algorithm [AHL12] and Karmitsa et al. use a diagonal bundle method [KBT17]. Fig. 8.2a illustrates the concept of K-means clustering. The unlabeled data (left) is split into 3 clusters according to their distance to the computed cluster centroid (crosses).

8.2.2 Distributed consensus formulation

Problem (8.2) describes the case in which the entire data set is accessible from a single node. However, this might not always be the case, especially if the underlying data is confidential. In the following it is assumed that the data set is split across several nodes $\mathcal{I} = \{1, \ldots, N_s\}$, with each node having access to the data-subset $\mathcal{J}_i \subset \mathcal{J}$. The MIQCP problem (8.2) can be extended to the case of multiple nodes,

$$\min_{w_{ijk}, d_{ijk}, \mathbf{m}_k} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}_i} \sum_{k \in \mathcal{K}} d_{ijk}, \tag{8.4a}$$

$$\text{s.t.} \sum_{k \in \mathcal{K}} w_{ijk} = 1, \forall i \in \mathcal{I}, j \in \mathcal{J}_i, \tag{8.4b}$$

$$d_{ijk} \geq \|\mathbf{y}_j - \mathbf{m}_k\|_2^2 - M_j \cdot (1 - w_{ijk}),\ \forall i \in \mathcal{I}, j \in \mathcal{J}_i, k \in \mathcal{K}, \tag{8.4c}$$

$$w_{ijk} \in \{0,1\}, d_{ijk} \geq 0,\ \forall i \in \mathcal{I}, j \in \mathcal{J}_i, k \in \mathcal{K},\ \mathbf{m}_k \in \mathbb{R}^{n_\mathbf{y}}\ \forall, k \in \mathcal{K}. \tag{8.4d}$$

The goal of problem (8.4) is again to compute a set of cluster centroids $\mathbf{m}_k$ and to assign the observations of all nodes to these clusters. However, if the nodes cannot share their data, problem (8.4) cannot be solved in a centralized manner. A simple distributed approach would be to solve a clustering problem in each node i. This could lead to a situation as depicted in Fig. 8.2b and Fig. 8.2c. If the data set is split across two nodes, each one can solve a clustering problem. However, both nodes will compute different cluster centroids.

The goal of a federated learning approach is to train a global model, i.e., global cluster centroids in the case of K-means clustering, without sharing the local data between the nodes. To this end each node i can compute individual cluster centroids $\mathbf{m}_{ik}$,

$$\min_{w_{ijk}, d_{ijk}, \mathbf{m}_{ik}} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}_i} \sum_{k \in \mathcal{K}} d_{ijk}, \tag{8.5a}$$

$$\text{s.t.} \sum_{k \in \mathcal{K}} w_{ijk} = 1, \forall i \in \mathcal{I}, j \in \mathcal{J}_i, \tag{8.5b}$$

$$d_{ijk} \geq \|\mathbf{y}_j - \mathbf{m}_{ik}\|_2^2 - M_j \cdot (1 - w_{ijk}),\ \forall i \in \mathcal{I}, j \in \mathcal{J}_i, k \in \mathcal{K}, \tag{8.5c}$$

$$\mathbf{m}_{ik} = \mathbf{m}_{i'k},\ \forall i \in \mathcal{I}, i' \in \mathcal{N}_i, k \in \mathcal{K}, \tag{8.5d}$$

$$w_{ijk} \in \{0,1\}, d_{ijk} \geq 0,\ \forall i \in \mathcal{I}, j \in \mathcal{J}_i, k \in \mathcal{K},$$
$$\mathbf{m}_{ik} \in \mathbb{R}^{n_\mathbf{y}}\ \forall, i \in \mathcal{I}, k \in \mathcal{K}. \tag{8.5e}$$

Since the goal is to obtain global cluster centroids, the individual cluster centroids are coupled through consensus constraints (8.5d), where $\mathcal{N}_i$ contains the set of neighboring nodes of node i. Problem (8.5) describes a set of N_s subproblems coupled through the consensus constraints. In the following subsection dual variables are used to decouple the clustering problems of the different nodes.

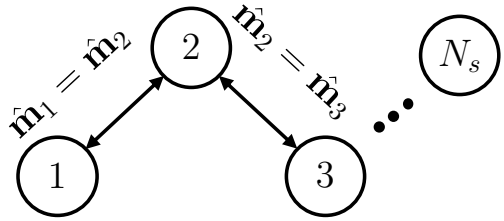


Figure 8.3: Illustration of a linear network topology and the resulting consensus constraints.

8.2.3 Dual decomposition-based distributed clustering

Problem (8.5) can be rewritten as a general constraint-coupled optimization problem by defining the matrix $\mathbf{A}$ describing the connections between the different nodes. In the following, only linear network topologies as depicted in Fig. 8.3 are considered for the sake of simplicity. Note that the discussion in the remainder of this chapter can be easily extended to different network topologies.

By defining the vector of stacked cluster centroids of each node i,

$$\hat{\mathbf{m}}_i \coloneqq \begin{bmatrix} \mathbf{m}_{i,1} \\ \vdots \\ \mathbf{m}_{i,k} \end{bmatrix} \in \mathbb{R}^{K \cdot n_{\mathbf{y}}}, \tag{8.6}$$

the consensus constraints can be rewritten as

$$\hat{\mathbf{m}}_1 - \hat{\mathbf{m}}_2 = \mathbf{0}, \tag{8.7a}$$
$$\hat{\mathbf{m}}_2 - \hat{\mathbf{m}}_3 = \mathbf{0}, \tag{8.7b}$$
$$\vdots$$
$$\hat{\mathbf{m}}_{N_s-1} - \hat{\mathbf{m}}_{N_s} = \mathbf{0}. \tag{8.7c}$$

Constraints (8.7) can subsequently be rewritten in matrix form

$$\underbrace{\begin{bmatrix} \mathbf{I} & -\mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & -\mathbf{I} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} & -\mathbf{I} \end{bmatrix}}_{=:\mathbf{A} \in \mathbb{R}^{K \cdot n_{\mathbf{y}} \cdot (N_s-1) \times K \cdot n_{\mathbf{y}} \cdot N_s}} \cdot \begin{bmatrix} \hat{\mathbf{m}}_1 \\ \hat{\mathbf{m}}_2 \\ \hat{\mathbf{m}}_3 \\ \vdots \\ \hat{\mathbf{m}}_{N_s} \end{bmatrix} = \mathbf{0}, \tag{8.8a}$$

or in a more compact way

$$\sum_{i \in \mathcal{I}} \mathbf{A}_i \hat{\mathbf{m}}_i = \mathbf{0} \tag{8.9}$$

with $\mathbf{A}_i \in \mathbb{R}^{K \cdot n_{\mathbf{y}} \cdot (N_s-1) \times K \cdot n_{\mathbf{y}}}$. By introducing dual variables for the consensus constraints (8.9) the Lagrange function

$$\mathcal{L}(w_{ijk}, d_{ijk}, \mathbf{m}_{ik}, \boldsymbol{\lambda}) = \sum_{i\in\mathcal{I}} \underbrace{\left(\sum_{j\in\mathcal{J}_i} \sum_{k\in\mathcal{K}} d_{ijk} + \boldsymbol{\lambda}^T \mathbf{A}_i \hat{\mathbf{m}}_i \right)}_{=:\mathcal{L}_i(w_{ijk}, d_{ijk}, \mathbf{m}_{ik}, \boldsymbol{\lambda})} \tag{8.10}$$

and subsequently, the dual function of the clustering problem (8.5) can be defined,

$$d(\boldsymbol{\lambda}) \coloneqq \min_{w_{ijk}, d_{ijk}, \mathbf{m}_{ik}} \sum_{i\in\mathcal{I}} \mathcal{L}_i(w_{ijk}, d_{ijk}, \mathbf{m}_{ik}, \boldsymbol{\lambda}), \tag{8.11a}$$

$$\text{s. t.} \sum_{k\in\mathcal{K}} w_{ijk} = 1, \forall i \in \mathcal{I}, j \in \mathcal{J}_i, \tag{8.11b}$$

$$d_{ijk} \geq \|\mathbf{y}_j - \mathbf{m}_{ik}\|_2^2 - M_j \cdot (1 - w_{ijk}),\ \forall i \in \mathcal{I}, j \in \mathcal{J}_i, k \in \mathcal{K}, \tag{8.11c}$$

$$w_{ijk} \in \{0, 1\}, d_{ijk} \geq 0,\ \forall i \in \mathcal{I}, j \in \mathcal{J}_i, k \in \mathcal{K},$$

$$\mathbf{m}_{ik} \in \mathbb{R}^{n_\mathbf{y}}\ \forall, i \in \mathcal{I}, k \in \mathcal{K}. \tag{8.11d}$$

The resulting dual problem can be solved in a distributed manner by solving the individual clustering problems for the current value of the dual variables.

8.2.4 Averaging heuristic

The K-means clustering problem involves integrality constraints and is therefore nonconvex. While the (optimal) value of the dual function (8.11) provides a lower bound on the optimal value of the primal problem (8.5), the feasibility of the primal problem is not guaranteed upon the convergence of a dual decomposition-based algorithm, i.e., the consensus constraints may not be satisfied. Nevertheless, in the case of K-means clustering it is straightforward to compute a feasible primal solution using an averaging step. In each iteration t of a dual decomposition-based algorithm, the coordinator communicates the dual variables $\boldsymbol{\lambda}^{(t)}$ to the nodes. The nodes in turn solve their clustering problems and communicate their computed cluster centroids $\hat{\mathbf{m}}_i(\boldsymbol{\lambda}^{(t)})$ to the coordinator. Based on this response the coordinator can compute the average of the primal variables, i.e., the average cluster centroids,

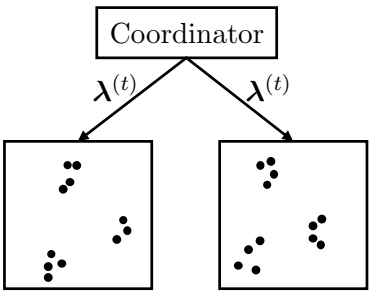
$$\overline{\mathbf{m}}_k(\boldsymbol{\lambda}^{(t)}) = \frac{1}{N_s} \sum_{i\in\mathcal{I}} \mathbf{m}_{ik}(\boldsymbol{\lambda}^{(t)}) \tag{8.12}$$

which are then communicated back to the nodes. Using the mean cluster centroids the nodes can compute their resulting primal objective value

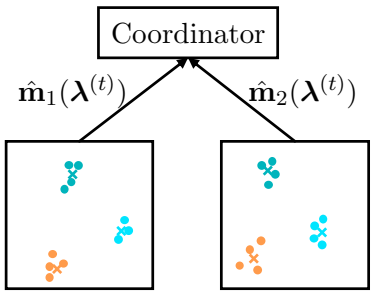
$$z_i(\boldsymbol{\lambda}^{(t)}) = \sum_{j\in\mathcal{J}} \min_{k\in\mathcal{K}} \|\mathbf{y}_j - \overline{\mathbf{m}}_k(\boldsymbol{\lambda}^{(t)})\|_2^2. \tag{8.13}$$

The primal objective value can be used to compute the relative duality gap in each iteration,

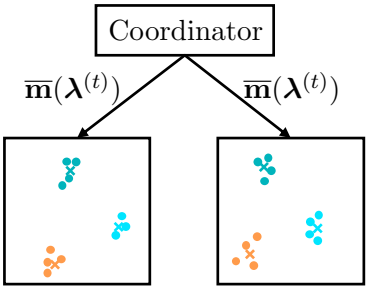
$$\text{rel. DG} = 100 \cdot \left(1 - \frac{d(\boldsymbol{\lambda}^{(t)})}{\sum_{i\in\mathcal{I}} z_i(\boldsymbol{\lambda}^{(t)})}\right). \tag{8.14}$$



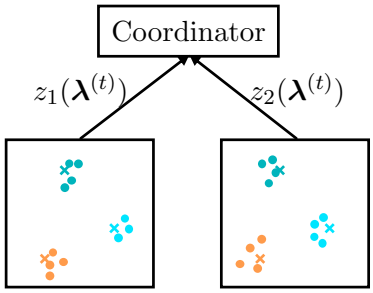
(a) The coordinator sends the current dual variables to the nodes.



(b) The nodes compute their cluster centroids and send them to the coordinator.



(c) The coordinator computes the average cluster centroids and sends them to the nodes.



(d) The nodes compute their objectives based on the received average centroids.

Figure 8.4: Communication process between the coordinator and the nodes in iteration t.

Since the value of the dual function provides a lower bound on the optimal primal objective value the relative duality gap can be used to assess the distance of a found solution to the global optimum. The entire communication process between the coordinator and the nodes is illustrated in Fig. 8.4. Note that the average cluster centroids are only used to compute the duality gap. They do not influence the update of the dual variables.

8.2.5 Symmetry breaking constraints

The clustering problem (8.5) is highly symmetric, i.e., it contains solutions with the same objective values. This is because the index assigned to a cluster does not influence the objective function. Fig. 8.5 illustrates the situation of two symmetric solutions. This symmetry can lead to problems for the averaging heuristic presented in the previous section, as the computed cluster centroids of a single node can switch from one iteration to the next. For instance, while some points are assigned to cluster k in iteration t, they could be assigned to cluster k' in iteration $t+1$ by switching the centroids of clusters k and k' without affecting the objective.

To prevent this behavior symmetry breaking constraints are added to the optimization problems of the nodes. In the first iteration, one of the nodes acts as the reference node, providing reference centroids $\overline{\mathbf{m}}_k^{\text{ref}}$. In the subsequent iterations the quadratic constraint

$$\|\mathbf{m}_{ik} - \overline{\mathbf{m}}_k^{\text{ref}}\|_2^2 \leq \|\mathbf{m}_{ik'} - \overline{\mathbf{m}}_k^{\text{ref}}\|_2^2, \forall k, k' \in \mathcal{K}, \tag{8.15}$$

is added to each node i. This ensures that cluster k of each node i will be the one closest to

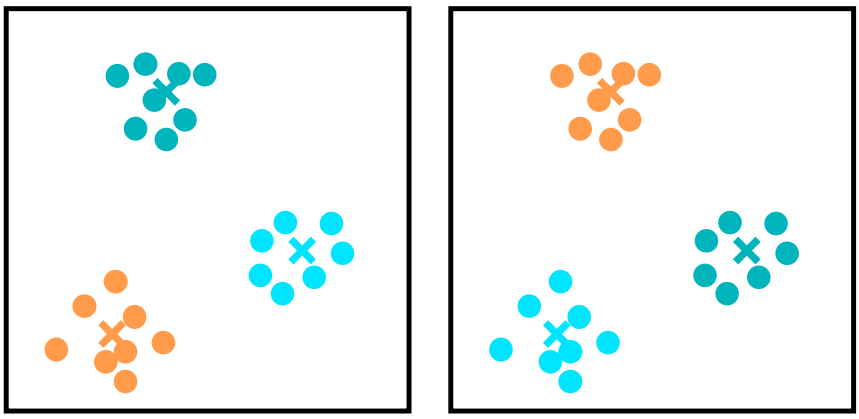


Figure 8.5: Example of symmetric clustering solutions. In the two cases, the data points are assigned to different clusters without affecting the objective function.

the reference centroid $\overline{\mathbf{m}}_k^{\text{ref}}$. The choice of the node which provides the reference centroid can be performed arbitrarily, as it does not affect the optimization of the other nodes. Furthermore, the added constraint also does not affect the optimal objective value while rendering all symmetric solutions, except for one, infeasible.

8.3 Numerical analysis for distributed clustering problems

The dual decomposition-based distributed clustering approach was evaluated on a set of benchmark problems of varying sizes. The data for each benchmark problem was generated randomly. First, initial cluster centroids $\mathbf{m}_k^0$ were generated, with $[\mathbf{m}_k^0]_l \in \mathcal{U}_c(-1, 1),\ l = 1, \ldots, n_{\mathbf{y}}$. Then, for each cluster k five random data points were added within a radius of 0.5 from the generated centroid. The parameters of the benchmark problems were varied as follows:

$$\begin{aligned}
&\text{Number of nodes: } N_s \in \{2, 3, 4\},\\
&\text{Number of dimensions: } n_{\mathbf{y}} \in \{2, 3, 4\},\\
&\text{Number of clusters: } K \in \{3, 4\}.
\end{aligned}$$

Five benchmark problems were generated for each combination of nodes, dimensions, and clusters, resulting in a total of 90 benchmark problems. A benchmark problem is characterized by its number of nodes, dimension of the data, and number of clusters. For instance, problem $3\text{N}2\text{D}4\text{K}_5$ is the 5th benchmark problem comprised of 3 nodes with 2-dimensional data sorted into 4 clusters.

The benchmark problems were solved using the subgradient method, the bundle trust method, and the quasi-Newton dual ascent algorithm. The use of ADMM was omitted for several reasons. First, in each communication round a feasible primal solution is obtained through the averaging heuristic (cf. Sec. 8.2.4). This primal solution does not correspond to the current dual variables. Due to the nonconvexity of the underlying MIP problem, no

Table 8.1: Parameter settings of the distributed optimization algorithms for the clustering benchmark problems.

	Value	Description	Algorithms
$\boldsymbol{\lambda}^{(0)}$	$\mathbf{0}$	initial dual variables	All
$\alpha^{(0)}$	0.5	initial step size/trust region parameter	All
$t_{\max}$	150	maximum number of iterations	All
ϵ_p	10^{-2}	primal residual convergence tolerance	All
ϵ_{DG}	0.25 %	relative duality gap tolerance	All
ϵ_b	1	bundle cuts threshold	QNDA
τ	50	allowed age of data points	BTM, QNDA
$\mathbf{B}^{(0)}$	$-\mathbf{I}$	initial approximated Hessian	QNDA

guarantee can be made that the consensus constraints will be satisfied at the dual optimum. This in turn means that the regularization term in ADMM might not vanish, which would result in different objective values of the Lagrange function and the augmented Lagrange function, leading to an overestimation of the dual value and a subsequent underestimation of the duality gap, Second, the regularization of ADMM introduces a bias towards the mean cluster centroids. Note that the averaging heuristic does not affect the iterations of the other distributed optimization algorithms. It merely serves to compute a feasible primal solution, i.e., an upper objective bound, in each iteration. Introducing a bias towards the mean centroids in the solution of the clustering problems of the nodes would result in a stagnation of the algorithm. For badly chosen regularization parameters all nodes would converge towards the initial mean centroids, which is not the case in general for the other algorithms. The QADA algorithm could also be used to solve the clustering problem. However, the numerical tests showed that the BTM and QNDA algorithms are already efficient enough to converge within the sampling phase of QADA. Its inclusion in the results was therefore omitted.

The initial step size (SG)/ trust region (BTM, QNDA) parameter was set to $\alpha^{(0)} = 0.5$ and varied according to

$$\alpha^{(t)} = \alpha^{(0)}/\sqrt{t}. \tag{8.16}$$

The bundle cuts for QNDA were used in every iteration, i.e., $\epsilon_b = 1$ and $\tau = 50$ points were used to construct the bundle in BTM and the bundle cuts in QNDA respectively. All algorithms were initialized with $\boldsymbol{\lambda}^{(0)} = \mathbf{0}$ and the initial approximated Hessian of the QNDA algorithm was set to the negative identity matrix. The algorithms were terminated either when the Euclidean norm of the primal residual

$$\|\mathbf{w}_p\|_2 = \left\| \sum_{i \in \mathcal{I}} \mathbf{A}_i \hat{\mathbf{m}}_i \right\|_2, \tag{8.17}$$

Table 8.2: Summary of the results for the distributed optimization of the clustering benchmark problems, $\bar{t}$: mean number of iterations until termination, $\overline{\text{rel. DG}}$: mean relative duality gap upon termination (in %), $\overline{T_{\text{comp}}}$: mean computation time (in s).

Algorithm	$\bar{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$
SG	136.75	2.27	996.28
BTM	57.44	1.86	515.77
QNDA	**54.48**	**1.81**	**483.22**

i.e., the violation of the consensus constraints, lied below a threshold of $\epsilon_p = 10^{-2}$ or when the relative duality gap (8.14) reached a value of $\epsilon_{DG} = 0.25$ %. The used parameters for the different algorithms are summarized in Tab. 8.1. The MIQCP clustering problems of all nodes were solved using the commercial solver Gurobi and the total computation time was obtained through eq. (6.1), with $T_{\text{comm}} = 800\ ms$.

