EURO Journal on Computational Optimization 11 (2023) 100058

Contents lists available at ScienceDirect

EURO Journal on Computational Optimization

journal homepage: www.elsevier.com/locate/ejco

Hierarchical distributed optimization of constraint-coupled convex and mixed-integer programs using approximations of the dual function

Vassilios Yfantis [a,*], Simon Wenzel [b,c], Achim Wagner [d], Martin Ruskowski [a,d], Sebastian Engell [c]

[a] *Chair of Machine Tools and Control Systems, Department of Mechanical and Process Engineering, TU Kaiserslautern, Gottlieb-Daimler-Straße 42, 67663 Kaiserslautern, Germany*
[b] *Evonik Operations GmbH, Paul-Baumann-Str. 1, 45772 Marl, Germany*
[c] *Process Dynamics and Operations Group, Department of Biochemical and Chemical Engineering, TU Dortmund University, Emil-Figge-Str. 70, 44227 Dortmund, Germany*
[d] *Innovative Factory Systems, German Research Center for Artificial Intelligence, Trippstadter Str. 122, 67663 Kaiserslautern, Germany*

ARTICLE INFO

Keywords:
Distributed optimization
Dual decomposition
Nonsmooth optimization
Subgradient method
Bundle method
ADMM
Quadratic approximation
Quasi-Newton

ABSTRACT

In this paper, two new algorithms for dual decomposition-based distributed optimization are presented. Both algorithms rely on the quadratic approximation of the dual function of the primal optimization problem. The dual variables are updated in each iteration through a maximization of the approximated dual function. The first algorithm approximates the dual function by solving a regression problem, based on the values of the dual function collected from previous iterations. The second algorithm updates the parameters of the quadratic approximation via a quasi-Newton scheme. Both algorithms employ step size constraints for the update of the dual variables. Furthermore, the subgradients from previous iterations are stored in order to construct cutting planes, similar to bundle methods for nonsmooth optimization. However, instead of using the cutting planes to formulate a piece-wise linear over-approximation of the dual function, they are used to construct

* Corresponding author.
E-mail address: vassilios.yfantis@mv.uni-kl.de (V. Yfantis).

https://doi.org/10.1016/j.ejco.2023.100058

valid inequalities for the update step. In order to demonstrate the efficiency of the algorithms, they are evaluated on a large set of constrained quadratic, convex and mixed-integer benchmark problems and compared to the subgradient method, the bundle trust method, the alternating direction method of multipliers and the quadratic approximation coordination algorithm. The results show that the proposed algorithms perform better than the compared algorithms both in terms of the required number of iterations and in the number of solved benchmark problems in most cases.

1. Introduction

Many industrial applications of optimization require the solution of a system-wide problem over a network of agents [48]. 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 [56]. 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 [72]. 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 parameters, etc. [30,41]. This is often the case in industrial complexes where production systems are coupled through interconnected networks of materials and energy [68]. The involved subsystems may not be willing or able to exchange the information required for the centralized solution of a system-wide optimization problems, e.g. because they belong to different business units or to different companies. Another area necessitating the solution of large-scale optimization problems is machine learning [24,58]. In addition to the size of the underlying optimization problems, data sovereignty plays an important role in many machine learning applications [39]. 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 [35].

Distributed optimization methods offer a way to circumvent the aforementioned issues by splitting the aggregated optimization problem into smaller subproblems, 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. Examples of decomposition approaches include game theoretic approaches [71], population-based approaches [3], primal decomposition [53] and dual decomposition [20]. This paper 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 which 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) [71]). In this work, hierarchical algorithms are considered which coordinate the solutions of the subproblems by iteratively adapting and broadcasting the additional variables, here the dual 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 [44] or the residual of the system-wide constraints [69], the optimal objective function values of the subsystems in each iteration, gradients of the subsystems' objective functions and constraints, and the Hessians of the Lagrangians of the subsystems [33]. 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 in a synchronous manner. 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 [11]. Usually less iterations are required if all subproblems are solved in each iteration.

Dual decomposition-based algorithms generally exhibit a slow rate of convergence. This issue was addressed by, e.g., Maxeiner and Engell [44] and Wenzel et al. [69] where efficient use of information from previous iterations was made. In this paper a new algorithm is proposed that uses some of the elements of the quadratic approximation coordination (QAC) algorithm proposed by Wenzel et al. [69]. In contrast to the QAC

algorithm, the new algorithm, quadratically approximated dual ascent (QADA), approximates the dual function of the system-wide optimization problem by a quadratic function in each iteration. This requires to exchange the values of the Lagrangians of the subproblems at each iteration but still maintains privacy of the local constraints and of the contributions to the system-wide constraints. As will be shown below, this improves the rate of convergence for convex problems with real-valued decision variables and in particular leads to an efficient distributed solution of mixed-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 an algorithm that approximates the dual function based on quasi-Newton updates, which is referred to as the quasi-Newton dual ascent (QNDA) algorithm. The algorithm was first introduced in [73] and is discussed in detail and compared for the benchmark problems in this paper.

The remainder of this paper is organized as follows: Section 2 presents the main mathematical notation. In Section 3 the concepts of duality and dual decomposition for constraint-coupled optimization problems are introduced. Several distributed optimization algorithms which will serve as references for the new proposed approaches are discussed in Section 4. 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. Section 5 discusses algorithms that update the dual variables based on an optimization of a smooth surrogate function. This includes the QAC algorithm introduced in [69] as well as the new proposed algorithms, QADA and QNDA, which compute a quadratic surrogate function of the dual function. The convergence properties of these algorithms are discussed at the end of this section for different classes of problems in a semi-formal manner, based on known results for the same type of problems and similar algorithms. The performance of the new algorithms, in comparison to known approaches is evaluated for a large set of benchmark problems for different problem classes in Section 6. The paper concludes with a summary and an outlook on future research in Section 7.