The results for the clustering benchmarks are summarized in Tab. 8.2. Out of the examined algorithms, QNDA shows the best performance in terms of the required number of iterations and computation time as well as in terms of the achieved relative duality gap. The BTM algorithm shows similar performance in terms of the number of iterations and the achieved duality gap. However, in the case of distributed clustering, each iteration is costly due to the underlying MIQCP problems. Therefore, a slight improvement in the number of iterations results in a more substantial performance improvement in terms of computation times. More detailed results for the clustering benchmarks are summarized in Tab. C.5 in the appendix.

Fig. 8.6 shows the evolution of the relative duality gap for benchmark problem 2N2D4K$_3$. The subgradient method converges rather slowly. In comparison, the BTM and QNDA algorithms exhibit a faster rate of convergence. Between these two algorithms, BTM exhibits an oscillatory behavior before converging. In contrast, the QNDA algorithm does not exhibit oscillations and therefore converges earlier. Additionally, it should be noted

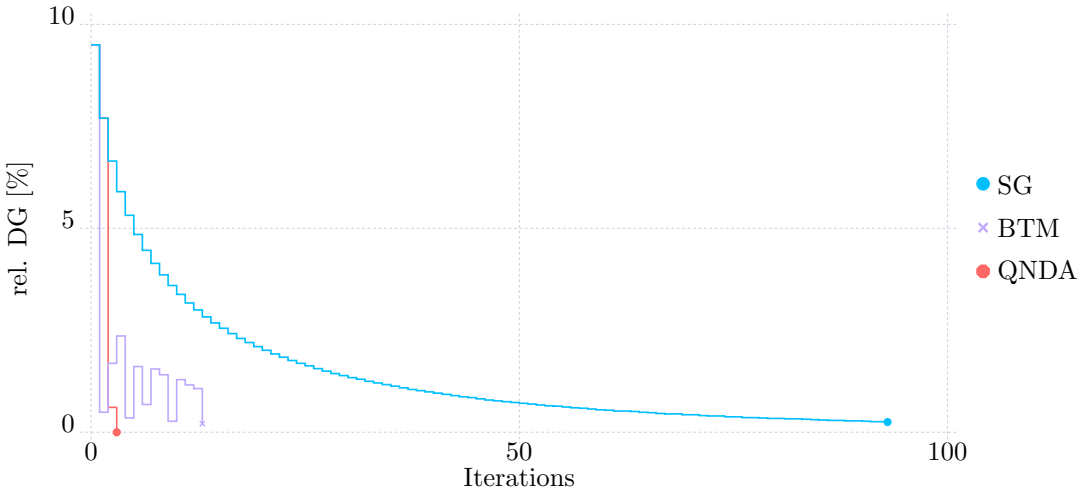
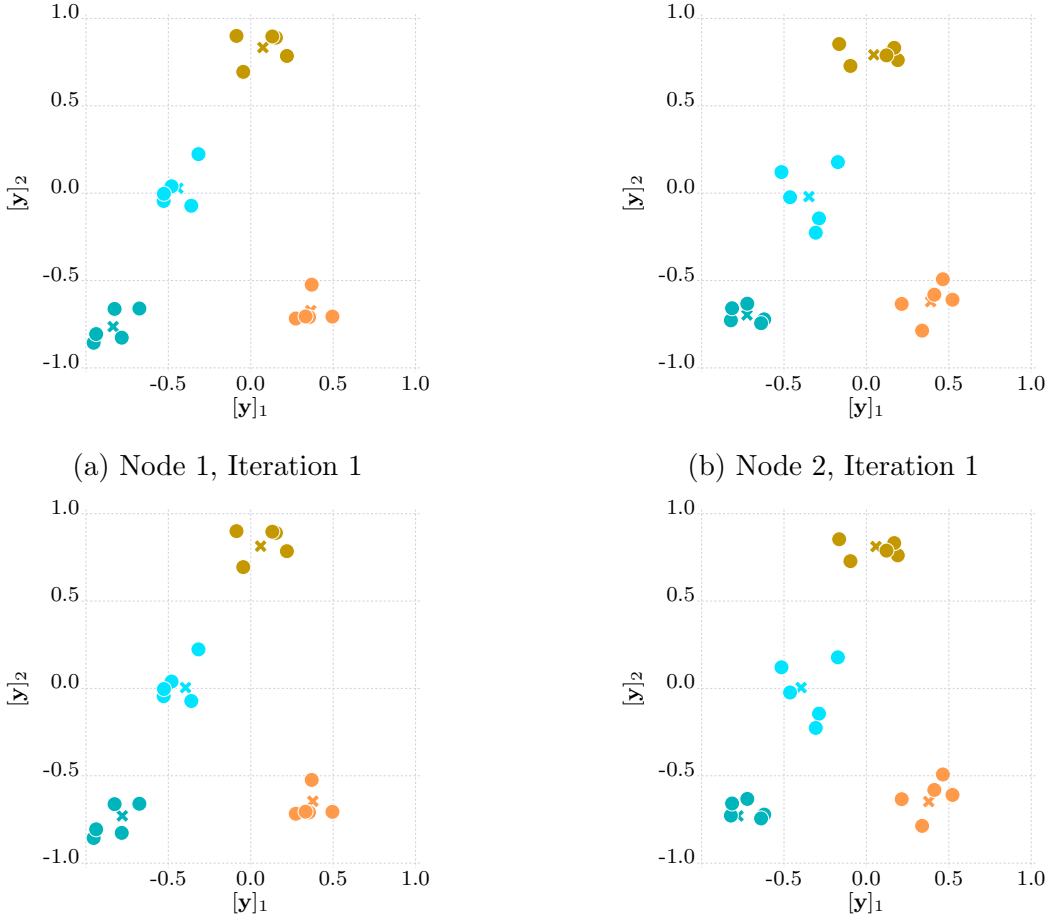


Figure 8.6: Evolution of the relative duality gap for benchmark problem 2N2D4K$_3$.



(a) Node 1, Iteration 1

(b) Node 2, Iteration 1

(c) Node 1, Iteration 4

(d) Node 2, Iteration 4

Figure 8.7: Exemplary clusters in different iterations of the QNDA algorithm for benchmark problem 2N2D4K$_3$.

that the QNDA algorithm achieves a relative duality gap of 0 %, i.e., it converges to a proven global optimum.

Fig. 8.7 provides some further illustrations of the results. Fig. 8.7a and Fig. 8.7b show the results of the clustering in the first iteration, i.e., the individual global optima. Fig. 8.7c and Fig. 8.7d depict the solutions upon the convergence of the QNDA algorithm. It can be seen that each node computes the same cluster centroids corresponding to the globally optimal solution for the entire data set, but not to the individual data sets. It is therefore possible to compute a global model locally in each node while only accessing local data.

8.4 Comparison to the central solution

As shown in the previous section, solving the MIQCP clustering problems is computationally expensive. This is due to the weak integer relaxation of problem (8.2), which means that the solution of the relaxed problem within the branch-and-bound algorithm is far away from the integer solution. This results in slow-moving relative integrality gaps and slow convergence of the solution algorithm. While the main motivation of the distributed

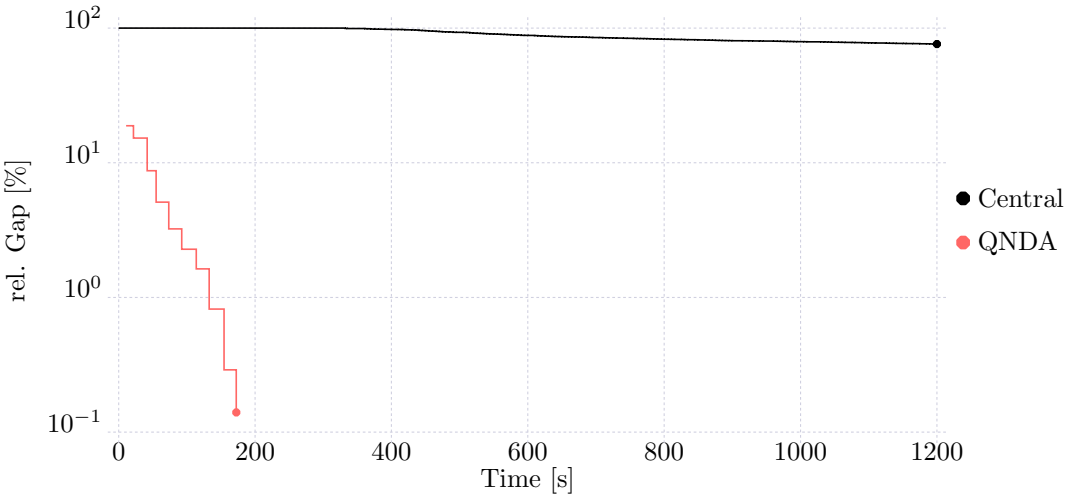


Figure 8.8: Evolution of the relative duality gap of QNDA compared to the relative integrality gap of the central solution using Gurobi for benchmark problem 4N4D4K$_3$.

clustering approach is the training of a global model without the exchange of local data, it can also be used to efficiently solve larger clustering problems. Fig. 8.8 depicts the evolution of the relative duality gap of the QNDA algorithm as well as the evolution of the relative integrality gap of Gurobi for the complete data set of benchmark problem 4N4D4K$_3$. The clustering problems of the individual nodes were solved sequentially in the case of QNDA, also using Gurobi. While the relative gap of the central solution improves very slowly, the QNDA algorithm quickly converges to a solution close to the global optimum. Note, that both relative gaps prove a worst-case distance to the global optimum. Hence, decomposing a large clustering problem into smaller subproblems and coordinating the solutions via a distributed optimization algorithm can offer significant performance improvements compared to a central solution.

8.5 Conclusion

This chapter demonstrated how dual decomposition-based distributed optimization can be applied to the solution of clustering problems. The approach ensures privacy, i.e., enables federated learning, as each node only has access to its local data. A global model can still be obtained by coordinating the solutions of the individual clustering problems. Numerical tests on a set of benchmark problems demonstrated that the QNDA algorithm outperforms the subgradient method and the BTM algorithm. Furthermore, the distributed optimization approach exhibited superior performance compared to a central solution approach. In the future, the developed algorithms can also be applied to other federated learning problems, like the distributed training of support vector machines (cf. Sec. 9.2.1). Additionally, they can be integrated into mixed-integer programming solvers to obtain tighter lower bounds for problems with weak integer relaxations within the branch-&-bound algorithm.

9 Conclusion and Outlook

In the following, the thesis is summarized. Afterward, an extensive, but not exhaustive, overview of potential future research directions is provided.

9.1 Summary

The goal of this thesis was the development of dual decomposition-based distributed optimization algorithms with improved rates of convergence compared to existing methods, without the need of sharing confidential information of the subsystems. Many contributions in the literature focus on distributed optimization of problems without individual constraints. This leads to a dual optimization problem that is convex, smooth, and whose gradients can easily be computed in a distributed manner. Furthermore, the bulk of research on dual decomposition-based distributed optimization has focused on convex problems, which exhibit desirable properties like strong duality. However, real-world applications often exhibit more challenges, like individual constraints of the subproblems, integrality constraints on certain variables, or general nonconvexities. These issues can deteriorate the performance of distributed optimization approaches or limit their applicability.

Within this thesis, two new efficient dual decomposition-based distributed optimization algorithms were presented that aimed at addressing the aforementioned issues. Both are based on the approximation of the dual function by a quadratic function. The quadratically approximated dual ascent (QADA) algorithm solves a regression problem based on information collected from previous iterations to estimate the parameters of the quadratic surrogate function. The quasi-Newton dual ascent (QNDA) algorithm updates the approximated Hessian of the dual function through a BFGS update. The update of the dual variables for both algorithms is subject to step size constraints. In contrast to the primal residual which was approximated in previous work based on quadratic surrogates, the dual function is concave, regardless of the problem class of the primal optimization problem. However, the dual function is usually nonsmooth, if the set of active individual constraints changes. This nonsmoothness was addressed by constructing cutting planes using subgradients from previous iterations and incorporating them into the update of the dual variables as additional constraints.

Results for a large number of general benchmark problems showed the efficiency of the

proposed algorithms. A remarkable result is that dual decomposition-based distributed optimization algorithms can be used to speed up the solution of mixed-integer programs where a centralized solution does not converge in reasonable amounts of time. This is especially true for problems with weak continuous relaxations, e.g., due to a large number of big-M constraints or general integrality constraints[4]. In such cases decomposing the system-wide optimization problems leads to much more tractable subproblems and the resulting dual problems yield a tighter lower bound than the continuous relaxations within shorter computation times. While the QADA algorithm showed superior performance for distributed MIQPs, the QNDA algorithm outperformed other state-of-the-art algorithms for general distributed convex problems. For distributed QPs the QADA and QNDA algorithms showed a similar performance, outperforming other algorithms. In addition to the general benchmark problems, two potential applications for dual decomposition-based distributed optimization were presented. Distributed linear model predictive control (DMPC) is an active field of research that can greatly benefit from more efficient distributed optimization algorithms. Dual decomposition is especially suited for problems where the involved subproblems cannot share their system models and constraints. Another application where data sovereignty is critical is machine learning. Here, distributed optimization can be employed to compute a global model with subproblems only needing access to local data. Some further applications of distributed optimization as well as some proposed algorithmic improvements are briefly reviewed in the following section.

9.2 Future research

Future research can progress in two directions, applications, and algorithmic improvements. Some suggestions are provided in the following sections.

9.2.1 Applications

This section briefly discusses how the DMPC approach for linear systems presented in Chapter 7 can be applied to systems with nonlinear dynamics. Furthermore, the application of dual decomposition-based distributed optimization in an industrial environment in the case of demand-side management is discussed. A federated learning application in the form of distributed training of support vector machines is also presented. Finally, an approach to integrate distributed optimization into the parallelized solution of mixed-integer nonlinear programming (MINLP) problems and an embedding in branch-&-bound algorithms is proposed.

[4]In this case this refers to variables being able to take integer values other than 0 and 1.

Distributed nonlinear model predictive control

Chapter 7 dealt with the distributed optimization of MPC problems with linear dynamics. Industrial applications of MPC traditionally relied on linear or linearized dynamic models, which leads to favorable properties of the underlying optimization problem, e.g., convexity. However, the improved performance of nonlinear programming algorithms renders the inclusion of nonlinear dynamics in the form of constraints a viable option, leading to nonlinear MPC (NMPC) applications. In the case of subsystems with nonlinear dynamics coupled through shared limited resources, distributed NMPC problems of the form

$$\min_{\mathbf{x}^{0:N_p},\mathbf{u}^{0:N_p-1}} \sum_{i\in\mathcal{I}} \left[J_i^f(\mathbf{x}_i^{N_p}) + \sum_{k=0}^{N_p-1} J_i(\mathbf{x}_i^k, \mathbf{u}_i^k) \right], \tag{9.1a}$$

$$\text{s.t. } \mathbf{x}_i^{k+1} = \mathbf{F}_i(\mathbf{x}_i^k, \mathbf{u}_i^k),\ \forall i \in \mathcal{I}, k = 0, \dots, N_p - 1, \tag{9.1b}$$

$$\mathbf{x}_i^0 = \tilde{\mathbf{x}}(t_0),\ \forall i \in \mathcal{I}, \tag{9.1c}$$

$$\mathbf{x}_i^k \in \mathcal{X}_i \subset \mathbb{R}^{n_{\mathbf{x}_i}},\ \forall i \in \mathcal{I}, k = 0, \dots, N_p, \tag{9.1d}$$

$$\mathbf{u}_i^k \in \mathcal{U}_i \subset \mathbb{R}^{n_{\mathbf{u}_i}},\ \forall i \in \mathcal{I}, k = 0, \dots, N_p - 1, \tag{9.1e}$$

$$\sum_{i\in\mathcal{I}} \mathbf{R}_i(\mathbf{x}_i^k, \mathbf{u}_i^k) \leq \mathbf{r}_{\max}^k,\ k = 0, \dots, N_p - 1, \tag{9.1f}$$

can arise, with nonlinear and possibly nonconvex functions $\mathbf{F}_i \colon \mathbb{R}^{n_{\mathbf{x}_i}} \times \mathbb{R}^{n_{\mathbf{u}_i}} \to \mathbb{R}^{n_{\mathbf{x}_i}}$ and $\mathbf{R}_i \colon \mathbb{R}^{n_{\mathbf{x}_i}} \times \mathbb{R}^{n_{\mathbf{u}_i}} \to \mathbb{R}^{n_{\mathbf{r}}}$. Dual decomposition-based distributed optimization can generally be applied to distributed NMPC problems, therefore the performance of the proposed algorithms could be compared to existing methods. Naturally, challenges like nonconvexity and the resulting duality gap have to be addressed in this case. However, these issues are related to dual decomposition in general, not only to the proposed algorithms.

Distributed demand-side management

One of the most commonly shared limited resources is electricity, or, more generally, energy. Due to the increased penetration of renewables into the electricity grid, both consumers and producers of electricity have to become more flexible due to the resulting variability in the available energy.

This variability can lead to an over-utilization of the available energy. An example is depicted in Fig. 9.1, where two consumers connected to the same grid decentrally plan their load profiles. However, due to the limited availability of electricity in the grid, constraints on the maximum load might be violated. Demand-side management (DSM) or demand response (DR) refers to the ability of energy consumers to adapt their operations according to the current availability of energy [SSS+16]. This availability is usually reflected by prices set by the grid operator. The goal is to incentivize consumers to operate during periods of high availability and discourage them from operating when availability is low. Dual decomposition-based distributed optimization can be applied to this price-based

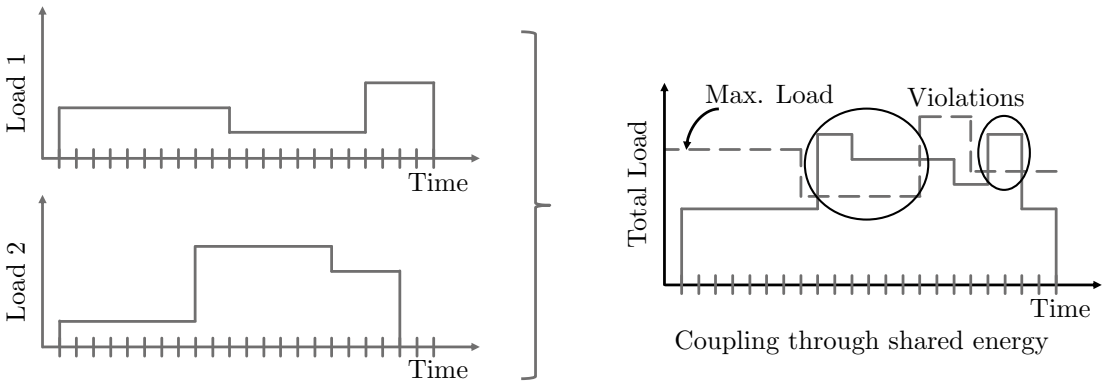


Figure 9.1: Example of two consumers with decentrally optimized load profiles. Due to the lack of coordination, the total load violates the constraint on the maximum available energy.

coordination mechanism, with the grid operator acting as the coordinator and the individual consumers as the subproblems. An extensive overview of DSM is given by Zhang and Grossmann [ZG16].

Yfantis et al. [YMB+22] proposed an architecture for distributed demand-side management, which is depicted in Fig. 9.2, extending the approach of Motsch et al. [MDS+20]. The consumers are depicted as cyber-physical production systems (CPPS), each of which executes its load control and scheduling by solving a mixed-intger programming problem. The planning functions receive actual energy prices from the grid and try to minimize their cost and makespan. Furthermore, a distributed controller in the form of a dual decomposition-based distributed optimization algorithm is added to compute shadow prices for the available energy, which serves to coordinate the individual CPPSs.

Distributed energy management can also be performed on the process control level, which results in MPC problems similar to the ones examined in Chapter 7. For instance, Biegel et al. [BAS+12] apply dual decomposition-based DMPC to congestion management in a grid.

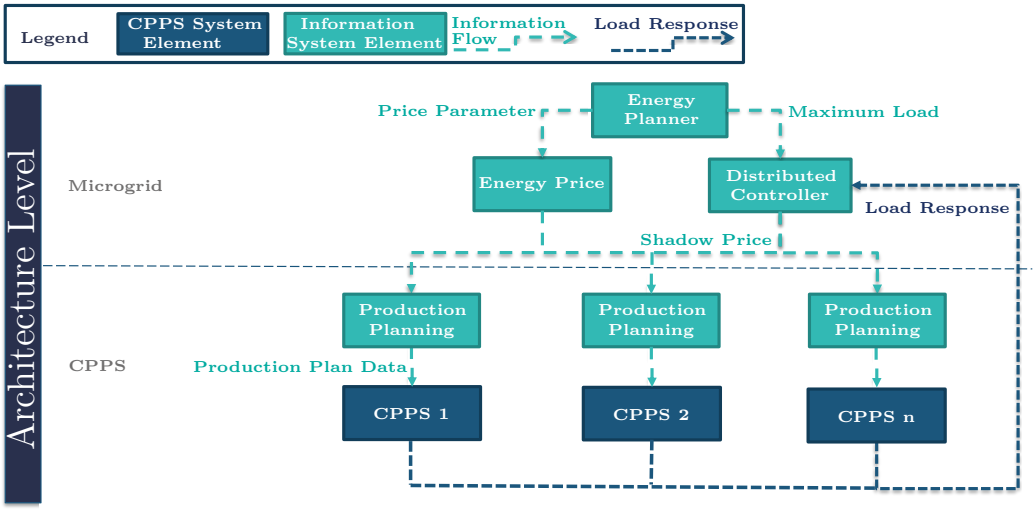


Figure 9.2: Proposed architecture for distributed demand-side management (from [YMB+22] © 2022 IEEE).

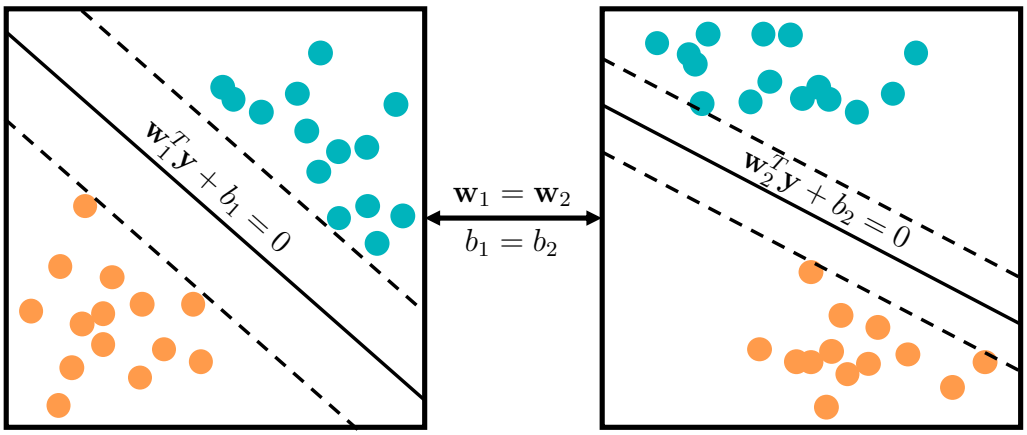


Figure 9.3: Example of distributed training of support vector machines linked by consensus constraints.

The goal of the approach for the subsystems is to minimize the deviation from previously contractually agreed load profiles. If coordination is performed on the planning level, the main challenge becomes the presence of integrality constraints and the resulting large-scale MIP problems. It was shown throughout this thesis that the proposed algorithms can be used to distributedly solve MIP problems, hence their application to distributed DSM problems can provide an interesting research application.

Distributed training of support vector machines

Chapter 8 showed how dual decomposition-based distributed optimization can be applied to machine learning. In this thesis, clustering problems were examined, which constitute unsupervised learning problems modeled as mixed-integer programs. In the future, the proposed algorithms could be applied to other machine learning problems. An example of a supervised learning problem that can be tackled by distributed optimization is the training of support vector machines [MRT18]. The training task is to compute a normal vector $\mathbf{w}$ and a bias term b of a hyperplane $\mathbf{w}^T\mathbf{y} + b = 0$, so that a set of data points $\mathbf{y}_j$ is classified according to their labels $z_j \in \{-1, 1\}$. The training problem can be formulated as a convex quadratic program [MRT18]

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2, \tag{9.2a}$$

$$\text{s.t. } z_j \cdot (\mathbf{w}^T\mathbf{y}_j + b) \geq 1, \ \forall j \in \mathcal{J}. \tag{9.2b}$$

If data is distributed across several nodes, as shown in Fig. 9.3, the individual problems can be linked through consensus constraints, which can then be dualized [FCG11]. Since problem (9.2) is convex, solving the dual problem also leads to a globally optimal primal solution.

Parallelization of convex MINLPs

It was shown in Chapters 6 and 8, that dual decomposition-based distributed optimization can be used to speed up the solution of mixed-integer programs. In the case of mixed-integer linear programs, powerful commercial solvers like Gurobi [Gur23], CPLEX [IBM22] or Xpress [FIC22] enable the solution of problems with millions of variables and constraints. Furthermore, they can also handle more complex problem classes, e.g., mixed-integer quadratic programs (MIQP) or mixed-integer quadratically constrained quadratic programs (MIQCQP), both with convex and nonconvex integer relaxations. However, mixed-integer nonlinear programming (MINLP) problems are still very challenging to solve. Kronqvist et al. [KBL+19] provide a survey of both open-source and commercial MINLP solvers.

In some cases, it might be possible to take advantage of the structure of an MINLP problem to apply dual decomposition-based distributed optimization. For instance, an MINLP problem of the form

$$\min_{\mathbf{x}_i,\mathbf{y}_i} \sum_{i\in\mathcal{I}_c} f_i(\mathbf{x}_i) + \sum_{i\in\mathcal{I}_d} \mathbf{c}_i^T\mathbf{y}_i, \tag{9.3a}$$

$$\text{s. t.} \sum_{i\in\mathcal{I}_c} \mathbf{A}_i\mathbf{x}_i + \sum_{i\in\mathcal{I}_d} \mathbf{A}_i\mathbf{y}_i \leq \mathbf{b}, \tag{9.3b}$$

$$\mathbf{x}_i \in \mathcal{X}_i \subset \mathbb{R}^{n_{\mathbf{x}_i}},\ \forall i \in \mathcal{I}_c, \tag{9.3c}$$

$$\mathbf{y}_i \in \mathcal{Y}_i \subset \mathbb{Z}^{n_{\mathbf{y}_i}},\ \forall i \in \mathcal{I}_d, \tag{9.3d}$$

with continuous variables $\mathbf{x}_i$, integer variables $\mathbf{y}_i$, convex individual continuous objective functions $f_i(\mathbf{x}_i)$, convex continuous constraints $\mathcal{X}_i$ and a compact polyhedral constraint set $\mathcal{Y}_i$ can be decomposed by introducing dual variables for the system-wide constraints (9.3b). This would result in two types of subproblems, convex problems

$$\min_{\mathbf{x}_i} \sum_{i\in\mathcal{I}_c} f_i(\mathbf{x}_i) + \boldsymbol{\lambda}^T\mathbf{A}_i\mathbf{x}_i, \tag{9.4a}$$

$$\text{s. t. } \mathbf{x}_i \in \mathcal{X}_i \subset \mathbb{R}^{n_{\mathbf{x}_i}}, \tag{9.4b}$$

and integer problems

$$\min_{\mathbf{y}_i} \mathbf{c}_i^T\mathbf{y}_i + \boldsymbol{\lambda}^T\mathbf{A}_i\mathbf{y}_i, \tag{9.5a}$$

$$\text{s. t. } \mathbf{y}_i \in \mathcal{Y}_i \subset \mathbb{Z}^{n_{\mathbf{y}_i}}, \tag{9.5b}$$

which can be solved in a distributed manner. Problem (9.3) is still nonconvex due to the integrality constraints, hence strong duality does not apply. The value of the dual function still provides a lower bound on the primal objective. A feasible primal solution, i.e., one that satisfies the system-wide constraints, could be obtained by fixing the results $\mathbf{y}_i(\boldsymbol{\lambda}^{(t)})$ of the MILP subproblems in each iteration t of the dual decomposition-based algorithm and re-solving the resulting convex problem,

$$\min_{\mathbf{x}_i \in \mathcal{X}_i,\, \forall i \in \mathcal{I}_c} \sum_{i \in \mathcal{I}_c} f_i(\mathbf{x}_i), \tag{9.6a}$$

$$\text{s. t.} \sum_{i \in \mathcal{I}_c} \mathbf{A}_i \mathbf{x}_i \leq \mathbf{b} - \sum_{i \in \mathcal{I}_d} \mathbf{A}_i \mathbf{y}_i(\boldsymbol{\lambda}^{(t)}). \tag{9.6b}$$

Problem (9.6) is a convex problem that can be efficiently solved centrally, or distributedly using dual decomposition, i.e., by using an inner distributed optimization loop within the outer loop of problem (9.3). If a feasible primal solution is obtained, the resulting duality gap can be used to assess its distance to the globally optimal solution. Parallelizing the solutions of problems (9.4), (9.5) and (9.6) on the same machine can help to quickly find a good warm-start solution for the central problem, including a cutting plane through the lower bound obtained from the dual value. Naturally, convergence cannot be guaranteed in this case. Therefore, extensive numerical tests can be used to validate the proposed approach.

Embedding in branch-&-bound algorithms

The results in Sec. 8.4 showed that solving the dual problem can yield significantly tighter bounds for MIP problems with weak continuous relaxation. Klostermeier et al. compared the lower bounds computed by the subgradient method, BTM and QNDA to the best bounds obtained in the search tree of three commercial MIP solvers [KYW+24]. They especially examined the influence of the parallelization of the solution of the subproblems on the performance of the dual decomposition-based distributed optimization algorithms. A bound computed by the dual problem prior to solving the root node of the branch-&-bound tree can be used in a termination criterion as long as the best bound from the continuous relaxations is weaker. Furthermore, the bound can be used to assess the quality of a heuristically computed feasible solution.

Another way to embed the presented algorithms into a branch-&-bound tree is by solving the dual problem within a parent node to compute a lower bound on a potential feasible solution in one of its child nodes. If an incumbent, i.e., a feasible solution, has already been found within the tree it provides an upper bound on the objective of the global optimum of the MIP problem (in case of a minimization problem). If the continuous relaxations of the MIP are weak, solving the dual problem may provide a tighter bound within a node. If the computed dual bound is larger than the incumbent's objective value, the node can be pruned from the search tree, as a potential feasible integer solution in this node or one of its child nodes could not improve the incumbent. Note that the bounds added to the nodes of the search tree by branching do not affect the separability of the problem. Thus, if the structure of the problem allows for it, the solution of the dual problem within a node is highly parallelizable.

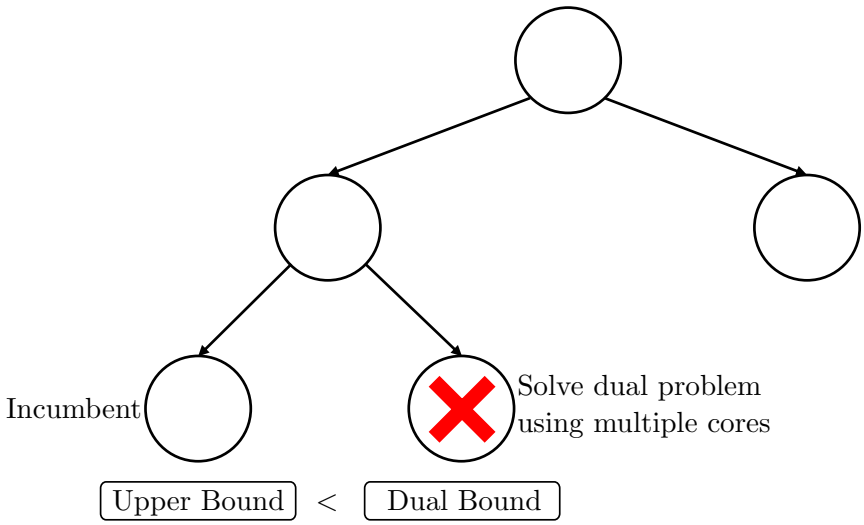


Figure 9.4: Illustration of the pruning of a node in the branch-&-bound tree due to a bound computed by a dual decomposition-based algorithm.

9.2.2 Algorithmic improvements

While the QADA and QNDA algorithms showed good performance in the conducted numerical experiments, there is still potential for improvement. Some suggestions are made in the following sections.

Combination of QADA and QNDA

The two algorithms were combined in this thesis by using the QNDA algorithm as an initialization within the sampling phase of the QADA algorithm. Some benchmark problems showed, that while QADA converged quickly to the vicinity of the optimum, many iterations were needed to finally converge to that optimum. In these instances, QNDA exhibited a more robust behavior while requiring more iterations to reach the vicinity of the optimum. The benefits of both algorithms could be exploited by switching back to QNDA updates, once the QADA algorithm has quickly reached the vicinity of the optimum.

A drawback of the QADA algorithm is the long sampling phase. In total $n_{\text{reg,min}}$ iterations are required to perform a regression. A regression problem could be solved before the necessary number of data points have been collected, which would however lead to an underdetermined system. One way to compute a surrogate quadratic function without sufficient sampling points could be to use a convex combination of the approximations computed by QNDA and by the underdetermined regression of QADA,

$$d(\boldsymbol{\lambda}) \approx \beta \cdot d_Q(\boldsymbol{\lambda}) + (1-\beta) \cdot d_B(\boldsymbol{\lambda}),\ \beta \in [0,1]. \tag{9.7}$$

As more sampling points are added, the parameter β could be gradually increased.

Predictor-Corrector approach

Both the QADA and QNDA algorithms compute an approximation of the dual function in each iteration. This surrogate dual function is then used to update the dual variables. After the dual variables have been updated and sent to the subproblems, the coordinator receives back the actual dual function value corresponding to the previously computed dual variables. The deviation of the actual dual value from the predicted dual value could be used as additional information to improve the approximation in the next iterations within a predictor-corrector-type approach.

Enforced negative definiteness

Depending on the data used for the approximation, both QADA and QNDA can yield a nonconvex surrogate dual function. This not only deteriorates the quality of the approximation since the underlying dual function is always concave but also renders the update step computationally expensive due to the resulting nonconvex problem. In the case of QADA, this situation could be avoided by solving the regression problem as a semi-definite program and adding constraints on the definiteness of the approximated Hessian. In the case of QNDA, negative definiteness can be enforced by adding a regularization term to the BFGS-update, which would necessitate further tuning parameters [EMR16, EMR17], or by computing the approximated Hessian via an update of its Cholesky factorization [Pow87].

Sparsity

The network topology of the distributed optimization problem, i.e., the sparsity structure of the system-wide constraint matrix $\mathbf{A}$ influences the structure of the dual function.

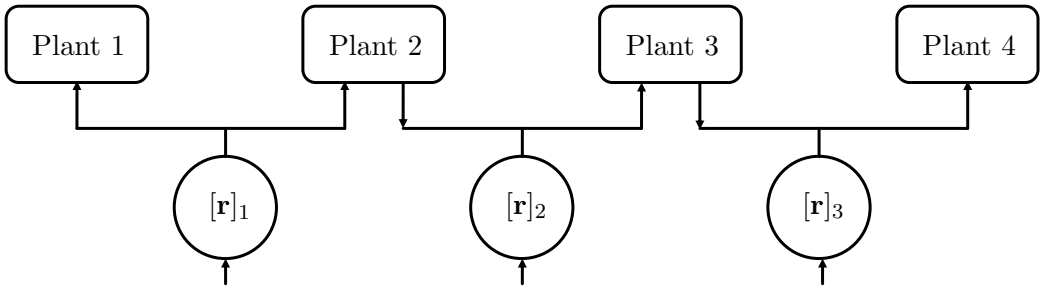


Figure 9.5: Example of a resource network with a sparsity structure.

Consider the resource network depicted in Fig. 9.5, consisting of 4 plants (subproblems) connected through 3 shared limited resources. For the sake of simplicity, each plant i includes a single decision variable and minimizes x_i^2. The system-wide optimization problem (without individual constraints) can be formulated as

$$\min_{x_1,\dots,x_4} \sum_{i=1}^{4} x_i^2, \tag{9.8a}$$

$$\text{s.t.} \sum_{i=1}^{4} \mathbf{A}_i x_i = \mathbf{b}. \tag{9.8b}$$

In the depicted examples the individual constraint matrices are

$$\mathbf{A}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \ \mathbf{A}_2 = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}, \mathbf{A}_3 = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}, \mathbf{A}_4 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{9.9}$$

which results in a sparse matrix for the system-wide constraints

$$\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3, \mathbf{A}_4] = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}. \tag{9.10}$$

The system-wide objective matrix $\mathbf{H}$ of problem (9.8) is the identity matrix. Computing the dual function of problem (9.8) results in the following Hessian:

$$\mathbf{Q} = -\mathbf{A}\mathbf{H}\mathbf{A}^T = -\mathbf{A}\mathbf{I}\mathbf{A}^T = \begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix}. \tag{9.11}$$

Note that $[\mathbf{Q}]_{1,3} = [\mathbf{Q}]_{3,1} = 0$ holds. This shows that the prices for resources 1 and 3 are uncorrelated. This is because resource 1 is only connected to plants 1 and 2 and resource 3 only to plants 3 and 4, i.e., due to the structure of the network topology.

By exploiting the sparsity structure of the network the number of required sampling points can be reduced in the case of QADA. The number of required sampling points is equal to the number of unknown parameters in the quadratic surrogate function. If dual variables are not correlated, the corresponding parameter has to be equal to 0. Therefore, for each pair of uncorrelated dual variables, one fewer sampling point is needed. This can significantly enhance the performance of the QADA algorithm for specific network structures. Note that this kind of sparse topology was used in Chapter 8 and is often encountered in machine learning applications.

9.3 Practical considerations

Wenzel [Wen20] and Maxeiner [Max21] discuss different practical and organizational barriers and issues related to dual decomposition-based distributed optimization in the context of the coordination of interconnected production systems. These issues include legal aspects regarding cross-entity and cross-company information exchange, availability of suitable models and algorithms, willingness and economic incentives to implement novel

coordination schemes, and trust in and fairness of the distributed optimization approach. The interested reader is referred to [Wen20] and [Max21] and the references therein. In the following, some related practical issues are discussed from a mathematical point of view.

Problem class

As shown throughout this thesis the problem class plays an essential role in dual decomposition-based distributed optimization. The most important property is the convexity of the system-wide optimization problem. To successfully apply distributed optimization, the coordinator must possess information on the problem classes of the individual subproblems. If all subproblems are convex, the coordinator can run a dual decomposition-based algorithm until the primal residual vanishes and the global system-wide optimum is found. However, if some or all of the subproblems are nonconvex, e.g., MIP problems, the primal residual might not vanish, even at the dual optimum. In this case, the coordinator needs to apply heuristics or perform contractions of the system-wide constraints. Hence, information on the class of the system-wide problem is required to apply dual decomposition-based distributed optimization.

Global optimality of the subproblems

Throughout this thesis, all subproblems in each benchmark problem were solved to global optimality. In the case of convex problems (QP, general convex, and DMPC benchmarks), a found locally optimal solution is also globally optimal. The considered nonconvex problems (MIQP and clustering benchmark problems) can be solved to (provable) global optimality using the branch-and-cut algorithms of commercial MIP solvers like Gurobi. Obtaining the global optimum of the subproblems is essential due to the definition of the dual function (Def. 12)

$$d(\boldsymbol{\lambda}) \coloneqq \inf_{\mathbf{x}_i \in \mathcal{X}_i, \forall i \in \mathcal{I}} \mathcal{L}(\mathbf{x}_i, \boldsymbol{\lambda}). \tag{2.28}$$

The dual function is defined as the infimum of the Lagrange function, i.e., as the largest lower bound. If some of the subproblems compute a suboptimal solution $\mathbf{x}_i^\circ$ with

$$\mathcal{L}_i(\mathbf{x}_i^\circ, \boldsymbol{\lambda}) > \mathcal{L}_i(\mathbf{x}_i^*, \boldsymbol{\lambda}), \tag{9.12}$$

the resulting computed dual value would overestimate the actual dual value

$$d(\boldsymbol{\lambda}) < \sum_{i \in \mathcal{I}} \mathcal{L}_i(\mathbf{x}_i^\circ, \boldsymbol{\lambda}), \tag{9.13}$$

resulting in incorrect values of the duality gap. This issue is especially relevant for nonconvex problems. While the dual function is still concave and proves a lower bound on the primal objective value of a nonconvex problem, it might not be possible to compute the required dual values if the subproblems are not solved to global optimality.