2. Notation

We use boldfaced upper-case letters 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. 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 relative interior of a set $\mathcal{X}$ is denoted by relint($\mathcal{X}$). The notation $\mathbf{x}^*$ indicates the optimum of an optimization problem.

3. Duality and dual decomposition

In this work we consider optimization problems of the form

$$\min_{\mathbf{x}_1,\dots,\mathbf{x}_{N_s}} \sum_{i\in\mathcal{I}} f_i(\mathbf{x}_i), \tag{1a}$$

$$\text{s.t.} \sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i \leq \mathbf{b}, \tag{1b}$$

$$\mathbf{x}_i \in \mathcal{X}_i,\ \forall i \in \mathcal{I}. \tag{1c}$$

(1) 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 $f_i\colon \mathbb{K}^{n_{\mathbf{x}_i}} \to \mathbb{K}$. In this paper we consider $\mathbb{K} \in \{\mathbb{R}, \mathbb{R}\times\mathbb{Z}\}$, i.e., continuous or mixed-integer optimization problems. The subsystems are coupled through the system-wide constraints (1b), 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. We assume that the system-wide objective function is additive in the subsystem objective function values. The goal is to minimize the sum of the objective functions of all subproblems (1a), also called a social welfare objective [57], while satisfying the system-wide constraints (1b) as well as the individual constraints (1c).

Problem (1) is obviously 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. Therefore the system-wide or central problem can be decomposed by introducing dual variables $\boldsymbol{\lambda} \in \mathbb{R}^{n_\mathbf{b}}$, also referred to as Lagrange multipliers, for the coupling constraints. With the dual variables, the Lagrange function can be formulated,

$$\mathcal{L}(\mathbf{x},\boldsymbol{\lambda}) := \sum_{i\in\mathcal{I}} f_i(\mathbf{x}_i) + \boldsymbol{\lambda}^T \sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i - \boldsymbol{\lambda}^T\mathbf{b}, \tag{2}$$

where $\mathbf{x} = [\mathbf{x}_1^T,\dots,\mathbf{x}_{N_s}^T]^T$. Based on the Lagrange functions, the dual function

$$d(\boldsymbol{\lambda}) := \inf_{\mathbf{x}_i\in\mathcal{X}_i,\forall i\in\mathcal{I}} \mathcal{L}(\mathbf{x},\boldsymbol{\lambda}) \tag{3}$$

can be defined. 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 Karush-Kuhn-Tucker conditions and

from the fact that the system-wide constraints (1b) are inequalities [51]. Generally, the domain of the dual variables $\boldsymbol{\lambda}$ consists of all values for which the Lagrange function (2) is bounded from below, i.e., all values for which $d(\boldsymbol{\lambda}) > -\infty$.

An important property of the dual function (3) is that its values always provide a lower bound for the objective value of the system-wide problem (1) (cf. [9]).

Since the dual function provides a lower bound for the objective values of the system-wide problem, 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,

$$\max_{\boldsymbol{\lambda} \in \mathbb{R}^{n_\mathbf{b}}} d(\boldsymbol{\lambda}), \tag{4a}$$

$$\text{s. t. } \lambda \geq \mathbf{0}. \tag{4b}$$

In contrast, the system-wide problem (1) is called the primal problem. We denote by $\boldsymbol{\lambda}^*$ the optimal solution of the dual problem (4). Due to the lower bound property of the dual function the relation

$$\sum_{i \in \mathcal{I}} f_i(\mathbf{x}_i^*) \geq d(\boldsymbol{\lambda}^*) \tag{5}$$

holds between the primal and dual optimal solutions. Inequality (5) is referred to as weak duality. The difference between the objective values of a feasible primal and corresponding feasible dual solution is called duality gap,

$$\text{DG} = \sum_{i \in \mathcal{I}} f_i(\mathbf{x}_i) - d(\boldsymbol{\lambda}). \tag{6}$$

If the primal problem, i.e., the system wide problem (1) is convex and a constraint qualification condition is satisfied, the optimal duality gap is zero. This implies, that

$$\sum_{i \in \mathcal{I}} f_i(\mathbf{x}_i^*) = d(\boldsymbol{\lambda}^*). \tag{7}$$

This condition is referred to as strong duality. A commonly used constraint qualification condition is Slater's condition, which is satisfied if a strictly feasible solution of problem (1) exists [9],

$$\exists\, \mathbf{x} \in \text{relint}\,(\mathcal{X}) \cap \{\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}} |,\ \mathbf{A}_i \mathbf{x}_i < \mathbf{b}\}, \tag{8}$$

with

$$\mathcal{X} := \left\{ \mathbf{x} = \left(\mathbf{x}_1^T, \ldots, \mathbf{x}_{N_s}^T\right)^T \mid \mathbf{x}_i \in \mathcal{X}_i,\ \forall i \in \mathcal{I} \right\}. \tag{9}$$

Another important property of the dual problem (4) is that the dual function is always concave, regardless whether or not the primal problem is convex (cf. Theorem 12.10, [51]). Since the dual problem (4) 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 (4). The solution of the dual problem is amendable to distributed computations, since the Lagrange function (2) is separable due to the relaxation of the system-wide constraints (1b). This means that the dual function can be evaluated by solving the individual optimization problems

$$\min_{\mathbf{x}_i \in \mathbb{R}^{n_{\mathbf{x}_i}}} \mathcal{L}_i(\mathbf{x}_i, \boldsymbol{\lambda}), \tag{10a}$$

$$\text{s.t. } \mathbf{x}_i \in \mathcal{X}_i \tag{10b}$$

in a distributed manner for a given value of the dual variables $\boldsymbol{\lambda}$. Satisfaction of the coupling constraints is only achieved upon convergence, contrary to primal decomposition-based methods, where in each iteration a feasible primal solution is computed by imposing (1b) [53,71]. In the case of nonconvex primal problems strong duality (7) does not hold. After convergence a feasible primal solution is usually recovered through the use of problem specific heuristics [11] or by modifying the primal problem a priori [60]. Note that we assume that the lower bound of problem (10)) is attainable, therefore replacing the infimum with the minimum.