Fairness

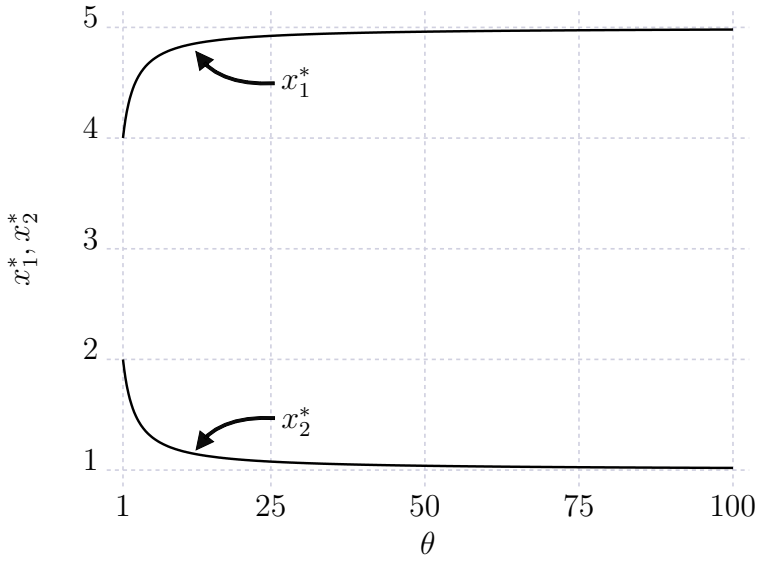


Figure 9.6: Evolution of the optimal solutions of two subsystems in (9.14) for varying weight parameters.

The objectives considered in this thesis are called utilitarian social welfare objectives [SZ19]. The underlying assumption is that all involved subproblems weight their objectives according to their actual goals, which requires a certain level of fairness. For instance, consider the optimization problem

$$\min_{x_1,x_2} \theta \cdot (x_1 - 5)^2 + (x_2 - 3)^2, \tag{9.14a}$$

$$\text{s.t. } x_1 + x_2 \leq 6, \tag{9.14b}$$

where two subsystems compete for limited resources, e.g., to satisfy their demand. The objective function of the first subsystem is weighed by the parameter θ. For $\theta = 1$ the solution to this problem is $x_1^* = 4$ and $x_2^* = 2$, which shows a compromise that both systems deviate from their target by the same amount. However, if the first subsystem artificially increases the weight parameter θ it prioritizes its demand satisfaction compared to the second subsystem. The weight does not influence the solution of a problem containing only a single objective function, apart from possible numerical issues. However, in a distributed optimization setting this would lead the coordinator to steer the price of the shared resource favorably toward the first subsystem. The resulting optimal values for increasing values of θ are depicted in Fig. 9.6. To successfully apply distributed optimization in an industrial setting a certain level of fairness has to be guaranteed, e.g., the subsystems should not artificially prioritize their objectives by increasing their weights.

9.4 Conclusion

Throughout this thesis, the potential of dual decomposition-based distributed optimization was demonstrated for a large set of benchmark problems and applications. While

the proposed algorithms showed significant performance improvements compared to other state-of-the-art algorithms, room for further improvements still exists. In addition to the numerical benchmarks, validation through real-world experiments is necessary for the adoption of distributed optimization. Industrial demonstrator platforms, like the ones provided by the *SmartFactory*$^{\mathrm{KL}}$, are well suited to conduct further research on the applicability and benefits of distributed optimization.[5] Finally, it should always be kept in mind that the underlying distributed optimization algorithms are only one part of a much larger required framework. The availability of efficient communication channels, suitable system architectures, hardware for the execution of control actions, etc., as well as previously discussed organizational issues are all of similar importance for the successful application of distributed optimization or even optimization in general. The goal of this thesis could therefore be described as adding one piece towards the completion of this puzzle and contributing to a future in which methods for the solution of large constraint-coupled problems are readily available.

[5] `https://smartfactory.de/` (last visited 25.04.2023)

Appendix

A Used software

Software	Version	Use
`Gurobi`	9.0.3	Solution of LP, QP and MIP problems
`Julia`	1.5.4	Programming language
`Matlab`	R2020a	Calling of the NAPS algorithm (implemented by Wenzel [Wen20])

Julia Package	Version	Use
`JuMP.jl`	0.21.10	Julia mathematical programming package
`Gurobi.jl`	0.9.14	Julia wrapper for the Gurobi optimizer
`Ipopt.jl`	0.7.0	Julia wrapper for the IPOPT solver
`MATLAB.jl`	0.8.2	Calling of Matlab functions
`Distributions.jl`	0.24.18	Computation of random numbers
`Gadfly.jl`	1.3.3	Plotting
`Cairo.jl`	1.0.5	Plotting
`Fontconfig.jl`	0.4.0	Plotting
`LaTeXStrings.jl`	1.2.1	Plotting
`DataFrames.jl`	1.1.1	Data storage and manipulation
`XLSX.jl`	0.7.8	Julia connection to Microsoft Excel
`JLD2.jl`	0.4.15	Exporting of benchmark problems
`FileIO.jl`	1.11.2	Reading of external files

B Benchmark problems

All benchmark problems used throughout this thesis are available on GitHub:
Generic problems:
`https://github.com/VaYf/EJCOMP_Benchmark_Problems`
DMPC problems:
`https://github.com/VaYf/DMPC-Benchmark-Problems`
Clustering problems:
`https://github.com/VaYf/Clustering-Benchmark-Problems`

C Summaries of computational results

C.1 Distributed QP problems

Table C.1: Results for the distributed optimization of the QP benchmark problems (mean values of the converged instances only), $\bar{t}$: mean number of iterations until convergence, $\overline{\|\mathbf{w}_p\|_2}$: mean primal residual of converged runs ($\times 10^{-3}$), $\overline{T_{\text{comp}}}$: mean computation time of converged runs (in s), $\%_c$: percentage of converged runs within $t_{\max}$ iterations.

	SG				ADMM			
QP	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
Mean	**384.74**	**9.88**	**308.36**	**16.83**	**179.55**	**6.81**	**145.54**	**82.3**
$(2,2)$	488.0	9.96	390.71	2	19.88	6.6	15.92	100
$(4,2)$	435.6	9.93	348.76	10	32.22	5.93	25.81	100
$(4,3)$	–	–	–	0	44.62	5.73	35.75	100
$(4,4)$	–	–	–	0	49.22	6.14	39.43	100
$(8,2)$	335.67	9.93	268.76	6	37.32	6.25	29.9	100
$(8,3)$	344.0	9.98	275.45	2	57.84	6.28	46.33	100
$(8,4)$	–	–	–	0	69.96	6.41	56.05	100
$(8,5)$	–	–	–	0	80.42	5.29	64.43	100
$(8,6)$	–	–	–	0	97.48	5.58	78.14	100
$(8,7)$	–	–	–	0	117.02	5.25	93.83	100
$(8,8)$	–	–	–	0	143.42	4.34	115.02	100
$(16,2)$	307.14	9.85	245.96	14	45.26	7.55	36.26	100
$(16,3)$	–	–	–	0	57.78	8.22	46.29	100
$(16,4)$	–	–	–	0	76.16	8.61	61.02	100
$(16,5)$	–	–	–	0	90.02	8.22	72.14	100
$(16,6)$	–	–	–	0	112.88	7.09	90.52	100
$(16,7)$	–	–	–	0	130.1	7.56	104.37	100
$(16,8)$	–	–	–	0	132.5	7.18	106.31	100
$(16,9)$	–	–	–	0	162.78	6.66	130.64	100
$(16,10)$	–	–	–	0	169.2	6.38	135.8	100
$(32,2)$	407.57	9.9	326.43	14	62.68	7.98	50.24	100
$(32,3)$	384.0	9.74	307.54	4	68.76	8.68	55.13	100
$(32,4)$	–	–	–	0	70.06	8.92	56.15	100
$(32,5)$	–	–	–	0	90.48	9.05	72.55	100
$(32,6)$	–	–	–	0	117.22	9.12	94.06	100
$(32,7)$	–	–	–	0	119.48	9.05	95.91	100
$(32,8)$	–	–	–	0	140.3	8.96	112.66	100

Continued on next page

	SG				ADMM			
QP	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
$(32, 9)$	–	–	–	0	158.14	9.28	127.01	100
$(32, 10)$	–	–	–	0	175.53	9.09	141.04	98
$(64, 2)$	269.73	9.81	216.11	30	119.82	7.55	96.14	100
$(64, 3)$	401.6	9.91	321.79	10	167.74	7.27	134.71	100
$(64, 4)$	232.0	9.94	185.91	2	154.4	7.12	124.04	96
$(64, 5)$	462.0	9.9	370.21	2	180.18	7.74	144.85	100
$(64, 6)$	–	–	–	0	184.04	7.15	148.13	98
$(64, 7)$	–	–	–	0	160.06	8.19	128.85	100
$(64, 8)$	–	–	–	0	173.29	8.74	139.54	98
$(64, 9)$	–	–	–	0	154.56	9.21	124.49	100
$(64, 10)$	–	–	–	0	165.35	9.76	133.2	98
$(128, 2)$	336.5	9.74	269.66	60	195.11	6.21	156.95	94
$(128, 3)$	357.54	9.86	286.51	26	265.8	5.87	214.33	82
$(128, 4)$	396.5	9.89	317.75	16	297.69	6.26	240.2	78
$(128, 5)$	483.0	9.85	387.09	2	329.59	5.98	266.25	64
$(128, 6)$	–	–	–	0	363.05	4.86	293.65	42
$(128, 7)$	430.5	9.91	345.11	4	388.74	4.2	314.79	38
$(128, 8)$	–	–	–	0	393.42	5.29	318.96	24
$(128, 9)$	–	–	–	0	379.73	5.41	308.82	22
$(128, 10)$	–	–	–	0	347.78	5.51	283.17	18
$(256, 2)$	264.78	9.75	212.3	72	285.66	4.99	233.81	76
$(256, 3)$	346.23	9.87	277.61	52	309.54	4.76	254.01	56
$(256, 4)$	367.05	9.87	294.31	40	332.08	4.94	272.93	24
$(256, 5)$	413.67	9.9	331.73	18	365.67	4.16	301.43	24
$(256, 6)$	444.2	9.9	356.26	10	449.0	6.89	370.87	8
$(256, 7)$	356.5	9.92	285.94	4	446.5	4.47	369.52	4
$(256, 8)$	487.0	9.83	390.68	2	358.0	4.02	296.67	2
$(256, 9)$	483.0	9.93	388.06	2				0
$(256, 10)$	–	–	–	0	–	–	–	0
	BTM				QAC			
QP	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
Mean	**196.95**	**7.47**	**160.07**	**95.11**	**199.24**	**2.15**	**167.73**	**57.32**
$(2, 2)$	90.24	5.68	72.7	100	71.59	1.7	59.78	98
$(4, 2)$	79.78	5.78	64.28	100	59.35	0.12	49.52	98
$(4, 3)$	146.82	6.52	118.31	100	133.77	0.61	111.15	88
$(4, 4)$	177.52	6.85	143.1	100	172.55	0.73	142.92	84

Continued on next page

	BTM				QAC			
QP	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
$(8,2)$	65.1	6.33	52.45	100	61.18	0.13	50.89	100
$(8,3)$	112.74	7.07	90.85	100	114.68	0.59	95.07	88
$(8,4)$	167.06	6.99	134.69	100	198.28	0.57	164.05	72
$(8,5)$	185.76	7.56	149.77	100	187.36	0.3	154.64	50
$(8,6)$	239.9	7.45	193.63	98	232.47	1.18	191.63	38
$(8,7)$	268.96	7.47	217.47	94	267.89	1.12	220.67	18
$(8,8)$	320.42	7.46	260.22	86	283.17	1.59	233.67	12
$(16,2)$	70.36	6.04	56.7	100	48.09	0.61	39.89	92
$(16,3)$	103.48	6.96	83.39	100	90.12	0.18	74.5	82
$(16,4)$	140.16	6.74	112.99	100	173.22	0.53	142.89	64
$(16,5)$	177.34	8.02	143.0	100	212.15	1.27	174.71	54
$(16,6)$	233.92	7.69	188.8	100	300.0	1.21	246.79	24
$(16,7)$	246.31	7.81	199.12	96	262.0	4.5	215.46	8
$(16,8)$	303.51	7.77	246.35	94	329.0	4.63	271.44	4
$(16,9)$	356.37	7.77	291.76	92	412.0	0.0	342.64	2
$(16,10)$	383.29	7.95	319.84	82	–	–	–	0
$(32,2)$	65.5	6.16	52.8	100	42.54	0.21	35.13	92
$(32,3)$	96.04	7.12	77.4	100	84.18	0.63	69.35	80
$(32,4)$	130.08	7.12	104.88	100	143.88	1.45	118.33	82
$(32,5)$	166.04	7.68	133.9	100	248.36	1.41	204.51	56
$(32,6)$	207.43	8.06	167.42	98	253.23	0.85	208.03	26
$(32,7)$	237.71	7.7	192.17	96	412.5	3.98	339.07	8
$(32,8)$	291.64	7.82	236.7	100	244.0	–	201.09	2
$(32,9)$	328.81	8.04	268.97	94	–	–	–	0
$(32,10)$	352.24	8.09	292.34	92	–	–	–	0
$(64,2)$	55.82	6.16	45.01	100	45.31	0.59	41.39	96
$(64,3)$	80.36	6.73	64.82	100	85.67	1.21	77.27	90
$(64,4)$	116.08	7.37	93.65	98	139.49	2.05	124.65	82
$(64,5)$	155.72	7.76	125.68	100	220.75	1.68	197.5	48
$(64,6)$	204.92	7.63	165.55	100	359.5	4.08	320.61	32
$(64,7)$	225.84	7.91	182.7	98	429.0	5.58	381.52	8
$(64,8)$	268.52	8.05	217.97	92	–	–	–	0
$(64,9)$	297.57	8.19	243.03	98	–	–	–	0
$(64,10)$	327.05	7.87	270.27	88	–	–	–	0
$(128,2)$	47.18	6.18	38.05	100	33.55	0.91	27.63	98
$(128,3)$	71.42	7.06	57.59	100	74.86	1.84	61.54	100

Continued on next page

	BTM				QAC			
QP	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
$(128, 4)$	107.2	7.74	86.48	100	144.81	3.16	119.0	84
$(128, 5)$	150.42	7.74	121.39	100	255.65	3.19	209.99	74
$(128, 6)$	184.9	7.95	149.3	100	317.08	3.64	260.34	26
$(128, 7)$	228.14	8.17	184.49	98	447.43	6.88	367.6	14
$(128, 8)$	270.06	8.21	219.05	94	–	–	–	0
$(128, 9)$	302.5	8.15	246.89	84	–	–	–	0
$(128, 10)$	328.77	8.44	271.64	86	–	–	–	0
$(256, 2)$	47.62	7.05	38.44	100	30.66	1.65	27.41	94
$(256, 3)$	79.18	6.99	63.92	100	65.47	3.01	53.86	68
$(256, 4)$	115.52	7.21	93.27	96	136.28	3.63	120.78	80
$(256, 5)$	161.58	8.1	130.5	96	214.71	4.14	189.67	68
$(256, 6)$	213.37	8.17	172.5	92	374.73	5.77	330.63	30
$(256, 7)$	275.6	8.2	223.0	90	354.0	7.29	311.04	8
$(256, 8)$	311.61	8.13	252.73	76	–	–	–	0
$(256, 9)$	317.19	8.61	258.52	62	–	–	–	0
$(256, 10)$	340.5	8.78	281.28	56	–	–	–	0
	QADA-SG				QADA-BTM			
QP	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
Mean	**133.49**	**3.31**	**139.5**	**96.46**	**128.10**	**3.26**	**135.22**	**96.96**
$(2, 2)$	92.1	1.48	94.36	98	79.29	1.16	81.5	98
$(4, 2)$	62.96	0.83	63.07	98	54.2	0.69	54.53	100
$(4, 3)$	98.19	0.84	94.21	96	83.78	1.0	80.78	100
$(4, 4)$	103.83	0.84	98.74	96	99.24	0.65	94.21	98
$(8, 2)$	55.68	0.34	54.11	100	50.18	0.75	49.0	100
$(8, 3)$	70.14	0.77	66.6	98	74.5	0.97	71.98	96
$(8, 4)$	102.63	2.0	96.3	98	90.65	1.52	85.24	96
$(8, 5)$	107.82	1.72	99.41	100	101.38	1.58	94.36	100
$(8, 6)$	135.53	2.73	126.11	98	126.63	3.07	118.89	98
$(8, 7)$	178.96	3.0	172.32	96	166.5	2.12	161.86	96
$(8, 8)$	217.6	2.75	219.73	94	204.39	2.61	207.13	92
$(16, 2)$	46.83	0.29	44.31	96	38.9	0.98	36.99	98
$(16, 3)$	72.41	1.4	68.11	98	77.4	1.04	73.76	100
$(16, 4)$	90.72	1.85	84.46	100	87.06	2.26	80.52	100
$(16, 5)$	106.08	3.24	96.94	100	99.9	2.19	92.27	100
$(16, 6)$	147.48	3.12	137.22	100	140.04	3.15	130.28	100
$(16, 7)$	181.55	4.18	174.51	94	169.7	3.15	163.62	92

Continued on next page

	QADA-SG				QADA-BTM			
QP	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
$(16, 8)$	203.02	4.0	203.03	92	191.09	4.52	189.8	92
$(16, 9)$	266.65	4.23	283.6	98	254.27	3.48	272.58	96
$(16, 10)$	298.77	4.33	347.32	86	287.04	4.55	331.77	90
$(32, 2)$	46.56	0.91	43.19	100	38.42	0.82	35.46	100
$(32, 3)$	62.56	1.89	58.2	100	57.86	1.16	54.2	98
$(32, 4)$	75.53	2.66	69.08	98	68.57	3.03	62.98	98
$(32, 5)$	118.35	4.32	107.45	96	105.58	3.75	96.52	96
$(32, 6)$	138.91	3.02	129.05	92	133.7	3.56	123.52	94
$(32, 7)$	169.02	3.28	160.11	96	164.98	3.43	158.62	96
$(32, 8)$	198.66	3.84	198.02	100	200.6	3.12	198.95	100
$(32, 9)$	241.02	3.83	255.18	92	229.2	4.03	243.75	90
$(32, 10)$	270.86	4.87	307.4	88	263.07	4.28	303.17	92
$(64, 2)$	31.62	1.02	43.44	100	33.58	1.55	44.82	100
$(64, 3)$	49.58	1.94	71.2	100	51.44	1.0	73.93	100
$(64, 4)$	69.34	3.83	93.72	94	68.62	3.43	94.95	96
$(64, 5)$	110.73	3.04	144.17	98	101.2	3.72	135.79	98
$(64, 6)$	130.98	4.59	174.69	98	121.0	4.6	160.05	98
$(64, 7)$	160.73	4.85	221.55	96	150.56	4.47	212.24	100
$(64, 8)$	187.64	4.01	246.8	90	183.59	4.49	257.16	92
$(64, 9)$	211.48	4.66	223.92	96	207.84	5.44	223.83	98
$(64, 10)$	248.17	5.04	285.49	96	243.56	5.43	282.29	96
$(128, 2)$	29.16	1.02	26.11	100	25.9	1.75	23.48	100
$(128, 3)$	56.67	1.64	52.49	98	40.96	1.77	38.04	98
$(128, 4)$	67.94	4.62	60.51	98	68.24	3.34	62.19	100
$(128, 5)$	91.12	4.59	81.62	100	88.96	4.44	80.45	100
$(128, 6)$	120.16	4.17	109.3	98	118.29	5.03	109.91	98
$(128, 7)$	155.96	5.05	150.04	94	151.77	6.01	146.24	96
$(128, 8)$	176.44	5.64	174.09	96	180.35	5.16	180.84	98
$(128, 9)$	216.81	5.78	229.86	96	216.63	5.68	232.67	98
$(128, 10)$	258.61	4.93	292.52	92	254.94	4.75	294.96	94
$(256, 2)$	24.26	2.26	31.32	100	24.92	2.11	31.03	100
$(256, 3)$	44.76	3.11	40.82	98	40.41	3.26	37.57	98
$(256, 4)$	64.12	4.62	57.22	100	63.68	5.26	57.08	100
$(256, 5)$	89.23	4.97	80.8	96	86.22	4.88	77.67	98
$(256, 6)$	111.79	5.25	104.01	94	109.28	4.76	102.81	92
$(256, 7)$	136.79	6.07	130.12	94	137.23	5.16	131.69	94