Example. Consider the following optimization problem:

$$\min_{x_1, x_2} 0.5x_1^2 + 0.5(x_2 - 1)^2, \tag{11a}$$

$$\text{s.t. } x_1 + x_2 = 0, \tag{11b}$$

$$x_1 \leq 1, \tag{11c}$$

$$x_1, x_2 \in \mathbb{R}. \tag{11d}$$

The Lagrange function of problem (11) is

$$\mathcal{L}(x_1, x_2, \lambda) = 0.5x_1^2 + 0.5(x_2 - 1)^2 + \lambda(x_1 + x_2) \tag{12}$$

and the dual function

$$d(\lambda) = \min_{x_1, x_2} 0.5x_1^2 + 0.5(x_2 - 1)^2 + \lambda(x_1 + x_2), \tag{13a}$$

$$\text{s.t. } x_1 \leq 1, \tag{13b}$$

$$x_1, x_2 \in \mathbb{R}. \tag{13c}$$

Note that since the coupling constraint (11b) is an equality, the domain of the dual variable is $\lambda \in \mathbb{R}$. The value of the dual function now depends on whether or not the individual constraint (11c) is active. Two cases can be distinguished:

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

For a fixed λ the value of the dual function (13) can be computed by setting the gradient of the Lagrange function (12) to zero:

$$\nabla \mathcal{L}(x_1, x_2, \lambda) = \begin{pmatrix} x_1 + \lambda \\ x_2 - 1 + \lambda \end{pmatrix} = \mathbf{0} \Rightarrow x_1 = -\lambda,\ x_2 = 1 - \lambda. \tag{14}$$

Substituting the values of x_1 and x_2 in (12) gives

$$d(\lambda) = -\lambda^2 + \lambda,\ \forall \lambda > -1. \tag{15}$$

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

Setting the gradient of the reduced Lagrange function to zero gives

$$\nabla \mathcal{L}(1, x_2, \lambda) = x_2 - 1 + \lambda = 0 \Rightarrow x_2 = 1 - \lambda. \tag{16}$$

Again, substituting the values for x_1 and x_2 into (12) gives

$$d(\lambda) = -\lambda^2 + 1.5\lambda + 0.5,\ \forall \lambda \leq -1. \tag{17}$$

The dual function for problem (11) is given by

$$d(\lambda) = \begin{cases} -\lambda^2 + \lambda, & \forall \lambda > -1, \\ -\lambda^2 + 1.5\lambda + 0.5, & \forall \lambda \leq -1. \end{cases} \tag{18}$$

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

$$\lim_{\lambda \to -1^+} d(\lambda) = \lim_{\lambda \to -1^-} d(\lambda) = -2. \tag{19}$$

However, the dual function does not have continuous derivatives,

$$\lim_{\lambda \to -1^+} \nabla d(\lambda) = \lim_{\lambda \to -1^+} -2\lambda + 1 = 3, \tag{20a}$$

$$\lim_{\lambda \to -1^-} \nabla d(\lambda) = \lim_{\lambda \to -1^-} -2\lambda + 1.5 = 4.5, \tag{20b}$$

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

$$\max_{\lambda \in \mathbb{R}} d(\lambda) \tag{21}$$

is a convex, but nonsmooth optimization problem. The nonsmoothness is caused by a changing set of active individual constraints. This nonsmoothness is typical for dual optimization problems. Algorithms that aim to solve the nonsmooth dual problem are reviewed in the following section in the context of dual decomposition-based distributed optimization.

4. Review of dual decomposition-based distributed optimization

The general idea of dual decomposition was introduced by Everett [20]. 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 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 [62,69]. In this context the dual variables are called prices or shadow prices [28].

In this section, 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.

4.1. Subgradient method

The simplest distributed optimization algorithm that is based on dual decomposition is the subgradient method. In this algorithm, the dual variables are updated along a subgradient direction of the dual function.

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

$$\phi(\boldsymbol{\chi}) \leq \boldsymbol{\xi}^T(\boldsymbol{\chi} - \boldsymbol{\chi}_0) + \phi(\boldsymbol{\chi}_0) \tag{22}$$

holds for all $\boldsymbol{\chi} \in \operatorname{dom} \phi$. The set of all subgradients comprises the subdifferential $\partial\phi(\boldsymbol{\chi}_0)$ of the function $\phi(\boldsymbol{\chi})$ at the point $\boldsymbol{\chi}_0$ [2]. The subgradient is a generalization of the gradient for nonsmooth (non-differentiable) convex functions. Note that (22) 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. 1 illustrates different subgradients both for differentiable (χ_0) and non-differentiable (χ_0') points. In the subgradient method for distributed optimiza-

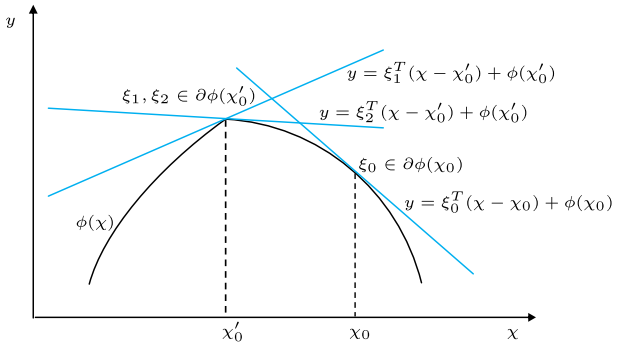


Fig. 1. Geometric interpretation of subgradients for differentiable (χ_0) and non-differentiable (χ_0') points.