Continued on next page

	QADA-SG				QADA-BTM			
QP	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
$(256, 8)$	181.6	5.34	184.46	96	181.83	5.65	184.45	96
$(256, 9)$	223.89	5.85	240.29	94	228.23	6.11	248.18	94
$(256, 10)$	265.65	5.18	309.53	92	256.17	4.45	299.57	92
	QADA-QNDA				QNDA			
QP	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
Mean	**127.23**	**3.29**	**135.47**	**96.57**	**134.31**	**3.94**	**126.4**	**98.5**
$(2, 2)$	73.71	1.32	76.14	98	114.79	1.81	110.15	96
$(4, 2)$	58.92	0.88	59.72	100	107.0	1.24	98.03	100
$(4, 3)$	88.22	1.13	85.64	100	162.68	3.05	147.07	88
$(4, 4)$	99.66	0.85	93.68	100	183.33	2.89	161.29	98
$(8, 2)$	48.82	0.6	48.29	100	67.62	2.03	58.86	100
$(8, 3)$	66.49	0.58	63.23	98	113.92	3.72	98.82	96
$(8, 4)$	86.49	1.38	81.01	98	162.72	3.45	143.16	100
$(8, 5)$	105.98	1.52	99.51	98	158.6	3.8	138.52	100
$(8, 6)$	125.84	2.09	118.4	98	196.5	4.65	174.04	100
$(8, 7)$	171.27	2.07	165.87	98	209.17	4.44	184.88	96
$(8, 8)$	208.91	3.68	210.65	92	249.43	5.49	224.42	94
$(16, 2)$	52.8	1.23	49.9	98	55.67	2.3	47.84	98
$(16, 3)$	72.73	1.08	69.9	98	93.02	3.35	80.1	100
$(16, 4)$	81.64	1.69	75.68	100	118.66	4.09	102.81	100
$(16, 5)$	102.96	3.26	95.29	100	139.9	4.25	121.31	100
$(16, 6)$	135.53	3.31	126.33	98	184.63	4.34	161.44	98
$(16, 7)$	168.29	3.9	164.59	90	175.94	5.46	157.71	98
$(16, 8)$	193.8	3.81	194.65	92	215.06	4.65	194.23	100
$(16, 9)$	248.6	4.25	266.73	96	241.82	5.34	224.1	100
$(16, 10)$	273.55	5.2	317.29	84	273.83	4.82	259.13	92
$(32, 2)$	47.54	0.87	44.06	100	56.96	2.8	49.13	100
$(32, 3)$	63.46	1.53	59.4	100	84.42	3.46	73.24	100
$(32, 4)$	73.53	1.82	67.76	98	104.52	2.86	90.92	100
$(32, 5)$	103.4	3.13	94.43	96	124.96	4.14	108.99	98
$(32, 6)$	127.04	2.97	117.49	92	139.69	4.78	123.51	98
$(32, 7)$	162.69	3.17	156.72	96	165.2	4.27	147.07	98
$(32, 8)$	194.74	3.45	194.06	100	200.04	4.73	181.42	100
$(32, 9)$	225.04	3.84	240.13	90	206.24	4.2	191.17	98
$(32, 10)$	260.49	5.3	302.28	94	217.28	5.3	206.8	100
$(64, 2)$	31.08	1.16	42.06	100	51.02	2.25	60.07	100

Continued on next page

	QADA-QNDA				QNDA			
QP	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
$(64, 3)$	50.65	1.79	78.23	98	63.92	2.81	78.38	100
$(64, 4)$	71.19	2.61	101.98	96	87.51	3.97	105.08	98
$(64, 5)$	98.82	4.44	131.8	98	106.36	4.09	129.2	100
$(64, 6)$	116.65	4.06	156.17	98	126.41	3.71	154.4	98
$(64, 7)$	157.96	4.76	231.93	98	141.68	4.6	178.34	100
$(64, 8)$	180.93	4.5	249.15	90	165.38	5.51	197.14	94
$(64, 9)$	206.29	5.01	223.49	98	176.47	5.31	164.97	98
$(64, 10)$	240.42	4.63	279.79	96	193.42	4.65	183.88	100
$(128, 2)$	28.78	1.76	26.24	100	36.2	2.12	31.54	100
$(128, 3)$	44.4	2.08	41.81	100	53.04	2.8	46.94	100
$(128, 4)$	64.8	3.81	59.89	98	72.96	3.84	64.4	100
$(128, 5)$	88.72	4.96	81.46	100	101.54	4.19	90.49	100
$(128, 6)$	107.71	4.67	100.64	98	104.16	4.8	93.86	100
$(128, 7)$	137.64	5.93	133.35	90	128.54	4.89	116.57	100
$(128, 8)$	179.18	5.22	181.73	98	148.16	4.74	136.43	100
$(128, 9)$	214.43	4.8	232.81	98	159.65	4.91	151.1	98
$(128, 10)$	251.77	5.29	294.71	94	179.27	4.34	172.01	98
$(256, 2)$	23.52	2.44	30.01	100	33.42	2.4	40.75	100
$(256, 3)$	51.31	3.92	49.09	98	50.52	3.11	45.01	100
$(256, 4)$	72.76	4.52	67.19	100	69.56	4.33	63.65	100
$(256, 5)$	86.71	4.95	79.88	98	84.71	4.14	77.23	98
$(256, 6)$	107.24	4.51	102.95	92	98.55	3.94	90.51	98
$(256, 7)$	138.79	6.7	138.27	94	119.44	4.14	111.61	96
$(256, 8)$	172.15	4.65	179.9	92	138.16	4.21	131.36	98
$(256, 9)$	218.28	5.79	238.54	92	148.18	4.9	145.65	100
$(256, 10)$	260.3	5.52	314.51	92	159.38	4.22	157.88	96

C.2 Distributed MIQP problems

Table C.2: Results for the distributed optimization of the MIQP benchmark problems (mean values of the converged instances only), $\bar{t}$: mean number of iterations until convergence, $\overline{\text{rel. DG}}$: mean relative duality gap of converged runs (in %) , $\overline{T_{\text{comp}}}$: mean computation time of converged runs (in s), $\%_c$: percentage of converged runs within $t_{\max}$ iterations.

	SG				ADMM			
MIQP	$\bar{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$	$\%_c$
Mean	**86.69**	**1.66**	**69.88**	**99.5**	**25.06**	**2.22**	**21.16**	**100**
$(100, 2)$	58.1	0.78	46.73	100	23.4	1.11	18.95	100
$(100, 3)$	73.4	1.65	59.11	100	21.2	1.69	17.15	100
$(100, 4)$	60.7	4.81	48.85	100	23.5	5.2	19.24	100
$(100, 5)$	51.6	9.76	41.61	100	22.7	10.09	18.43	100
$(200, 2)$	113.33	0.35	90.94	90	25.6	1.08	20.8	100
$(200, 3)$	66.6	0.84	53.5	100	24.5	1.06	20.07	100
$(200, 4)$	146.5	2.9	118.08	100	23.7	3.91	19.66	100
$(200, 5)$	65.1	2.97	52.54	100	23.2	3.61	19.11	100
$(300, 2)$	129.9	0.31	104.55	100	27.0	0.88	22.31	100
$(300, 3)$	66.2	0.58	53.34	100	25.4	0.79	21.19	100
$(300, 4)$	71.2	1.25	57.54	100	24.3	1.49	20.39	100
$(300, 5)$	61.1	1.56	49.37	100	24.7	1.82	20.82	100
$(400, 2)$	89.4	0.13	72.04	100	28.4	0.59	23.93	100
$(400, 3)$	165.2	0.76	133.14	100	26.9	2.26	23.16	100
$(400, 4)$	77.0	1.06	62.28	100	25.4	1.99	21.89	100
$(400, 5)$	54.3	1.17	43.94	100	24.2	1.63	21.03	100
$(500, 2)$	170.3	0.16	137.18	100	29.9	1.17	25.84	100
$(500, 3)$	58.9	0.28	47.55	100	26.3	0.57	23.37	100
$(500, 4)$	76.4	0.63	61.81	100	25.6	1.31	22.85	100
$(500, 5)$	78.5	1.21	63.56	100	25.4	2.08	22.97	100
	BTM				QAC			
MIQP	$\bar{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$	$\%_c$
Mean	**80.52**	**1.66**	**65.41**	**100**	**59.4**	**2.13**	**52.83**	**86.0**
$(100, 2)$	29.3	0.81	23.72	100	63.0	0.03	56.64	20
$(100, 3)$	66.1	1.67	53.59	100	81.0	1.66	71.81	60
$(100, 4)$	98.7	4.71	80.04	100	82.14	4.49	74.82	70
$(100, 5)$	124.0	9.64	100.67	100	55.6	11.16	47.98	100
$(200, 2)$	57.6	0.5	46.54	100	101.17	0.44	92.76	60

Continued on next page

	BTM				QAC			
MIQP	$\bar{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$	$\%_c$
$(200, 3)$	57.6	0.86	46.56	100	95.5	0.92	85.42	100
$(200, 4)$	157.5	2.98	127.89	100	59.22	4.91	52.42	90
$(200, 5)$	124.6	2.94	101.29	100	42.2	3.29	36.12	100
$(300, 2)$	44.7	0.32	36.24	100	89.2	0.37	80.62	100
$(300, 3)$	51.9	0.58	42.15	100	48.6	0.74	42.99	100
$(300, 4)$	86.0	1.23	69.97	100	34.89	2.26	29.85	90
$(300, 5)$	110.8	1.53	90.18	100	35.5	1.61	29.91	100
$(400, 2)$	34.7	0.16	28.15	100	65.0	0.2	59.38	80
$(400, 3)$	99.7	0.76	80.91	100	30.57	2.98	26.28	70
$(400, 4)$	84.2	1.06	68.55	100	27.6	1.81	23.37	100
$(400, 5)$	93.9	1.15	76.49	100	39.8	1.57	33.7	100
$(500, 2)$	55.2	0.18	44.75	100	51.33	0.43	46.84	90
$(500, 3)$	42.6	0.28	34.56	100	79.4	0.36	71.45	100
$(500, 4)$	75.4	0.63	61.44	100	30.3	1.6	25.99	100
$(500, 5)$	115.9	1.21	94.52	100	76.0	1.71	68.26	90

	QADA-SG				QADA-BTM			
MIQP	$\bar{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$	$\%_c$
Mean	**19.37**	**2.54**	**18.37**	**100**	**20.78**	**3.53**	**18.62**	**100**
$(100, 2)$	23.4	1.59	34.3	100	17.6	0.99	26.82	100
$(100, 3)$	23.4	2.1	32.31	100	17.4	2.8	18.07	100
$(100, 4)$	19.8	5.09	17.21	100	26.1	5.58	22.19	100
$(100, 5)$	22.7	13.42	19.53	100	32.1	27.97	27.19	100
$(200, 2)$	15.2	1.01	17.43	100	12.5	0.83	11.05	100
$(200, 3)$	18.3	1.46	17.62	100	16.9	1.03	15.03	100
$(200, 4)$	23.3	4.56	19.73	100	27.0	7.36	23.07	100
$(200, 5)$	23.5	6.08	19.94	100	29.4	3.79	24.91	100
$(300, 2)$	15.6	0.4	14.61	100	10.5	0.38	9.22	100
$(300, 3)$	17.3	0.92	16.6	100	15.7	0.85	14.18	100
$(300, 4)$	18.8	2.04	15.88	100	22.1	2.55	18.53	100
$(300, 5)$	22.9	2.04	19.14	100	28.9	2.85	24.28	100
$(400, 2)$	12.5	0.19	10.87	100	12.4	0.18	12.49	100
$(400, 3)$	19.4	2.44	16.9	100	18.6	1.64	15.97	100
$(400, 4)$	18.9	1.29	15.97	100	23.3	1.66	19.6	100
$(400, 5)$	23.0	1.45	19.08	100	29.5	5.01	24.96	100
$(500, 2)$	13.0	0.56	11.32	100	12.2	0.29	10.64	100
$(500, 3)$	16.7	0.49	15.42	100	15.0	0.5	13.3	100

Continued on next page

	QADA-SG				QADA-BTM			
MIQP	$\bar{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$	$\%_c$
$(500, 4)$	16.6	1.44	13.93	100	19.2	1.53	16.23	100
$(500, 5)$	23.1	2.14	19.67	100	29.1	2.78	24.57	100
	QADA-QNDA				QNDA			
MIQP	$\bar{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$	$\%_c$
Mean	**22.2**	**2.86**	**22.76**	**100**	**79.9**	**1.73**	**74.91**	**100**
$(100, 2)$	43.7	0.95	77.42	100	29.4	0.78	26.3	100
$(100, 3)$	23.8	1.82	29.72	100	65.5	1.65	60.62	100
$(100, 4)$	30.9	5.94	30.3	100	100.4	4.92	91.62	100
$(100, 5)$	29.6	19.24	26.36	100	122.0	9.1	115.9	100
$(200, 2)$	13.0	0.95	12.22	100	58.2	0.5	52.43	100
$(200, 3)$	16.1	1.02	15.2	100	57.7	0.88	52.75	100
$(200, 4)$	27.9	6.39	26.16	100	152.3	4.67	143.7	100
$(200, 5)$	29.4	4.59	26.34	100	122.9	2.98	116.56	100
$(300, 2)$	10.3	0.33	9.17	100	45.0	0.32	40.4	100
$(300, 3)$	16.9	1.02	15.91	100	51.4	0.57	47.45	100
$(300, 4)$	21.0	2.39	18.86	100	84.7	1.21	79.91	100
$(300, 5)$	28.5	2.4	25.83	100	109.4	1.55	103.82	100
$(400, 2)$	11.9	0.21	11.88	100	35.1	0.15	31.74	100
$(400, 3)$	19.0	1.67	17.62	100	99.4	0.76	92.27	100
$(400, 4)$	22.8	2.01	20.92	100	84.0	1.06	79.31	100
$(400, 5)$	27.1	1.43	24.91	100	92.8	1.14	89.43	100
$(500, 2)$	12.5	0.38	11.54	100	55.8	0.19	50.58	100
$(500, 3)$	14.9	0.51	14.15	100	42.8	0.28	39.82	100
$(500, 4)$	18.4	1.26	16.71	100	74.2	0.63	71.13	100
$(500, 5)$	26.3	2.73	23.92	100	115.1	1.22	112.45	100

C.3 Distributed convex problems

Table C.3: Results for the distributed optimization of the convex benchmark problems (mean values of the converged instances only), $\bar{t}$: mean number of iterations until convergence, $\overline{\|\mathbf{w}_p\|_2}$: mean primal residual of converged runs ($\times 10^{-3}$), $\overline{T_{\text{comp}}}$: mean computation time of converged runs (in s), $\%_c$: percentage of converged runs within $t_{\max}$ iterations.

	SG				ADMM			
Conv	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
Mean	**342.51**	**9.72**	**310.45**	**46.07**	**117.44**	**7.44**	**104.97**	**93.0**
$(2,2)$	–	–	–	0	27.2	8.2	22.26	100
$(4,2)$	–	–	–	0	25.8	6.92	21.31	100
$(4,3)$	–	–	–	0	37.6	7.04	31.05	100
$(4,4)$	–	–	–	0	38.6	7.34	31.69	100
$(8,2)$	–	–	–	0	42.8	7.15	35.25	100
$(8,3)$	–	–	–	0	51.7	8.39	42.51	100
$(8,4)$	–	–	–	0	50.1	7.67	41.44	100
$(8,5)$	–	–	–	0	55.0	5.93	45.93	100
$(8,6)$	–	–	–	0	64.0	6.26	53.24	100
$(8,7)$	–	–	–	0	83.8	5.54	69.81	100
$(8,8)$	–	–	–	0	96.6	6.59	80.13	100
$(16,2)$	464.0	9.92	381.03	10	35.4	7.66	29.7	100
$(16,3)$	345.0	9.87	287.14	10	49.6	8.25	41.39	100
$(16,4)$	–	–	–	0	66.5	7.91	55.52	100
$(16,5)$	–	–	–	0	54.5	7.68	45.43	100
$(16,6)$	–	–	–	0	67.8	7.55	56.55	100
$(16,7)$	–	–	–	0	72.2	7.2	60.33	100
$(16,8)$	–	–	–	0	77.5	7.73	65.06	100
$(16,9)$	–	–	–	0	83.0	7.55	69.74	100
$(16,10)$	–	–	–	0	107.5	7.17	90.01	100
$(32,2)$	229.0	9.57	191.38	20	71.9	7.91	60.72	100
$(32,3)$	429.5	9.94	361.66	20	51.4	8.22	43.72	100
$(32,4)$	432.0	9.91	364.56	10	54.8	8.7	46.59	100
$(32,5)$	–	–	–	0	50.4	8.16	42.87	100
$(32,6)$	–	–	–	0	55.4	8.48	47.0	100
$(32,7)$	–	–	–	0	69.7	8.5	59.0	100
$(32,8)$	–	–	–	0	94.9	8.7	82.04	100
$(32,9)$	–	–	–	0	88.22	8.15	74.93	90

Continued on next page

	SG				ADMM			
Conv	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
$(32, 10)$	469.0	9.51	395.54	10	98.8	7.62	84.84	100
$(64, 2)$	332.0	9.92	281.45	20	88.0	6.59	75.52	100
$(64, 3)$	361.0	9.93	308.74	20	101.9	6.81	89.45	100
$(64, 4)$	332.0	9.92	285.71	20	62.0	8.0	53.59	100
$(64, 5)$	–	–	–	0	98.2	7.56	84.45	100
$(64, 6)$	–	–	–	0	69.3	8.48	62.24	100
$(64, 7)$	411.5	9.93	353.78	20	94.5	8.53	83.29	100
$(64, 8)$	–	–	–	0	81.78	8.0	70.77	90
$(64, 9)$	–	–	–	0	106.22	8.16	91.6	90
$(64, 10)$	–	–	–	0	135.71	6.38	118.44	70
$(128, 2)$	268.5	9.84	238.85	60	172.89	5.99	153.89	90
$(128, 3)$	309.2	9.74	277.81	50	188.2	5.16	170.22	100
$(128, 4)$	268.0	8.92	252.23	30	200.1	6.92	184.32	100
$(128, 5)$	289.0	9.81	259.23	60	219.6	6.81	200.39	100
$(128, 6)$	409.0	9.94	360.74	20	238.4	7.7	211.37	100
$(128, 7)$	340.0	9.67	317.57	40	291.0	7.92	272.24	80
$(128, 8)$	323.2	9.71	289.48	50	283.88	8.26	259.66	80
$(128, 9)$	375.5	9.86	335.07	40	278.71	7.62	255.45	70
$(128, 10)$	429.0	9.4	397.06	40	185.86	8.79	167.21	70
$(256, 2)$	175.29	9.66	169.3	70	258.25	3.87	249.57	40
$(256, 3)$	243.38	9.63	238.09	80	441.57	6.63	427.84	70
$(256, 4)$	306.86	9.69	312.87	70	–	–	–	0
$(256, 5)$	256.7	9.46	257.95	100	453.0	7.61	437.14	10
$(256, 6)$	346.0	9.71	339.83	90	–	–	–	0
$(256, 7)$	339.75	9.73	332.69	80	–	–	–	0
$(256, 8)$	348.1	9.69	345.85	100	–	–	–	0
$(256, 9)$	358.62	9.7	362.9	80	–	–	–	0
$(256, 10)$	399.29	9.51	393.96	70	–	–	–	0
	BTM				QAC			
Conv	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
Mean	**160.29**	**7.8**	**139.29**	**78.26**	**207.16**	**6.55**	**186.9**	**81.52**
$(2, 2)$	128.2	6.61	105.05	100	134.0	4.87	114.82	50
$(4, 2)$	98.0	7.62	86.08	100	84.3	4.91	73.94	100
$(4, 3)$	107.44	7.25	89.48	90	146.0	6.12	127.36	90
$(4, 4)$	148.1	6.89	123.98	100	214.8	4.36	186.9	100
$(8, 2)$	74.0	5.83	61.39	80	89.88	4.42	78.25	80