tion, the primal variables $\mathbf{x}_i$ and the dual variables $\boldsymbol{\lambda}$ are updated according to

$$\forall i \in \mathcal{I}, \mathbf{x}_i^{(t+1)} = \arg\min_{\mathbf{x}_i \in \mathcal{X}_i} \mathcal{L}_i(\mathbf{x}_i, \boldsymbol{\lambda}^{(t)}) \tag{23a}$$

$$\boldsymbol{\lambda}^{(t+1)} = \left[\boldsymbol{\lambda}^{(t)} + \alpha^{(t)} \mathbf{g}(\boldsymbol{\lambda}^{(t)})\right]^+ \tag{23b}$$

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 (23a) 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 $\mathbf{x}^{(t+1)}$ [71], i.e.,

$$\mathbf{g}(\boldsymbol{\lambda}^{(t)}) := \left(\sum_{i \in \mathcal{I}} \mathbf{A}_i \mathbf{x}_i^{(t+1)} - \mathbf{b}\right) \in \partial d(\boldsymbol{\lambda}^{(t)}). \tag{24}$$

The update step (23b) 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 by means of the Lipschitz constant of the gradient of the dual function [49]. However, this information is usually not available in a distributed optimization setting. For practical applications, the step size is adapted over the course of the iterations [4].

In this paper, we use as the termination criterion 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, i.e.

$$\left(\|\mathbf{w}_p^{(t)}\|_2 \leq \epsilon_p \wedge \|\mathbf{w}_d^{(t)}\|_2 \leq \epsilon_d\right) \vee (t \geq t_{\max}). \tag{25}$$

The primal residual indicates feasibility of the system-wide constraints. If these constraints (1b) are posed as inequalities, the primal residual is defined as

$$[\mathbf{w}_p^{(t)}]_l := \max\left\{\left[\sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i^{(t)} - b\right]_l, 0\right\}, \quad l = 1,\ldots,n_{\mathbf{b}}. \tag{26}$$

If they are posed as equalities it is defined as

$$\mathbf{w}_p^{(t)} := \sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i^{(t)} - \mathbf{b}, \tag{27}$$

i.e., equal to the subgradient. We also use the norm of the primal residual to compute the step size parameter, as proposed by [69],

$$\alpha^{(t)} = \frac{\alpha^{(0)}}{\max\{\|\mathbf{w}_p^{(0)}\|_2,\ldots,\|\mathbf{w}_p^{(t)}\|_2\}}. \tag{28}$$

The dual residual indicates convergence of the dual variables to a stationary value and is defined as

$$\mathbf{w}_d^{(t)} := \boldsymbol{\lambda}^{(t+1)} - \boldsymbol{\lambda}^{(t)}. \tag{29}$$

Algorithm 1 summarizes the subgradient method (SG). Note that the 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, $t_{\max}$
1: $t \leftarrow 0$
2: **repeat**
3: $t \leftarrow t+1$
4: Send $\boldsymbol{\lambda}^{(t)}$ to all subproblems
5: **for all** $i = 1,\ldots,N_s$ **do**
6: $\mathbf{x}_i^{(t+1)} \leftarrow \arg\min_{\mathbf{x}_i\in\mathcal{X}_i} \mathcal{L}_i(\mathbf{x}_i, \boldsymbol{\lambda}^{(t)})$
7: Send $\mathbf{A}_i\mathbf{x}_i^{(t+1)}$ to the coordinator
8: **end for**
9: $\mathbf{g}(\boldsymbol{\lambda}^{(t)}) \leftarrow \sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i^{(t+1)} - \mathbf{b}$
10: **if** Constraints (1b) are inequalities **then**
11: **for all** $l = 1,\ldots,n_{\mathbf{b}}$ **do**
12: $[\mathbf{w}_p^{(t)}]_l \leftarrow \max\left\{\left[\mathbf{g}(\boldsymbol{\lambda}^{(t)})\right]_l, 0\right\}$
13: **end for**
14: **else if** Constraints (1b) are equalities **then**
15: $\mathbf{w}_p^{(t)} \leftarrow \mathbf{g}(\boldsymbol{\lambda}^{(t)})$
16: **end if**
17: $\alpha^{(t)} \leftarrow \alpha^{(0)} / \max\{\|\mathbf{w}_p^{(0)}\|_2,\ldots,\|\mathbf{w}_p^{(t)}\|_2\}$
18: **if** Constraints (1b) are inequalities **then**
19: $\boldsymbol{\lambda}^{(t+1)} \leftarrow [\boldsymbol{\lambda}^{(t)} + \alpha^{(t)}\mathbf{g}(\boldsymbol{\lambda}^{(t)})]^+$
20: **else if** Constraints (1b) are equalities **then**
21: $\boldsymbol{\lambda}^{(t+1)} \leftarrow \boldsymbol{\lambda}^{(t)} + \alpha^{(t)}\mathbf{g}(\boldsymbol{\lambda}^{(t)})$
22: **end if**
23: $\mathbf{w}_d^{(t)} \leftarrow \boldsymbol{\lambda}^{(t+1)} - \boldsymbol{\lambda}^{(t)}$
24: **until** $(\|\mathbf{w}_p^{(t)}\|_2 \le \epsilon_p \wedge \|\mathbf{w}_d^{(t)}\|_2 \le \epsilon_d) \vee (t \ge t_{\max})$
25: **return** $\boldsymbol{\lambda}^{(t)}$

4.2. Bundle trust method

The subgradient method descibed in Sec. 4.1 only employs the subgradient of the previous iteration in order 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 are bundle methods [43]. Bundle methods are among the most efficient algorithms for nonsmooth optimization [2]. 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 [55] or for the coordination of energy networks [75]. 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 [37]. In the following, the bundle trust method (BTM) according to [2], as described in [73], is presented.