Continued on next page

	BTM				QAC			
Conv	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
$(8,3)$	99.4	5.88	82.6	100	117.1	3.78	101.48	100
$(8,4)$	108.6	7.45	90.65	100	157.2	5.31	136.01	100
$(8,5)$	103.9	7.66	86.58	100	154.6	5.89	133.19	100
$(8,6)$	145.6	6.87	121.68	100	300.0	6.57	258.93	70
$(8,7)$	199.5	8.29	167.84	100	395.4	6.9	342.82	50
$(8,8)$	220.8	8.37	185.82	100	333.4	6.88	286.44	50
$(16,2)$	46.9	4.8	39.15	100	64.0	6.74	55.7	90
$(16,3)$	88.7	5.79	74.66	100	138.1	5.51	121.15	100
$(16,4)$	109.7	7.73	92.0	100	156.33	6.33	135.76	90
$(16,5)$	110.9	7.96	92.33	100	178.1	8.52	153.3	100
$(16,6)$	172.5	8.06	144.77	100	235.78	8.02	203.73	90
$(16,7)$	191.67	8.07	161.48	90	354.75	8.14	307.39	80
$(16,8)$	209.3	8.34	176.79	100	378.75	8.16	328.46	40
$(16,9)$	225.88	8.97	191.94	80	439.5	7.08	382.58	20
$(16,10)$	333.67	9.13	286.26	60	–	–	–	0
$(32,2)$	56.6	6.08	47.84	100	55.88	3.54	49.33	80
$(32,3)$	62.9	5.82	53.21	100	73.6	6.85	65.25	100
$(32,4)$	84.5	8.2	71.58	100	90.5	5.96	80.05	100
$(32,5)$	138.3	8.1	117.08	100	145.0	6.9	127.45	90
$(32,6)$	148.11	8.29	125.71	90	222.3	7.22	194.63	100
$(32,7)$	239.83	8.33	203.34	60	344.56	7.17	300.29	90
$(32,8)$	280.0	8.85	239.04	60	433.67	8.96	378.71	30
$(32,9)$	344.0	8.89	293.6	50	458.0	7.81	406.43	10
$(32,10)$	331.5	9.43	284.68	20	–	–	–	0
$(64,2)$	68.5	6.61	58.72	100	46.3	3.27	41.35	100
$(64,3)$	106.7	8.31	93.44	100	64.0	5.64	57.8	90
$(64,4)$	101.56	7.6	87.82	90	86.5	7.18	77.84	100
$(64,5)$	138.0	8.36	119.09	80	190.5	6.76	171.24	100
$(64,6)$	153.0	8.69	131.54	20	208.6	8.35	191.2	100
$(64,7)$	231.5	8.77	201.97	40	371.2	9.12	339.08	100
$(64,8)$	262.0	9.68	227.68	30	412.25	7.29	369.97	40
$(64,9)$	–	–	–	0	–	–	–	0
$(64,10)$	–	–	–	0	–	–	–	0
$(128,2)$	38.7	6.5	34.32	100	32.33	6.87	30.23	90
$(128,3)$	153.8	7.22	139.38	100	54.5	4.59	51.73	100
$(128,4)$	140.75	8.39	128.16	40	82.7	6.3	79.4	100

Continued on next page

	BTM				QAC			
Conv	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
$(128, 5)$	163.0	8.94	146.74	30	126.44	7.39	118.16	90
$(128, 6)$	210.0	7.81	186.94	30	322.5	7.16	297.78	100
$(128, 7)$	269.0	8.85	238.95	10	398.25	8.76	369.4	40
$(128, 8)$	–	–	–	0	–	–	–	0
$(128, 9)$	–	–	–	0	–	–	–	0
$(128, 10)$	–	–	–	0	–	–	–	0
$(256, 2)$	41.67	7.48	40.3	90	33.8	4.87	34.64	100
$(256, 3)$	188.5	7.9	186.54	100	53.0	5.27	54.75	100
$(256, 4)$	270.75	8.12	266.69	40	90.89	6.78	96.88	90
$(256, 5)$	227.5	9.8	222.26	20	173.8	7.11	184.27	100
$(256, 6)$	–	–	–	0	388.1	8.39	400.16	100
$(256, 7)$	–	–	–	0	498.0	7.27	501.07	10
$(256, 8)$	–	–	–	0	–	–	–	0
$(256, 9)$	–	–	–	0	–	–	–	0
$(256, 10)$	–	–	–	0	–	–	–	0

	QADA				QNDA			
Conv	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
Mean	**123.52**	**6.95**	**149.39**	**75.1**	**97.13**	**4.58**	**114.71**	**97.14**
$(2, 2)$	132.62	5.58	129.93	80	98.12	4.06	84.55	80
$(4, 2)$	101.4	5.42	106.85	100	70.7	2.34	66.92	100
$(4, 3)$	115.7	5.37	120.47	100	132.56	3.88	127.71	90
$(4, 4)$	124.3	5.74	126.38	100	129.5	3.78	120.74	100
$(8, 2)$	122.88	5.12	128.55	80	71.12	3.54	65.38	80
$(8, 3)$	66.33	6.49	70.7	90	93.33	3.34	89.99	90
$(8, 4)$	125.1	6.73	128.81	100	90.1	3.84	87.91	100
$(8, 5)$	102.3	7.28	104.77	100	82.6	5.79	83.3	100
$(8, 6)$	121.8	6.22	127.86	100	104.7	3.7	103.46	100
$(8, 7)$	158.89	6.93	176.31	90	135.0	4.97	130.73	100
$(8, 8)$	184.6	8.11	224.27	100	142.5	4.49	143.87	100
$(16, 2)$	61.3	5.61	60.78	100	54.67	2.08	51.27	90
$(16, 3)$	80.6	5.17	86.59	100	100.3	2.55	100.26	100
$(16, 4)$	126.56	7.23	132.23	90	95.5	4.44	97.24	100
$(16, 5)$	90.6	7.46	94.26	100	82.8	4.26	81.0	100
$(16, 6)$	127.5	7.34	137.42	100	116.3	3.94	120.19	100
$(16, 7)$	144.3	7.64	161.92	100	125.8	5.37	127.2	100
$(16, 8)$	153.3	7.99	189.9	100	105.5	5.07	109.11	100

Continued on next page

	QADA				QNDA			
Conv	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
$(16, 9)$	185.78	7.94	256.69	90	137.6	3.87	151.06	100
$(16, 10)$	246.86	8.03	360.54	70	176.22	6.27	203.63	90
$(32, 2)$	63.3	3.82	67.01	100	66.9	2.07	64.47	100
$(32, 3)$	45.4	7.66	49.21	100	83.6	2.99	93.87	100
$(32, 4)$	82.5	8.2	88.87	100	64.2	4.81	68.41	100
$(32, 5)$	88.6	7.69	97.37	100	84.3	3.37	88.93	100
$(32, 6)$	109.1	8.44	118.63	100	96.0	4.99	100.6	100
$(32, 7)$	166.43	8.59	197.02	70	103.4	4.53	111.76	100
$(32, 8)$	130.57	7.72	160.34	70	108.2	4.32	137.21	100
$(32, 9)$	161.71	8.35	217.13	70	111.6	4.93	125.65	100
$(32, 10)$	150.33	7.8	200.96	30	164.1	6.41	222.04	100
$(64, 2)$	52.0	5.34	58.92	100	111.67	2.97	117.66	90
$(64, 3)$	61.11	7.37	69.24	90	73.7	4.21	89.08	100
$(64, 4)$	101.67	7.73	115.04	90	54.1	2.68	58.95	100
$(64, 5)$	198.62	7.03	228.11	80	92.5	5.01	109.2	100
$(64, 6)$	70.0	7.38	82.49	10	81.0	6.34	95.44	100
$(64, 7)$	161.75	9.4	207.02	40	115.2	5.31	163.71	100
$(64, 8)$	146.33	9.19	191.32	30	108.2	4.84	127.24	100
$(64, 9)$	170.0	8.09	240.05	20	107.2	6.18	124.94	100
$(64, 10)$	198.0	6.57	326.19	10	121.5	5.09	149.86	100
$(128, 2)$	40.8	6.33	44.26	100	63.5	1.84	68.96	100
$(128, 3)$	67.44	7.38	81.15	90	66.9	3.59	81.81	100
$(128, 4)$	257.2	7.03	318.01	50	88.44	4.97	110.79	90
$(128, 5)$	275.67	9.13	336.91	30	62.89	4.35	70.19	90
$(128, 6)$	96.0	5.98	114.36	30	73.6	6.21	88.67	100
$(128, 7)$	104.0	0.7	133.61	10	94.8	6.06	123.52	100
$(128, 8)$	102.5	7.98	156.0	40	85.11	5.76	96.92	90
$(128, 9)$	–	–	–	0	119.1	6.2	181.47	100
$(128, 10)$	–	–	–	0	126.67	7.44	194.22	90
$(256, 2)$	36.8	6.5	43.9	100	46.3	3.62	60.89	100
$(256, 3)$	114.38	7.03	148.99	80	81.7	4.91	116.57	100
$(256, 4)$	170.5	8.73	227.01	40	68.8	5.36	99.93	100
$(256, 5)$	57.0	4.08	75.53	10	51.3	5.47	74.96	100
$(256, 6)$	–	–	–	0	65.56	5.74	85.24	90
$(256, 7)$	–	–	–	0	99.3	6.04	174.23	100
$(256, 8)$	–	–	–	0	93.7	6.46	144.96	100

Continued on next page

	QADA				QNDA			
Conv	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
$(256, 9)$	–	–	–	0	121.78	5.5	198.38	90
$(256, 10)$	–	–	–	0	137.67	4.54	257.37	90

C.4 Distributed DMPC problems

Table C.4: Results for the distributed optimization of the DMPC benchmark problems (mean values of the converged instances only), $\bar{t}$: mean number of iterations until convergence, $\overline{\|\mathbf{w}_p\|_2}$: mean primal residual of converged runs ($\times 10^{-3}$), $\overline{T_{\text{comp}}}$: mean computation time of converged runs (in s), $\%_c$: percentage of converged runs within $t_{\max}$ iterations.

	SG				BTM			
DMPC	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
Mean	**42.83**	**8.18**	**35.24**	**80.96**	**273.83**	**7.09**	**228.17**	**57.58**
$N_s = 5, n_{\mathbf{r}} = 2$	75.96	8.92	62.18	83.33	248.51	5.8	206.64	83.0
$N_s = 5, n_{\mathbf{r}} = 3$	79.18	9.36	65.1	84.44	309.54	6.93	257.53	52.22
$N_s = 5, n_{\mathbf{r}} = 4$	81.44	9.38	67.24	93.33	373.43	8.34	310.17	31.11
$N_s = 5, n_{\mathbf{r}} = 5$	88.44	9.62	73.32	90.0	425.67	8.98	354.84	10.0
$N_s = 10, n_{\mathbf{r}} = 2$	47.38	8.63	38.91	88.67	244.71	7.19	203.75	86.67
$N_s = 10, n_{\mathbf{r}} = 3$	44.09	8.87	36.38	88.89	305.44	7.93	253.95	47.22
$N_s = 10, n_{\mathbf{r}} = 4$	39.2	9.05	32.4	92.22	389.08	7.9	323.86	27.78
$N_s = 10, n_{\mathbf{r}} = 5$	29.04	8.76	24.14	86.67	458.0	9.42	382.54	10.0
$N_s = 20, n_{\mathbf{r}} = 2$	26.87	7.65	22.07	84.33	261.61	7.38	218.46	84.33
$N_s = 20, n_{\mathbf{r}} = 3$	21.02	8.01	17.36	88.89	301.23	8.0	250.84	43.89
$N_s = 20, n_{\mathbf{r}} = 4$	18.73	8.12	15.53	90.0	399.12	7.4	332.71	26.67
$N_s = 20, n_{\mathbf{r}} = 5$	16.15	7.53	13.42	86.67	404.0	6.16	338.32	3.33
$N_s = 50, n_{\mathbf{r}} = 2$	21.98	6.21	18.12	65.0	249.21	6.86	208.35	70.67
$N_s = 50, n_{\mathbf{r}} = 3$	16.33	6.98	13.52	57.22	317.98	8.05	264.98	30.56
$N_s = 50, n_{\mathbf{r}} = 4$	15.56	6.5	12.92	67.78	431.18	8.16	360.08	12.22
$N_s = 50, n_{\mathbf{r}} = 5$	17.44	6.56	14.51	53.33	–	–	–	–

	ADMM				QNDA			
DMPC	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
Mean	**62.24**	**7.65**	**51.92**	**99.21**	**42.6**	**7.07**	**38.31**	**98.21**
$N_s = 5, n_{\mathbf{r}} = 2$	68.5	9.22	56.33	95.33	38.93	7.48	35.09	92.0
$N_s = 5, n_{\mathbf{r}} = 3$	75.28	9.38	62.32	97.78	37.7	7.99	34.64	95.56
$N_s = 5, n_{\mathbf{r}} = 4$	78.77	9.32	65.7	100.0	30.11	7.8	27.28	97.78
$N_s = 5, n_{\mathbf{r}} = 5$	80.93	9.59	68.17	100.0	43.37	8.63	43.53	100.0
$N_s = 10, n_{\mathbf{r}} = 2$	60.5	8.86	49.88	99.67	30.63	7.41	26.58	98.0
$N_s = 10, n_{\mathbf{r}} = 3$	63.36	9.0	52.62	100.0	30.1	7.54	26.34	99.44
$N_s = 10, n_{\mathbf{r}} = 4$	66.06	9.09	55.35	100.0	40.56	7.93	36.68	100.0
$N_s = 10, n_{\mathbf{r}} = 5$	66.5	9.12	56.35	100.0	45.17	8.02	42.11	100.0
$N_s = 20, n_{\mathbf{r}} = 2$	54.66	7.5	45.24	100.0	29.46	6.79	25.41	99.67
$N_s = 20, n_{\mathbf{r}} = 3$	55.0	7.97	45.96	100.0	42.42	6.97	37.56	100.0

Continued on next page

	ADMM				QNDA			
DMPC	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
$N_s = 20, n_{\mathbf{r}} = 4$	55.2	8.28	46.63	100.0	62.19	7.11	57.37	100.0
$N_s = 20, n_{\mathbf{r}} = 5$	54.73	8.57	46.84	100.0	77.9	7.2	74.7	100.0
$N_s = 50, n_{\mathbf{r}} = 2$	57.15	3.88	47.82	100.0	41.31	6.15	36.06	100.0
$N_s = 50, n_{\mathbf{r}} = 3$	60.78	5.14	51.65	100.0	68.63	5.71	61.71	99.44
$N_s = 50, n_{\mathbf{r}} = 4$	61.62	5.42	53.29	100.0	82.43	6.49	77.11	100.0
$N_s = 50, n_{\mathbf{r}} = 5$	61.97	5.47	54.67	100.0	111.93	6.8	109.76	100.0

C.5 Distributed clustering problems

Table C.5: Results for the distributed optimization of the clustering benchmark problems, $\bar{t}$: mean number of performed iterations, $\overline{\text{rel. DG}}$: mean relative duality gap (in %), $\overline{T_{\text{comp}}}$: mean computation time (in s).

	SG			BTM		
Clustering	$\bar{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$	$\bar{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$
Mean	**136.75**	**2.27**	**996.28**	**57.44**	**1.86**	**515.77**
2N2D3K	126.0	1.94	166.08	68.0	1.84	95.01
2N2D4K	113.2	0.89	431.02	64.4	0.8	348.56
2N3D3K	120.0	1.8	223.69	71.6	1.6	167.72
2N3D4K	115.6	0.27	782.18	13.6	0.1	93.92
2N4D3K	91.6	0.31	184.21	36.2	0.16	108.0
2N4D4K	90.4	0.25	965.26	8.6	0.08	93.7
3N2D3K	138.4	7.01	404.59	123.2	6.14	424.47
3N2D4K	150.0	4.7	879.48	62.8	3.99	751.33
3N3D3K	150.0	2.33	301.63	67.0	1.76	160.05
3N3D4K	150.0	0.82	1469.97	66.6	0.35	906.26
3N4D3K	150.0	2.07	354.48	37.2	1.63	110.82
3N4D4K	150.0	1.06	3295.09	37.8	0.56	820.42
4N2D3K	150.0	5.01	311.78	103.2	3.66	262.33
4N2D4K	150.0	7.58	1319.14	93.8	5.71	1346.59
4N3D3K	150.0	1.32	317.69	6.6	0.15	16.75
4N3D4K	150.0	1.55	2786.47	65.8	0.53	2046.2
4N4D3K	150.0	1.57	441.69	35.0	0.42	122.26
4N4D4K	150.0	1.7	2593.41	7.2	0.14	118.24

	QNDA		
Clustering	$\bar{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$
Mean	**54.48**	**1.81**	**483.22**
2N2D3K	63.2	1.82	92.57
2N2D4K	62.2	0.73	345.59
2N3D3K	62.2	1.58	150.11
2N3D4K	5.0	0.06	34.76
2N4D3K	32.8	0.12	97.89
2N4D4K	4.8	0.08	47.19
3N2D3K	121.0	6.21	412.26
3N2D4K	64.2	3.98	747.3
3N3D3K	64.0	1.71	151.79

Continued on next page

Clustering	$\bar{t}$	QNDA rel. DG	$\overline{T_{\text{comp}}}$	
3N3D4K	63.8	0.28	858.76	
3N4D3K	35.0	1.36	111.44	
3N4D4K	37.2	0.46	731.33	
4N2D3K	94.6	3.61	249.28	
4N2D4K	93.8	5.68	1281.33	
4N3D3K	9.4	0.17	22.76	
4N3D4K	66.0	0.49	1901.69	
4N4D3K	37.4	0.41	129.74	
4N4D4K	9.8	0.17	178.24	

D Further publications by the author

Published

Peer-reviewed Journals

C. Klanke, **V. Yfantis**, F. Corominas., S. Engell, 2021. **Short-term scheduling of make-and-pack processes in the consumer goods industry using discrete-time and precedence-based MILP models**. *Computers & Chemical Engineering*. DOI: https://doi.org/10.1016/j.compchemeng.2021.107453.

Conference Proceedings

V. Yfantis, A. Babskiy, B. Dörig, T. Winterer, A. Wagner, M. Ruskowski, 2023. **Evolutionary Algorithm-based Optimal Parametrization of Multi-objective Mixed-integer Linear Programming Scheduling Models**. *22nd IFAC World Congress*. DOI: https://doi.org/10.1016/j.ifacol.2023.10.185.

W. Motsch, **V. Yfantis**, A. Wagner, M. Ruskowski, 2023. **Utilizing Extensive-Form Games for Energy-aware Production Plan Adaptation in Modular Skill-based Production Systems**. *22nd IFAC World Congress*. DOI: https://doi.org/10.1016/j.ifacol.2023.10.1421.

N. Gafur, L. Weber, **V. Yfantis**, A. Wagner, M. Ruskowski, 2022. **Dynamic path planning and reactive scheduling for a robotic manipulator using nonlinear model predictive control**. *30th Mediterranean Conference on Control and Automation (MED)*. DOI: https://doi.org/10.1109/MED54222.2022.9837147.

V. Yfantis, A. Babskiy, C. Klanke, M. Ruskowski, S. Engell, 2022. **An improved optimization model for scheduling of an industrial formulation plant based on integer linear programming**. *14th International Symposium on Process Systems Engineering*. DOI: https://doi.org/10.1016/B978-0-323-85159-6.50081-6.

S. Jungbluth, N. Gafur, J. Popper, **V. Yfantis**, M. Ruskowski, 2022. **Reinforcement Learning-based Scheduling of a Job-Shop Process with Distributedly Controlled Robotic Manipulators for Transport Operations**. *14th IFAC Workshop on Intelligent Manufacturing Systems (IMS 2022)*. DOI: https://doi.org/10.1016/j.ifacol.2022.04.186.

N. Gafur, **V. Yfantis**, M. Ruskowski, 2021. **Optimal scheduling and non-cooperative distributed model predictive control for multiple robotic manipulators**. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. DOI: https://doi.org/10.1109/IROS51168.2021.9636118.

J. Popper, **V. Yfantis**, M. Ruskowski, 2021. **Simultaneous production and AGV scheduling using multi-agent deep reinforcement learning**. *54th CIRP Conference on Manufacturing Systems (CMS 2021)*. DOI: https://doi.org/10.1016/j.procir.2021.11.257.

C. Klanke, D. Bleidorn, **V. Yfantis**, S. Engell, 2021. **Combining Constraint Programming and Temporal Decomposition Approaches-Scheduling of an Industrial Formulation Plant**. *18th International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR 2021)*. DOI: https://doi.org/10.1007/978-3-030-78230-6_9.