The idea of bundle methods is to employ subgradient information collected from multiple previous iterations in order 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

$$\mathcal{B}^{(t)} = \{(\boldsymbol{\lambda}^{(j)}, d(\boldsymbol{\lambda}^{(j)}), \mathbf{g}(\boldsymbol{\lambda}^{(j)})) \in \mathbb{R}^{n_\mathfrak{b}} \times \mathbb{R} \times \mathbb{R}^{n_\mathfrak{b}} | \ 1 \leq j \leq t\} \tag{30}$$

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

$$\hat{d}^{(t)}(\boldsymbol{\lambda}) := \min_{j \in \mathcal{J}^{(t)}} \{d(\boldsymbol{\lambda}^{(j)}) + \mathbf{g}^T(\boldsymbol{\lambda}^{(j)})(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(j)})\}, \tag{31}$$

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, we only store the bundle information up to a certain iteration age τ,

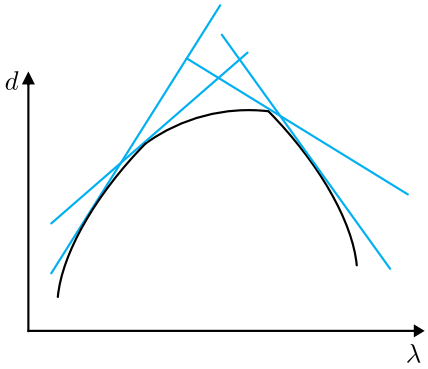
$$\mathcal{J}^{(t)} := \{\max\{1, t - \tau + 1\}, \ldots, t\}. \tag{32}$$

Fig. 2 illustrates the cutting plane model for a nonsmooth dual function. The approximation can be written in an equivalent form as

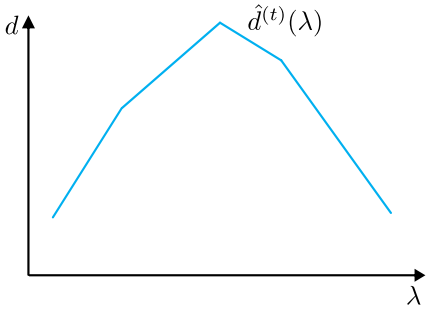
$$\hat{d}^{(t)}(\boldsymbol{\lambda}) = \min_{j \in \mathcal{J}^{(t)}} \{d(\boldsymbol{\lambda}^{(t)}) + \mathbf{g}^T(\boldsymbol{\lambda}^{(j)})(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)}) - \beta^{(j,t)}\}, \tag{33}$$

with the linearization error

$$\beta^{(j,t)} = d(\boldsymbol{\lambda}^{(t)}) - d(\boldsymbol{\lambda}^{(j)}) - \mathbf{g}^T(\boldsymbol{\lambda}^{(j)})(\boldsymbol{\lambda}^{(t)} - \boldsymbol{\lambda}^{(j)}), \ \forall j \in \mathcal{J}^{(t)}. \tag{34}$$



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



(b) Resulting cutting plane model.

Fig. 2. Illustration of the cutting plane model.

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

$$\max_{\mathbf{s}\in\mathbb{R}^{n_\mathbf{b}}} \ \hat{d}^{(t)}(\boldsymbol{\lambda}^{(t)}+\mathbf{s}), \tag{35a}$$

$$\text{s. t. } \|\mathbf{s}\|_2^2 \leq \alpha^{(t)}, \tag{35b}$$

$$\boldsymbol{\lambda}^{(t)}+\mathbf{s} \geq \mathbf{0}. \tag{35c}$$

The constraint (35b) represents a trust region, preventing too aggressive update steps. For the radius of the trust region we use Eq. (28). The constraint (35c) ensures feasibility of the updated dual variables and can be omitted if the system-wide constraints (1b) are equalities. It replaces the projection onto the feasible set used in (23b). Problem (35) is still nonsmooth and can be rewritten as a smooth quadratic direction finding problem

$$\max_{v\in\mathbb{R},\ \mathbf{s}\in\mathbb{R}^{n_\mathbf{b}}} \ v, \tag{36a}$$

$$\text{s. t. } \|\mathbf{s}\|_2^2 \leq \alpha^{(t)}, \tag{36b}$$

$$\mathbf{g}^T(\boldsymbol{\lambda}^{(j)})\mathbf{s} - \beta^{(j,t)} \geq v, \ \forall j \in \mathcal{J}^{(t)}, \tag{36c}$$

$$\boldsymbol{\lambda}^{(t)} + \mathbf{s} \geq \mathbf{0}. \tag{36d}$$

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

$$\forall i \in \mathcal{I},\ \mathbf{x}_i^{(t+1)} = \arg\min_{\mathbf{x}_i \in \mathcal{X}_i} \mathcal{L}_i(\mathbf{x}_i, \boldsymbol{\lambda}^{(t)}), \tag{37a}$$

$$\mathbf{s}^{(t)} = \arg\max_{v \in \mathbb{R},\ \mathbf{s} \in \mathbb{R}^{n_\mathbf{b}}} v, \tag{37b}$$

$$\text{s.t. (36b)-(36d)},$$

$$\boldsymbol{\lambda}^{(t+1)} = \boldsymbol{\lambda}^{(t)} + \mathbf{s}^{(t)}. \tag{37c}$$

Bundle methods often employ a null step, i.e., $\boldsymbol{\lambda}^{(t+1)} = \boldsymbol{\lambda}^{(t)}$, in order 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 (37c) 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

$$d_i(\boldsymbol{\lambda}^{(t)}) = f_i(\mathbf{x}_i^{(t+1)}) + \boldsymbol{\lambda}^{(t),T} \mathbf{g}_i(\boldsymbol{\lambda}^{(t)}) = \mathcal{L}_i(\mathbf{x}_i^{(t+1)}, \boldsymbol{\lambda}^{(t)}) \tag{38}$$

to the coordinator. The termination criteria are identical to the subgradient method. In this paper an aggregated bundle method is considered, i.e., the coordinator aggregates the contributions of all subsystems to the dual function and the subgradient. Algorithm 2 summarizes the bundle trust method. Again note that steps 6–10 are performed in parallel for the different subproblems, while steps 11–29 are performed by the coordinator.