V. Yfantis, S. Büscher, C. Klanke, F. Corominas, S. Engell, 2020. **A Two-stage Simulated Annealing-based Scheduling Algorithm for a Make-and-Pack Production Plant**. *21st IFAC World Congress*. DOI: https://doi.org/10.1016/j.ifacol.2020.12.2861.

N. Bajcinca, M.V.A. Pedrosa, **V. Yfantis**, 2020. **Mixed-Integer Model Predictive Control of Hybrid Impulsive Linear Systems**. *21st IFAC World Congress*. DOI: https://doi.org/10.1016/j.ifacol.2020.12.327.

A. Tika, N. Gafur, **V. Yfantis**, N. Bajcinca, 2020. **Optimal scheduling and model predictive control for trajectory planning of cooperative robot manipulators**. *21st IFAC World Congress*. DOI: https://doi.org/10.1016/j.ifacol.2020.12.2136.

C. Klanke, **V. Yfantis**, F. Corominas, S. Engell, 2020. **Scheduling of a Large-scale Industrial Make-and-Pack Process with Finite Intermediate Buffer using Discrete-time and Precedence-based Models**. *30th European Symposium on Computer Aided Process Engineering (ESCAPE 30)*. DOI: https://doi.org/10.1016/B978-0-12-823377-1.50193-2.

V. Yfantis, F. Corominas, S. Engell, 2019. **Scheduling of a Consumer Goods Production Plant with Intermediate Buffer by Decomposition and Mixed-integer Linear Programming**. *9th IFAC Conference on Manufacturing Modelling, Management and Control (MIM 2019)*. DOI: https://doi.org/10.1016/j.ifacol.2019.11.469.

V. Yfantis, T. Siwczyk, M. Lampe, N. Kloye, M. Remelhe, S. Engell, 2019. **Iterative Medium-Term Production Scheduling of an Industrial Formulation Plant**. *29th European Symposium on Computer Aided Process Engineering (ESCAPE 29)*. DOI: `https://doi.org/10.1016/B978-0-12-818634-3.50004-7`.

S. Wenzel, **V. Yfantis**, W. Gao, 2017. **Comparison of regression data selection strategies for quadratic approximation in RTO**. *27th European Symposium on Computer Aided Process Engineering (ESCAPE 27)*. DOI: `https://doi.org/10.1016/B978-0-444-63965-3.50287-7`.

Conferences without Proceedings

A. Elekidis, **V. Yfantis**, F. Corominas, M.C. Georgiadis, S. Engell, 2019. **Optimal Production Scheduling in the Packaged Consumer Goods Industry**. *12th European Congress of Chemical Engineering (ECCE12)*.

Submitted

Peer-reviewed Journals

V. Yfantis, A. Wagner M. Ruskowski. **Numerical Benchmarking of Dual Decomposition-based Optimization Algorithms for Distributed Model Predictive Control**. *Results in Control and Optimization*.

V. Yfantis, A. Wagner M. Ruskowski. **Federated K-Means Clustering via Dual Decomposition-based Distributed Optimization**. *Franklin Open*.

Conference Proceedings

M. Klostermeier, **V. Yfantis**, A. Wagner M. Ruskowski. **Numerical Study on the Parallelization of Dual Decomposition-based Distributed Mixed-Integer Programming**. *European Control Conference*.

E Supervised theses

Rheinland-Pfälzische Techische Universität Kaiserslautern-Landau

M. Klostermeier, 2023. **Optimal Parameterization of Multi-Objective MILP Models using Metaheuristic Algorithms**, Master Thesis.

M. Klostermeier, 2022. **Mixed-Integer Model Predictive Control of Hybrid Impulsive Nonlinear Systems**, Research Project.

X. Wang, 2022. **Entwicklung eines Erkundungsschrittverfahrens in der verteilten Optimierung**, Bachelor Thesis.

A. Babskiy, 2022. **Optimale Parametrierung von Soft-Constraints in Produktionsplanungsmodellen durch Methoden der Computational Intelligence**, Master Thesis.

A. Babskiy, 2022. **Produktionsplanung einer Industriellen Formulieranlage durch gemischt-ganzzahlige lineare Programmierung**, Research Project.

M. Abdelraman, 2022. **Scheduling of a Job-shop process with variable transport times by mixed-integer programming**, Master Thesis.

N. Bach, 2021. **Optimales Demand-Side Management einer flexiblen Produktionsanlage**, Master Thesis.

S. Jungbluth, 2020. **Job-Shop Scheduling mit Reinforcement Learning**, Master Thesis.

Technische Universität Dortmund

A. Preuß, 2019. **Hybrid Scheduling of an Industrial Formulation Plant by Metaheuristic Optimization and Constraint Programming**, Master Thesis.

S. Büscher, 2019. **Metaheuristic-based Scheduling of an Agile Consumer Goods Production Plant**, Master Thesis.

Bibliography

[AAK+22] R.S. Antunes, C. André da Costa, A. Küderle, I.A. Yari, and B. Eskofier. Federated learning for healthcare: Systematic review and architecture proposal. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(4):1–23, 2022.

[ABG+20] 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.

[AD14] C. Altay and H. Deliç. Distributed energy management of microgrids with Dantzig-Wolfe decomposition. In *IEEE PES Innovative Smart Grid Technologies, Europe*, pages 1–5. IEEE, 2014.

[AHL12] D. Aloise, P. Hansen, and L. Liberti. An improved column generation algorithm for minimum sum-of-squares clustering. *Mathematical Programming*, 131:195–220, 2012.

[AHU58] K.J. Arrow, L. Hurwicz, and H. Uzawa. *Studies in linear and non-linear programming*. Stanford University Press, 1958.

[AKM05] T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, 2005.

[ALT19] M.F. Anjos, A. Lodi, and M. Tanneau. A decentralized framework for the optimal coordination of distributed energy resources. *IEEE Transactions on Power Systems*, 34(1):349–359, 2019.

[AW13] T. Achterberg and R. Wunderling. Mixed integer programming: Analyzing 12 years of progress. *Facets of Combinatorial Optimization: Festschrift for Martin Grötschel*, pages 449–481, 2013.

[BAS+12] B. Biegel, P. Andersen, J. Stoustrup, and J. Bendtsen. Congestion management in a smart grid via shadow prices. *IFAC Proceedings Volumes*, 45(21):518–523, 2012.

[BCC+15] A. Barbato, A. Capone, L. Chen, F. Martignon, and S. Paris. A distributed demand-side management framework for the smart grid. *Computer Communications*, 57:13–24, 2015.

[BEK+17] J. Bezanson, A. Edelman, S. Karpinski, and V.B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[Ben62] J.F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962.

[Ber99] D. P. Bertsekas. *Nonlinear programming*. Athena Scientific, 1999.

[BFL07] L. Bertacco, M. Fischetti, and A. Lodi. A feasibility pump heuristic for general mixed-integer problems. *Discrete Optimization*, 4(1):63–76, 2007.

[Bix12] R.E. Bixby. A brief history of linear and mixed-integer programming computation. *Documenta Mathematica*, 2012:107–121, 2012.

[BKM14] A. Bagirov, N. Karmitsa, and M. Mäkelä. *Introduction to Nonsmooth Optimization: Theory, Practice and Software*. Springer, 2014.

[BL11] J.R. Birge and F. Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[BLY+15] M.A. Bragin, P.B. Luh, J.H. Yan, N. Yu, and G.A. Stern. Convergence of the surrogate Lagrangian relaxation method. *Journal of Optimization Theory and Applications*, 164(1):173–201, 2015.

[BLY+19] M.A. Bragin, P.B. Luh, B. Yan, and X. Sun. A scalable solution methodology for mixed-integer linear programming problems arising in automation. *IEEE Transactions on Automation Science and Engineering*, 16(2):531–541, 2019.

[BPC+11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[BSA14] B. Biegel, J. Stoustrup, and P. Andersen. Distributed MPC via dual decomposition. In *Distributed Model Predictive Control Made Easy*, pages 179–192. Springer, 2014.

[BSB+12] B. Biegel, J. Stoustrup, J. Bendtsen, and P. Andersen. Model predictive control for power flows in networks with limited capacity. In *2012 American Control Conference (ACC)*, pages 2959–2964. IEEE, 2012.

[BT89] D. Bertsekas and J. Tsitsiklis. *Parallel and distributed computation*. Prentice Hall, 1989.

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[BY06] A.M Bagirov and J. Yearwood. A new nonsmooth optimization algorithm

for minimum sum-of-squares clustering problems. *European Journal of Operational Research*, 170(2):578–596, 2006.

[CBK+21] M.A.P. Chamikara, P. Bertok, I. Khalil, D. Liu, and S. Camtepe. Privacy preserving distributed machine learning with federated learning. *Computer Communications*, 171:112–125, 2021.

[CDZ15] N. Chatzipanagiotis, D. Dentcheva, and M.M. Zavlanos. An augmented Lagrangian method for distributed optimization. *Mathematical Programming*, 152(1):405–434, 2015.

[CJK+02] E. Camponogara, D. Jia, B.H. Krogh, and S. Talukdar. Distributed model predictive control. *IEEE Control Systems Magazine*, 22(1):44–52, 2002.

[CNN21] A. Camisa, I. Notarnicola, and G. Notarstefano. Distributed primal decomposition for large-scale MILPs. *IEEE Transactions on Automatic Control*, 67(1):413–420, 2021.

[CSdlP+13] P.D. Christofides, R. Scattolini, D. M. de la Pena, and J. Liu. Distributed model predictive control: A tutorial review and future research directions. *Computers & Chemical Engineering*, 51:21–41, 2013.

[CSV09] A.R. Conn, K. Scheinberg, and L.N. Vicente. *Introduction to Derivative-Free Optimization*. Society for Industrial and Applied Mathematics, 2009.

[CSZ+12] C. Conte, T. Summers, M.N. Zeilinger, M. Morari, and C.N. Jones. Computational aspects of distributed optimization in model predictive control. In *51st IEEE conference on Decision and Control (CDC)*, pages 6819–6824. IEEE, 2012.

[CWS18] R. Chen, J. Wang, and H. Sun. Clearing and pricing for coordinated gas and electricity day-ahead markets considering wind power uncertainty. *IEEE Transactions on Power Systems*, 33(3):2496–2508, 2018.

[CZ17] N. Chatzipanagiotis and M.M. Zavlanos. On the convergence of a distributed augmented Lagrangian method for nonconvex optimization. *IEEE Transactions on Automatic Control*, 62(9):4405–4420, 2017.

[Dan51] G.B Dantzig. Maximization of a linear function of variables subject to linear inequalities. *Activity Analysis of Production and Allocation*, 13:339–347, 1951.

[Dan90] G.B. Dantzig. Origins of the simplex method. In *A history of scientific computing*, pages 141–151. 1990.

[DHL17] I. Dunning, J. Huchette, and M. Lubin. JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[DKD11] M.D. Doan, T. Keviczky, and B. De Schutter. A distributed optimization-based approach for hierarchical MPC of large-scale systems with coupled dynamics and constraints. In *50th IEEE Conference on Decision and Control (CDC) and European Control Conference (ECC)*, pages 5236–5241. IEEE, 2011.

[DLS21] D. K. Dennis, T. Li, and V. Smith. Heterogeneity for the win: One-shot federated clustering. In *International Conference on Machine Learning*, pages 2611–2620. PMLR, 2021.

[DMC15] N. Dhanachandra, K. Manglem, and Y.J. Chanu. Image segmentation using k-means clustering algorithm and subtractive clustering algorithm. *Procedia Computer Science*, 54:764–771, 2015.

[DPW09] D.-Z. Du, P.M. Pardalos, and W. Wu. History of optimization. *Encyclopedia of Optimization*, pages 1538–1542, 2009.

[DRP05] E. Danna, E. Rothberg, and C.L. Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, 102:71–90, 2005.

[DW60] G.B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960.

[EDC14] M. Ellis, H. Durand, and P.D. Christofides. A tutorial review of economic model predictive control methods. *Journal of Process Control*, 24(8):1156–1178, 2014.

[EMR16] M. Eisen, A. Mokhtari, and A. Ribeiro. A decentralized quasi-Newton method for dual formulations of consensus optimization. In *55th IEEE Conference on Decision and Control (CDC)*, pages 1951–1958. IEEE, 2016.

[EMR17] M. Eisen, A. Mokhtari, and A. Ribeiro. Decentralized quasi-Newton methods. *IEEE Transactions on Signal Processing*, 65(10):2613–2628, 2017.

[Eng07] S. Engell. Feedback control for optimal process operation. *Journal of Process Control*, 17(3):203–219, 2007.

[ESK+22] S. Eser, P. Stoffel, A. Kümpel, and D. Müller. Distributed model predictive control of a nonlinear building energy system using consensus ADMM. In *30th Mediterranean Conference on Control and Automation (MED)*, pages 902–907. IEEE, 2022.

[Eve63] H. Everett. Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, (11 (3)):399–417, 1963.

[FCG10] P.A. Forero, A. Cano, and G.B Giannakis. Consensus-based distributed support vector machines. *Journal of Machine Learning Research*, 11(5), 2010.

[FCG11] P.A. Forero, A. Cano, and G.B. Giannakis. Distributed clustering using wireless sensor networks. *IEEE Journal of Selected Topics in Signal Processing*, 5(4):707–724, 2011.

[FG83] M. Fortin and R. Glowinski. *Augmented Lagrangian methods: Applications to the numerical solution of boundary-value problems*, volume 15 of *Studies in mathematics and its applications*. North-Holland Pub. Co, Amsterdam and New York, 1983.

[FIC22] FICO Xpress. FICO Xpress Optimization Help, 2022.

[GDK+13] P. Giselsson, M.D. Doan, T. Keviczky, B. De Schutter, and A. Rantzer. Accelerated gradient methods and dual decomposition in distributed model predictive control. *Automatica*, 49(3):829–833, 2013.

[GF10] V. Gunnerud and B. Foss. Oil production optimization—a piecewise linear model, solved with two decomposition strategies. *Computers & Chemical Engineering*, 34(11):1803–1812, 2010.

[GGN21] C. Gambella, B. Ghaddar, and J. Naoum-Sawaya. Optimization problems for machine learning: A survey. *European Journal of Operational Research*, 290(3):807–828, 2021.

[GH14] L. Georgopoulos and M. Hasler. Distributed machine learning in networks by consensus. *Neurocomputing*, 124:2–12, 2014.

[GKW07] Z. Guo, G.J. Koehler, and A.B. Whinston. A market-based optimization algorithm for distributed systems. *Management Science*, 53(8):1345–1358, 2007.

[GKW+22] N. Gafur, G. Kanagalingam, A. Wagner, and M. Ruskowski. Dynamic collision and deadlock avoidance for multiple robotic manipulators. *IEEE Access*, 2022.

[GM76] D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17–40, 1976.

[GMM14] S.I. Guler, M. Menendez, and L. Meier. Using connected vehicle technology to improve the efficiency of intersections. *Transportation Research Part C: Emerging Technologies*, 46:121–131, 2014.

[GOS+14] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk. Fast alternat-

ing direction optimization methods. *SIAM Journal on Imaging Sciences*, 7(3):1588–1623, 2014.

[GR10] P. Giselsson and A. Rantzer. Distributed model predictive control with suboptimality and stability guarantees. In *49th IEEE Conference on Decision and Control (CDC)*, pages 7272–7277. IEEE, 2010.

[GR13] P. Giselsson and A. Rantzer. On feasibility, stability and performance in distributed model predictive control. *IEEE Transactions on Automatic Control*, 59(4):1031–1036, 2013.

[Gur23] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023.

[GWE16] W. Gao, S. Wenzel, and S. Engell. A reliable modifier-adaptation strategy for real-time optimization. *Computers & Chemical Engineering*, 91(9):318–328, 2016.

[HFD16] 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.

[HH16] H. He and D. Han. A distributed Douglas-Rachford splitting method for multi-block convex minimization problems. *Advances in Computational Mathematics*, 42(1):27–53, 2016.

[HLR22] V. Hegiste, T. Legler, and M. Ruskowski. Application of federated machine learning in manufacturing. In *2022 International Conference on Industry 4.0 Technology (I4Tech)*, pages 1–8. IEEE, 2022.

[HS22] B. Houska and J. Shi. Distributed MPC with ALADIN–a tutorial. *arXiv preprint arXiv:2204.01654*, 2022.

[HTP17] S. Han, U. Topcu, and G.J. Pappas. Differentially private distributed constrained optimization. *IEEE Transactions on Automatic Control*, 62(1):50–64, 2017.

[HVP+16] R. Halvgaard, L. Vandenberghe, N.K. Poulsen, H. Madsen, and J.B. Jørgensen. Distributed model predictive control for smart energy systems. *IEEE Transactions on Smart Grid*, 7(3):1675–1682, 2016.

[HWL+17] C. He, L. Wu, T. Liu, and M. Shahidehpour. Robust co-optimization scheduling of electricity and natural gas systems via ADMM. *IEEE Transactions on Sustainable Energy*, 8(2):658–670, 2017.

[HYW00] B.S. He, H. Yang, and S.L. Wang. Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities. *Journal of Optimization Theory and Applications*, 106(2):337–356, 2000.

[IBM22] IBM ILOG CPLEX. User's Manual for CPLEX v20.1, 2022.

[JI13] J.-Y. Joo and M.D. Ilić. Multi-layered optimization of demand resources using Lagrange dual decomposition. *IEEE Transactions on Smart Grid*, 4(4):2081–2088, 2013.

[JKO+20] J.C. Jiang, B. Kantarci, S. Oktug, and T. Soyata. Federated learning in smart city sensing: Challenges and opportunities. *Sensors*, 20(21):6230, 2020.

[JLW+15] D. Jia, K. Lu, J. Wang, X. Zhang, and X. Shen. A survey on platoon-based vehicular cyber-physical systems. *IEEE Communications Surveys & Tutorials*, 18(1):263–284, 2015.

[JT16] L. Jia and L. Tong. Dynamic pricing and distributed energy management for demand response. *IEEE Transactions on Smart Grid*, 7(2):1128–1136, 2016.

[Kan39] L.V. Kantorovich. Mathematical methods of production organization and planning. *Leningrad, LSU Publ*, 1939.

[Kar39] W. Karush. Minima of function of several variables with inequalities as side conditions, 1939.

[Kar84] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*, pages 302–311, 1984.

[KBL+19] J. Kronqvist, D.E. Bernal, A. Lundell, and I.E. Grossmann. A review and comparison of solvers for convex MINLP. *Optimization and Engineering*, 20:397–455, 2019.

[KBP+22] T. Koch, T. Berthold, J. Pedersen, and C. Vanaret. Progress in mathematical programming solvers from 2001 to 2020. *EURO Journal on Computational Optimization*, 10:100031, 2022.

[KBS+18] T. Kansal, S. Bahuguna, V. Singh, and T. Choudhury. Customer segmentation using k-means clustering. In *International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*, pages 135–139. IEEE, 2018.

[KBT17] N. Karmitsa, A.M. Bagirov, and S. Taheri. New diagonal bundle method for clustering problems in large data sets. *European Journal of Operational Research*, 263(2):367–379, 2017.

[KCD15] A. Kozma, C. Conte, and M. Diehl. Benchmarking large-scale distributed convex quadratic programming algorithms. *Optimization Methods and Soft-*

ware, 30(1):191–214, 2015.

[KKN20] H.H. Kumar, V.R. Karthik, and M.K. Nair. Federated k-means clustering: A novel edge AI based approach for privacy preservation. In *2020 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pages 52–56. IEEE, 2020.

[KKP20] V. Kulkarni, M. Kulkarni, and A. Pant. Survey of personalization techniques for federated learning. In *4th World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pages 794–797. IEEE, 2020.

[KMA19] J. Köhler, M. A Müller, and F. Allgöwer. Distributed model predictive control—recursive feasibility under inexact dual optimization. *Automatica*, 102:1–9, 2019.

[KMR15] J. Konečnỳ, B. McMahan, and D. Ramage. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*, 2015.

[KMR+16] J. Konečnỳ, B. McMahan, D. Ramage, and P. Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

[KT51] H. Kuhn and A. Tucker. Nonlinear programming. In *Proceedings of the 2nd Berkeley Symposium on Mathematical Statictics and Probability*, pages 481–492. Univ. of California Press, 1951.

[KYW+24] M. Klostermeier, V. Yfantis, A. Wagner, and M. Ruskowski. Numerical study on the parallelization of dual decomposition-based distributed mixed-integer programming. In *IEEE European Control Conference (ECC) (submitted)*, 2024.

[LD60] A.H. Land and A.G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

[LFT+20] L. Li, Y. Fan, M. Tse, and K.-Y. Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020.