4.3. Alternating direction method of multipliers

Another approach to solve the nonsmooth dual problem is to smoothen the problem by further convexifying the Lagrange function. This is used in augmented Lagrangian methods [1]. 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 in [23] and [22] 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. [8]. In its standard form the ADMM algorithm solves problems of the form

$$\min_{\mathbf{x}_1 \in \mathbb{R}^{n_{\mathbf{x}_1}}, \mathbf{x}_2 \in \mathbb{R}^{n_{\mathbf{x}_2}}} f_1(\mathbf{x}_1) + f_1(\mathbf{x}_2) \tag{39a}$$

$$\text{s.t. } \mathbf{A}_1\mathbf{x}_1 + \mathbf{A}_2\mathbf{x}_2 = \mathbf{b}, \tag{39b}$$

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 (1b) 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 (1b) are equalities **then**
    $\mathbf{w}_p^{(t)} \leftarrow \mathbf{g}(\boldsymbol{\lambda}^{(t)})$
  end if
  $\alpha^{(t)} \leftarrow \alpha^{(0)} / \max\{\|\mathbf{w}_p^{(0)}\|_2, \ldots, \|\mathbf{w}_p^{(t)}\|_2\}$
  $\mathcal{J}^{(t)} \leftarrow \{\max\{1, t-\tau+1\}, \ldots, t\}$
  if Constraints (1b) are inequalities **then**
    $\mathbf{s}^{(t)} \leftarrow \arg\max_{v \in \mathbb{R},\ \mathbf{s} \in \mathbb{R}^{n_{\mathbf{b}}}} v$, s. t. (36b)-(36d)
  else if Constraints (1b) are equalities **then**
    $\mathbf{s}^{(t)} \leftarrow \arg\max_{v \in \mathbb{R},\ \mathbf{s} \in \mathbb{R}^{n_{\mathbf{b}}}} v$, s. t. (36b), (36c)
  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 $(\|\mathbf{w}_p^{(t)}\|_2 \leq \epsilon_p \wedge \|\mathbf{w}_d^{(t)}\|_2 \leq \epsilon_d) \vee (t \geq t_{\max})$
return $\boldsymbol{\lambda}^{(t)}$

by defining an augmented Lagrange function

$$\hat{\mathcal{L}}_\rho(\mathbf{x}_1, \mathbf{x}_2, \boldsymbol{\lambda}) := f_1(\mathbf{x}_1) + f_1(\mathbf{x}_2) + \boldsymbol{\lambda}^T(\mathbf{A}_1\mathbf{x}_1 + \mathbf{A}_2\mathbf{x}_2 - \mathbf{b}) + \frac{\rho}{2}\|\mathbf{A}_1\mathbf{x}_1 + \mathbf{A}_2\mathbf{x}_2 - \mathbf{b}\|_2. \tag{40}$$

The primal variables are then updated in an alternating manner according to

$$\mathbf{x}_1^{(t+1)} = \arg\min_{\mathbf{x}_1 \in \mathbb{R}^{n_{\mathbf{x}_1}}} \hat{\mathcal{L}}_\rho(\mathbf{x}_1, \mathbf{x}_2^{(t)}, \boldsymbol{\lambda}^{(t)}), \tag{41a}$$

$$\mathbf{x}_2^{(t+1)} = \arg\min_{\mathbf{x}_2 \in \mathbb{R}^{n_{\mathbf{x}_2}}} \hat{\mathcal{L}}_\rho(\mathbf{x}_1^{(t+1)}, \mathbf{x}_2, \boldsymbol{\lambda}^{(t)}), \tag{41b}$$

$$\boldsymbol{\lambda}^{(t+1)} = \boldsymbol{\lambda}^{(t)} + \rho(\mathbf{A}_1\mathbf{x}_1^{(t+1)} + \mathbf{A}_2\mathbf{x}_2^{(t+1)} - \mathbf{b}). \tag{41c}$$

The update of the primal variables (41a) and (41b) cannot be performed in parallel. In this paper we consider problems were multiple subsystems are connected through shared limited resources (1). This case suits the use of the optimal exchange version of

ADMM, as described in [8] (Sec. 7.3.2) and [64] (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. Using the auxiliary variables, problem (1) can be reformulated,

$$\min_{\mathbf{x}_1,\ldots,\mathbf{x}_{N_s}} \sum_{i\in\mathcal{I}} f_i(\mathbf{x}_i), \tag{42a}$$

$$\text{s. t. } \mathbf{A}_i\mathbf{x}_i \leq \mathbf{z}_i,\ \forall i \in \mathcal{I}, \tag{42b}$$

$$\sum_{i\in\mathcal{I}} \mathbf{z}_i = \mathbf{b}, \tag{42c}$$

$$\mathbf{x}_i \in \mathcal{X}_i,\ \forall i \in \mathcal{I}. \tag{42d}$$

The individual augmented Lagrange functions for problem (42) are defined as,

$$\hat{\mathcal{L}}_{i,\rho}(\mathbf{x}_i, \boldsymbol{\lambda}, \mathbf{z}_i) := f_i(\mathbf{x}_i) + \lambda^T(\mathbf{A}_i\mathbf{x}_i - \mathbf{z}_i) + \frac{\rho}{2}\left\|\mathbf{A}_i\mathbf{x}_i - \mathbf{z}_i\right\|_2^2. \tag{43}$$