[LHB+22] S. Li, S. Hou, B. Buyukates, and S. Avestimehr. Secure federated clustering. *arXiv preprint arXiv:2205.15564*, 2022.

[LHZ+22] J. Liu, J. Huang, Y. Zhou, X. Li, S. Ji, H. Xiong, and D. Dou. From distributed machine learning to federated learning: A survey. *Knowledge and Information Systems*, 64(4):885–917, 2022.

[Li18] M. Li. Generalized Lagrange multiplier method and KKT conditions with

an application to distributed optimization. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 66(2):252–256, 2018.

[LLB+20] A.-B. Liu, P.B. Luh, M.A. Bragin, and B. Yan. Ordinal-optimization concept enabled decomposition and coordination of mixed-integer linear programming problems. *IEEE Robotics and Automation Letters*, 5(4):5051–5058, 2020.

[LLH+20] W.Y.B. Lim, N.C. Luong, D.T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(3):2031–2063, 2020.

[Llo82] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[LMZ+18] Y. Liu, J.M. Montenbruck, D. Zelazo, M. Odelga, S. Rajappa, H.H. Bülthoff, F. Allgöwer, and A. Zell. A distributed control approach to formation balancing and maneuvering of multiple multirotor UAVs. *IEEE Transactions on Robotics*, 34(4):870–882, 2018.

[Lou03] Robin Lougee-Heimer. The common optimization interface for operations research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, 47(1):57–66, 2003.

[LSV07] Q. Le, A. Smola, and S. Vishwanathan. Bundle methods for machine learning. *Advances in Neural Information Processing Systems*, 20, 2007.

[Lun14] J. Lunze. *Regelungstechnik 2: Mehrgrößensysteme, Digitale Regelung*. Springer-Verlag, 2014.

[LZ18] Y. Lu and M. Zhu. Privacy preserving distributed optimization using homomorphic encryption. *Automatica*, 96(6):314–325, 2018.

[MA17] M. A Müller and F. Allgöwer. Economic and distributed model predictive control: Recent developments in optimization-based control. *SICE Journal of Control, Measurement, and System Integration*, 10(2):39–52, 2017.

[Mäk02] M. Mäkelä. Survey of bundle methods for nonsmooth optimization. *Optimization Methods and Software*, 17(1):1–29, 2002.

[Max21] L.S. Maxeiner. *Dual-based Methods for Distributed Optimization of Interconnected Systems*. Shaker Verlag, 2021.

[MdlPC11a] J.M. Maestre, D.M. de la Peña, and E. F. Camacho. Distributed model predictive control based on a cooperative game. *Optimal Control Applications and Methods*, 32(2):153–176, 2011.

[MdlPC+11b] J.M. Maestre, D.M. de la Peña, EF Camacho, and T Alamo. Distributed model predictive control based on agent negotiation. *Journal of Process Control*, 21(5):685–697, 2011.

[MDS+20] W. Motsch, A. David, K. Sivalingam, A. Wagner, and M. Ruskowski. Approach for dynamic price-based demand side management in cyber-physical production systems. *Procedia manufacturing*, 51:1748–1754, 2020.

[ME17] L.S. Maxeiner and S. Engell. Hierarchical MPC of batch reactors with shared resources. *IFAC-PapersOnLine*, 50(1):12041–12046, 2017.

[ME20] L.S. Maxeiner and S. Engell. An accelerated dual method based on analytical extrapolation for distributed quadratic optimization of large-scale production complexes. *Computers & Chemical Engineering*, 135(1):106728, 2020.

[MGP11] S. Mouret, I.E. Grossmann, and P. Pestiaux. A new Lagrangian decomposition approach applied to the integration of refinery planning and crude-oil scheduling. *Computers & Chemical Engineering*, 35(12):2750–2766, 2011.

[MKJ+17] C. Ma, J. Konečnỳ, M. Jaggi, V. Smith, M.I. Jordan, P. Richtárik, and M. Takáč. Distributed optimization with arbitrary local solvers. *Optimization Methods and Software*, 32(4):813–848, 2017.

[MMR+17] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[MMW+02] H. Marchand, A. Martin, R. Weismantel, and L. Wolsey. Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics*, 123(1-3):397–446, 2002.

[MN14] J.M. Maestre and R.R. Negenborn. *Distributed model predictive control made easy*. Springer, 2014.

[MRR+00] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[MRT18] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[MSN+13] R. Martí, D. Sarabia, D. Navia, and C. de Prada. A method to coordinate decentralized NMPC controllers in oxygen distribution networks. *Computers & chemical engineering*, 59:122–137, 2013.

[MYW+23] W. Motsch, V. Yfantis, A. Wagner, and M. Ruskowski. Utilizing extensive-

form games for energy-aware production plan adaptation in modular skill-based production systems. *22nd IFAC World Congress (accepted)*, 2023.

[NB01] A. Nedić and D. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization*, 12(1):109–138, 2001.

[Ned20] A. Nedić. Distributed gradient methods for convex machine learning problems in networks: Distributed optimization. *IEEE Signal Processing Magazine*, 37(3):92–101, 2020.

[Nes83] Y. Nesterov. A method of solving a convex programming problem with convergence rate $\mathcal{O}(1/k^2)$. In *Sov. Math. Dokl*, volume 27, pages 372–376, 1983.

[Nes04] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87 of *Applied Optimization*. Kluwer Acad. Publ, Boston, Mass., 2004.

[NFN16] I. Notarnicola, M. Franceschelli, and G. Notarstefano. A duality-based approach for distributed min-max optimization with application to demand side management. In *55th IEEE Conference on Decision and Control (CDC)*, pages 1877–1882. IEEE, 2016.

[NL18] A. Nedić and J. Liu. Distributed optimization for control. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:77–103, 2018.

[NM14] R.R. Negenborn and J.M. Maestre. Distributed model predictive control: An overview and roadmap of future research opportunities. *IEEE Control Systems Magazine*, 34(4):87–97, 2014.

[NN20] I. Notarnicola and G. Notarstefano. Constraint-coupled distributed optimization: A relaxation and duality approach. *IEEE Transactions on Control of Network Systems*, 7(1):483–492, 2020.

[NNC19] G. Notarstefano, I. Notarnicola, and A. Camisa. Distributed optimization for smart cyber-physical networks. *Foundations and Trends® in Systems and Control*, 7(3):253–383, 2019.

[NND11] I. Necoara, V. Nedelcu, and I. Dumitrache. Parallel and distributed optimization methods for estimation and control in networks. *Journal of Process Control*, 21(5):756–766, 2011.

[NS09] I. Necoara and J.A.K. Suykens. Interior-point Lagrangian decomposition method for separable convex optimization. *Journal of Optimization Theory and Applications*, 143(3):567–588, 2009.

[NSH15] H.K. Nguyen, J.B. Song, and Z. Han. Distributed demand side management with energy storage in smart grid. *IEEE Transactions on Parallel and Distributed Systems*, 26(12):3346–3357, 2015.

[NW06] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[PAL14] P. Pflaum, M. Alamir, and M.Y. Lamoudi. Comparison of a primal and a dual decomposition for distributed MPC in smart districts. In *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 55–60. IEEE, 2014.

[PAL15] P. Pflaum, M. Alamir, and M.Y. Lamoudi. Scalability study for a hierarchical NMPC scheme for resource sharing problems. In *IEEE European Control Conference (ECC)*, pages 1468–1473. IEEE, 2015.

[PB14] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[PC06] 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, 2006.

[Ped21] W. Pedrycz. Federated FCM: clustering under privacy requirements. *IEEE Transactions on Fuzzy Systems*, 30(8):3384–3388, 2021.

[PG13] D. Peteiro-Barral and B. Guijarro-Berdiñas. A survey of methods for distributed machine learning. *Progress in Artificial Intelligence*, 2:1–11, 2013.

[PLF+18] H.X. Pham, H. M. La, D. Feil-Seifer, and M.C. Deans. A distributed control framework of multiple unmanned aerial vehicles for dynamic wildfire tracking. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(4):1537–1548, 2018.

[Pow87] M.J.D. Powell. Updating conjugate directions by the BFGS formula. *Mathematical Programming*, 38:29–46, 1987.

[RCG+17] R. Rahmaniani, T.G. Crainic, M. Gendreau, and W. Rei. The Benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801–817, 2017.

[RCP+20] M. Razzanelli, E. Crisostomi, L. Pallottino, and G. Pannocchia. Distributed model predictive control for energy management in a network of microgrids using the dual decomposition method. *Optimal Control Applications and Methods*, 41(1):25–41, 2020.

[RHH+20] M. Ruskowski, A. Herget, J. Hermann, W. Motsch, P. Pahlevanne-

jad, A. Sidorenko, S. Bergweiler, A. David, C. Plociennik, J. Popper, K. Sivalingam, and A. Wagner. Production Bots für Production Level 4. *atp-Magazin*, 62(9):62–71, 2020.

[RNF23] M. Reiszadeh, H. Narimani, and M. S. Fazel. Improving convergence properties of autonomous demand side management algorithms. *International Journal of Electrical Power & Energy Systems*, 146:108764, 2023.

[Rot07] E. Rothberg. An evolutionary algorithm for polishing mixed integer programming solutions. *INFORMS Journal on Computing*, 19(4):534–541, 2007.

[RSB+19] K. Rahimi-Adli, P.D. Schiermoch, B. Beisheim, S. Wenzel, and S. Engell. A model identification approach for the evaluation of plant efficiency. In *Computer Aided Chemical Engineering*, volume 46, pages 913–918. Elsevier, 2019.

[Sca09] R. Scattolini. Architectures for distributed and hierarchical model predictive control–a review. *Journal of process control*, 19(5):723–731, 2009.

[SCZ+19] S. Sun, Z. Cao, H. Zhu, and J. Zhao. A survey of optimization methods from a machine learning perspective. *IEEE Transactions on Cybernetics*, 50(8):3668–3681, 2019.

[SEF22] G. Stomberg, A. Engelmann, and T. Faulwasser. A compendium of optimization algorithms for distributed linear-quadratic MPC. *at-Automatisierungstechnik*, 70(4):317–330, 2022.

[SGC+22] P. Scarabaggio, S. Grammatico, R. Carli, and M. Dotoli. Distributed demand side management with stochastic wind power forecasting. *IEEE Transactions on Control Systems Technology*, 30(1):97–112, 2022.

[SGK17] G.M. Scheepmaker, R.M.P. Goverde, and L.G. Kroon. Review of energy-efficient train control and timetabling. *European Journal of Operational Research*, 257(2):355–376, 2017.

[Sia14] P. Siano. Demand response and smart grids – a survey. *Renewable and Sustainable Energy Reviews*, 30:461–478, 2014.

[SKR85] N.Z. Shor, K.C. Kiwiel, and A. Ruszcayǹski. Minimization methods for non-differentiable functions, 1985.

[SSS+16] M.H. Shoreh, P. Siano, M. Shafie-Khah, V. Loia, and J.P.S. Catalão. A survey of industrial applications of demand response. *Electric Power Systems Research*, 141:31–49, 2016.

[SVA75] N.R. Sandell, P. Varaiya, and M. Athans. A survey of decentralized control

methods for large scale systems. *Systems Eng. for Power, US Dept. of Commerce*, pages 334–335, 1975.

[SVR+10] B.T. Stewart, A.N. Venkat, J.B. Rawlings, S. Wright, and G. Pannocchia. Cooperative distributed model predictive control. *Systems & Control Letters*, 59(8):460–469, 2010.

[SW22] M Stallmann and A. Wilbik. Towards federated clustering: A federated fuzzy c-means algorithm (FFCM). *arXiv preprint arXiv:2201.07316*, 2022.

[SZ19] A.M. Sampat and V.M. Zavala. Fairness measures for decision-making and conflict resolution. *Optimization and Engineering*, 20(4):1249–1272, 2019.

[TER09] S. Tosserams, L.F.P. Etman, and J.E. Rooda. A classification of methods for distributed system optimization based on formulation structure. *Structural and Multidisciplinary Optimization*, 39(5):503–517, 2009.

[TLR12] K.I. Tsianos, S. Lawlor, and M.G. Rabbat. Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning. In *50th Annual Allerton Conference on Communication, Control, and Computing*, pages 1543–1550. IEEE, 2012.

[TYC+22] A.Z. Tan, H. Yu, L. Cui, and Q. Yang. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[ULG+20] C.A. Uribe, S. Lee, A. Gasnikov, and A. Nedić. A dual approach for optimal algorithms in distributed optimization over networks. In *Information Theory and Applications Workshop (ITA)*, pages 1–37. IEEE, 2020.

[vdHG03] S.A. van den Heever and I.E. Grossmann. A strategy for the integration of production planning and reactive scheduling in the optimization of a hydrogen supply network. *Computers & Chemical Engineering*, 27(12):1813–1839, 2003.

[VEG+16] R. Vujanic, P.M. Esfahani, P.J. Goulart, S. Mariéthoz, and M. Morari. A decomposition method for large scale MILPs, with performance guarantees and a power system application. *Automatica*, 67:144–156, 2016.

[VWK+20] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J.S. Rellermeyer. A survey on distributed machine learning. *ACM Computing Surveys*, 53(2):1–33, 2020.

[Wal87] D.A. Walker. Walras's theories of tatonnement. *Journal of Political Economy*, (95 (4)):758–774, 1987.

[WB06] 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, 2006.

[WE19] S. Wenzel and S. Engell. Coordination of coupled systems of systems with quadratic approximation. *IFAC-PapersOnLine*, 52(3):132–137, 2019.

[Wen20] S. Wenzel. *Distributed Optimization of Coupled Production Systems Via Market-like Coordination*. Shaker Verlag, 2020.

[WGE15] S. Wenzel, W. Gao, and S. Engell. Handling disturbances in modifier adaptation with quadratic approximation. *IFAC-PapersOnLine*, 48(25):132–137, 2015.

[WJG+22] Y. Wang, M. Jia, N. Gao, L. Von Krannichfeldt, M. Sun, and G. Hug. Federated clustering for electricity consumption pattern extraction. *IEEE Transactions on Smart Grid*, 13(3):2425–2439, 2022.

[WL01] S.L. Wang and L.Z. Liao. Decomposition method with a variable parameter for a class of monotone variational inequality problems. *Journal of Optimization Theory and Applications*, 109(2):415–429, 2001.

[WPK+16] S. Wenzel, R. Paulen, S. Krämer, B. Beisheim, and S. Engell. Price adjustment in price-based coordination using quadratic approximation. In *Computer Aided Chemical Engineering*, volume 38, pages 193–198. Elsevier, 2016.

[WPS+16] S. Wenzel, R. Paulen, G. Stojanovski, S. Krämer, B. Beisheim, and S. Engell. Optimal resource allocation in industrial complexes by distributed optimization and dynamic pricing. *at-Automatisierungstechnik*, 64(6):428–442, 2016.

[WRE20] S. Wenzel, F. Riedl, and S. Engell. An efficient hierarchical market-like coordination algorithm for coupled production systems based on quadratic approximation. *Computers & Chemical Engineering*, 134(8):106704, 2020.

[WYG17] S. Wenzel, V. Yfantis, and W. Gao. Comparison of regression data selection strategies for quadratic approximation in rto. In *Computer Aided Chemical Engineering*, volume 40, pages 1711–1716. Elsevier, 2017.

[WYW13] Z. Wang, K. Yang, and X. Wang. Privacy-preserving energy scheduling in microgrid systems. *IEEE Transactions on Smart Grid*, 4(4):1810–1820, 2013.

[WZD+15] Q. Wang, C. Zhang, Y. Ding, G. Xydis, J. Wang, and J. Østergaard. Review of real-time electricity markets for integrating distributed energy resources and demand response. *Applied Energy*, 138:695–706, 2015.

[XGS+21] J. Xu, B.S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang. Federated learning for healthcare informatics. *Journal of Healthcare Informatics Research*, 5(1):1–19, 2021.

[YGW+22] V. Yfantis, N. Gafur, A. Wagner, and M. Ruskowski. Hierarchical distributed model predictive control based on dual decomposition and quadratic approximation. In *30th Mediterranean Conference on Control and Automation (MED)*, pages 914–919. IEEE, 2022.

[YJ10] B. Yang and M. Johansson. Distributed optimization and games: A tutorial overview. In *Networked Control Systems*, volume 406 of *Lecture Notes in Control and Information Sciences*, pages 109–148. Springer, London, 2010.

[YMB+22] V. Yfantis, W. Motsch, N. Bach, A. Wagner, and M. Ruskowski. Optimal load control and scheduling through distributed mixed-integer linear programming. In *30th Mediterranean Conference on Control and Automation (MED)*, pages 920–926. IEEE, 2022.

[YR22] V. Yfantis and M. Ruskowski. A hierarchical dual decomposition-based distributed optimization algorithm combining quasi-Newton steps and bundle methods. In *30th Mediterranean Conference on Control and Automation (MED)*, pages 31–36. IEEE, 2022.

[YTY+17] J. Yin, T. Tang, L. Yang, J. Xun, Y. Huang, and Z. Gao. Research and development of automatic train operation for railway transportation systems: A survey. *Transportation Research Part C: Emerging Technologies*, 85:548–572, 2017.

[YWR23] V. Yfantis, A. Wagner, and M. Ruskowski. Federated k-means clustering via dual decomposition-based distributed optimization. *arXiv preprint arXiv:2307.13267*, 2023.

[YWR24] V. Yfantis, A. Wagner, and M. Ruskowski. Numerical benchmarking of dual decomposition-based optimization algorithms for distributed model predictive control. *Results in Control and Optimization (submitted)*, 2024.

[YWW+23] V. Yfantis, S. Wenzel, A. Wagner, M. Ruskowski, and S. Engell. Hierarchical distributed optimization of constraint-coupled convex and mixed-integer programs using approximations of the dual function. *EURO Journal on Computational Optimization*, 11:100058, 2023.

[YYW+19] 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.

[ZG16] Q. Zhang and I.E. Grossmann. Enterprise-wide optimization for indus-

trial demand side management: Fundamentals, advances, and perspectives. *Chemical Engineering Research and Design*, 116:114–131, 2016.

[ZGG12] Y. Zhang, N. Gatsis, and G.B. Giannakis. Robust distributed energy management for microgrids with renewables. In *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 510–515. IEEE, 2012.

[ZGG13] Y. Zhang, N. Gatsis, and G.B. Giannakis. Disaggregated bundle methods for distributed market clearing in power networks. In *IEEE Global Conference on Signal and Information Processing*, pages 835–838. IEEE, 2013.

[ZLG+15] Z.-H. Zhan, X.-F. Liu, Y.-J. Gong, J. Zhang, H.S.-H. Chung, and Y. Li. Cloud computing resource scheduling and a survey of its evolutionary approaches. *ACM Computing Surveys*, 47(4):1–33, 2015.

[ZLL+16] Y. Zheng, S.E. Li, K. Li, F. Borrelli, and J.K. Hedrick. Distributed model predictive control for heterogeneous vehicle platoons under unidirectional topologies. *IEEE Transactions on Control Systems Technology*, 25(3):899–910, 2016.

[ZLQ12] Y. Zheng, S. Li, and H. Qiu. Networked coordination-based distributed model predictive control for large-scale system. *IEEE Transactions on Control Systems Technology*, 21(3):991–998, 2012.

[ZRO+14] M. Zargham, A. Ribeiro, A. Ozdaglar, and A. Jadbabaie. Accelerated dual descent for network flow optimization. *IEEE Transactions on Automatic Control*, 59(4):905–920, 2014.

[ZTL+11] Z. Zhu, J. Tang, S. Lambotharan, W.H. Chin, and Z. Fan. An integer linear programming and game theory based optimization for demand-side management in smart grid. In *IEEE GLOBECOM Workshops*, pages 1205–1210. IEEE, 2011.

Lebenslauf

Persönliche Daten

Name:	Vassilios Yfantis
Geburtsort:	Düsseldorf
Staatsangehörigkeit:	Deutsch/Griechisch

Ausbildung

2016 - 2018	Technische Universität Dortmund Studiengang: Chemieingenieurwesen (M.Sc.)
2012 - 2015	Technische Universität Berlin Studiengang: Chemieingenieurwesen (B.Sc.)

Berufserfahrung

seit 2023	Gurobi GmbH Technical Account Manager
2019 - 2023	Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau *Wissenschaftlicher Mitarbeiter*
2018 - 2019	Technische Universität Dortmund *Wissenschaftlicher Mitarbeiter*