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

$$\forall i \in \mathcal{I},\ \mathbf{x}_i^{(t+1)} = \arg\min_{\mathbf{x}_i\in\mathcal{X}_i} \hat{\mathcal{L}}_{i,\rho}(\mathbf{x}_i, \boldsymbol{\lambda}^{(t)}, \mathbf{z}_i^{(t)}) \tag{44a}$$

$$\forall i \in \mathcal{I},\ \mathbf{z}_i^{(t+1)} = \mathbf{A}_i\mathbf{x}_i^{(t+1)} - \text{ave}(\mathbf{A}\mathbf{x}^{(t+1)} - \mathbf{b}) \tag{44b}$$

$$\boldsymbol{\lambda}^{(t+1)} = [\boldsymbol{\lambda}^{(t)} + \rho^{(t)}\text{ave}(\mathbf{A}\mathbf{x}^{(t+1)} - \mathbf{b})]^+ \tag{44c}$$

where

$$\text{ave}(\mathbf{A}\mathbf{x}^{(t+1)} - \mathbf{b}) \colon \frac{1}{N_s}\sum_{i\in\mathcal{I}}\left(\mathbf{A}_i\mathbf{x}_i^{(t+1)} - \mathbf{b}\right) \tag{45}$$

denotes the average of the system-wide constraints. Note that the update of the primal variables (44a) can now be performed in parallel. The update step of the auxiliary variables (44b) ensures the satisfaction of (42c) [64],

$$\sum_{i\in\mathcal{I}} \mathbf{z}_i^{(t+1)} = \sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i^{(t+1)} - \sum_{i\in\mathcal{I}} \text{ave}(\mathbf{A}\mathbf{x}^{(t+1)} - \mathbf{b}) \tag{46a}$$

$$= \sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i^{(t+1)} - N_s\text{ave}(\mathbf{A}\mathbf{x}^{(t+1)} - \mathbf{b}) \tag{46b}$$

$$= \sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i^{(t+1)} - \left(\sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i^{(t+1)} - \mathbf{b}\right) \tag{46c}$$

$$= \mathbf{b}. \tag{46d}$$

ADMM can be interpreted as a proximal algorithm. The (scaled) proximal operator is defined as [54]

$$\mathbf{x} = \text{prox}_{\rho,f}(\boldsymbol{\chi}) := \arg\min_{\mathbf{y}} f(\mathbf{y}) + \frac{1}{2\rho}\|\mathbf{y} - \boldsymbol{\chi}\|_2^2. \tag{47}$$

For a better interpretation of ADMM, Wenzel [64] defined a modified scaled proximal operator as

$$\mathbf{x} = \text{prox}'_{\rho,f}(\boldsymbol{\chi}) := \arg\min_{\mathbf{y}} f(\mathbf{y}) + \frac{1}{2\rho}\|\mathbf{L}(\mathbf{y}) - \boldsymbol{\chi}\|_2^2, \tag{48}$$

which differs only in the first term of the regularization term, where a linear mapping $\mathbf{L}(\mathbf{y})$ is used. Using the definition (48) and omitting the constant term $-\boldsymbol{\lambda}^{(t),T}\mathbf{z}_i$ the update of the primal variables (44a) can be rewritten as

$$\begin{aligned}
\forall i \in \mathcal{I},\ \mathbf{x}_i^{(t+1)} &= \arg\min_{\mathbf{x}_i \in \mathcal{X}_i} f(\mathbf{x}_i) + \lambda^{(\mathbf{t}),\mathbf{T}}\mathbf{A}_i\mathbf{x}_i && \text{(49a)}\\
&\quad + \frac{\rho}{2}\|\mathbf{A}_i\mathbf{x}_i - \mathbf{z}_i^{(t)}\|_2^2 \\
&= \arg\min_{\mathbf{x}_i \in \mathcal{X}_i} \mathcal{L}_i(\mathbf{x}_i, \boldsymbol{\lambda}^{(t)}) + \frac{\rho}{2}\|\mathbf{A}_i\mathbf{x}_i - \mathbf{z}_i^{(t)}\|_2^2 && \text{(49b)}\\
&= \text{prox}'_{1/\rho,\mathcal{L}}(\mathbf{z}_i^{(t)}). && \text{(49c)}
\end{aligned}$$

Using eq. (49) 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 [36]. It converges under milder assumptions than the subgradient method for convex primal problems [8]. While convergence can be proven for constant values of the regularization parameter ρ, a variation of the parameter over the course of the iterations works well in practice. In this work, the adaptation strategy reported in [31] and [63] is employed:

$$\rho^{(t+1)} = \begin{cases} \tau^{\text{incr}}\rho^{(t)}, & \text{if } \|\mathbf{w}_p^{(t)}\|_2 > \mu\|\mathbf{w}_d^{(t)}\|_2 \\ \rho^{(t)}/\tau^{\text{decr}}, & \text{if } \|\mathbf{w}_d^{(t)}\|_2 > \mu\|\mathbf{w}_p^{(t)}\|_2 \\ \rho^{(t)}, & \text{otherwise.} \end{cases} \tag{50}$$

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

$$\mathbf{w}_d^{(t)} := \mathbf{z}^{(t+1)} - \mathbf{z}^{(t)}, \tag{51}$$

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 in order to update the auxiliary variables (44b), 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.

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, $t_{\max}$
1: $t \leftarrow 0$
2: **repeat**
3: $t \leftarrow t + 1$
4: Send $\boldsymbol{\lambda}^{(t)}$, $\mathbf{z}_i^{(t)}$ and $\rho^{(t)}$ to all subproblems
5: **for all** $i = 1, \ldots, N_s$ **do**
6: $\mathbf{x}_i^{(t+1)} \leftarrow \text{prox}'_{1/\rho, \mathcal{L}}(\mathbf{z}_i^{(t)})$
7: Send $\mathbf{A}_i \mathbf{x}_i^{(t+1)}$ to the coordinator
8: **end for**
9: **for all** $i = 1, \ldots, N_s$ **do**
10: $\mathbf{z}_i^{(t+1)} \leftarrow \mathbf{A}_i \mathbf{x}_i^{(t+1)} - \text{ave}(\mathbf{A}\mathbf{x}^{(t+1)} - \mathbf{b})$
11: **end for**
12: **if** Constraints (1b) are inequalities **then**
13: **for all** $l = 1, \ldots, n_{\mathbf{b}}$ **do**
14: $[\mathbf{w}_p^{(t)}]_l \leftarrow \max\left\{\left[\sum_{i \in \mathcal{I}} \mathbf{A}_i \mathbf{x}_i^{(t+1)} - b\right]_l, 0\right\}$
15: **end for**
16: **else if** Constraints (1b) are equalities **then**
17: $\mathbf{w}_p^{(t)} \leftarrow \sum_{i \in \mathcal{I}} \mathbf{A}_i \mathbf{x}_i^{(t+1)} - b$
18: **end if**
19: **if** Constraints (1b) are inequalities **then**
20: $\boldsymbol{\lambda}^{(t+1)} \leftarrow [\boldsymbol{\lambda}^{(t)} + \rho^{(t)} \text{ave}(\mathbf{A}\mathbf{x}^{(t+1)} - \mathbf{b})]^+$
21: **else if** Constraints (1b) are equalities **then**
22: $\boldsymbol{\lambda}^{(t+1)} \leftarrow \boldsymbol{\lambda}^{(t)} + \rho^{(t)} \text{ave}(\mathbf{A}\mathbf{x}^{(t+1)} - \mathbf{b})]$
23: **end if**
24: $\mathbf{w}_d^{(t)} \leftarrow \mathbf{z}^{(t+1)} - \boldsymbol{z}^{(t)}$
25: $\rho^{(t+1)} \leftarrow \text{Update}(50)$
26: **until** $(\|\mathbf{w}_p^{(t)}\|_2 \leq \epsilon_p \wedge \|\mathbf{w}_d^{(t)}\|_2 \leq \epsilon_d) \vee (t \geq t_{\max})$
27: **return** $\boldsymbol{\lambda}^{(t)}$

4.4. 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 [47] 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 [43]. Bragin et al. [11] 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 which forms an acute angle towards the optimal dual values is used. The algorithm was extended in [10] by introducing an additional absolute value penalty in the objective function of the primal problem (surrogate absolute value Lagrangian relaxation, SAVLR) and in [40] by employing ordinal optimization. These methods were used to solve mixed-integer linear programming (MILP) problems with the main concern being the computational scalability. However, the recovery of a primal feasible solution is 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 on the other hand require 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. [59] adapt the fast gradient method (FGM) proposed by Nesterov [50] 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 [18], where consensus constraints are introduced and subsequently relaxed through dual decomposition. The authors propose a quasi-Newton method which relies on decentralized computations to update the approximated Hessians locally. The same algorithm is further studied in [19], 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. [74] locally approximate the inverse of the Hessian matrices by allowing direct communication between the subproblems. Notarnicola and Notarstefano [52] 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 in [38] to a wider class of functions that still satisfy the strong duality condition. These generalizations are subsequently applied to distributed optimization.

Goldstein et al. [27] extended 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. In order to improve the rate of convergence second-order information can be exploited. Houska at al. [33] extended 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 paper. Chatzipanagiotis et al. [13] 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 [14]. ADMM is also generalized to nonconvex problems in [39], where nonconvex equality constraints are considered. As discussed in Sec. 4.3, 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 [32]. 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 in order to handle constraints [61]. Necoara and Suykens [46] combine dual decomposition with interior point methods by adding self-concordant barrier terms to the Lagrange function.

Maxeiner and Engell [44] 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 towards 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.

5. Algorithms based on smooth approximations

This section 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 $\mathcal{B}^{(t)}$,

$$\psi^{(t)}(\boldsymbol{\lambda}) := \arg \min_{\psi\colon \mathbb{R}^{n_{\mathbf{b}}} \to \mathbb{R}} \sum_{j \in \mathcal{J}^{(t)}} L(\psi(\boldsymbol{\lambda}^{(j)}), \mathcal{B}^{(t)}). \tag{52}$$

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

$$\boldsymbol{\lambda}^{(t+1)} = \arg \min_{\boldsymbol{\lambda} \in \mathcal{M}} \psi^{(t)}(\boldsymbol{\lambda}). \tag{53}$$

First, two algorithms based on the solution of a regression problem are presented. These are the quadratic approximation coordination (QAC) algorithm presented in [69] 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. In the QAC algorithm the squared Euclidean norm of the primal residual $\|\mathbf{w}_p^{(t)}(\boldsymbol{\lambda})\|_2^2$ is approximated by a quadratic function. The QADA algorithm is based on approximations of 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.

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 [73], also approximates the dual function as a quadratic function. However, the approximation of the Hessian is based on Broyden-Fletcher-Goldfarb-Shanno (BFGS) updates.

A discussion of the convergence of the proposed algorithms is provided at the end of the section.

5.1. Regression-based approximations

This section presents the algorithms in which the surrogate function is obtained as the solution of 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. Afterwards, the QAC algorithm is summarized briefly and the QADA algorithm is presented.

5.1.1. Fitting the parameters of a quadratic model

The regression-based algorithms follow the basic idea of derivative-free optimization according to Conn et al. [16], where locally a surrogate, in this case quadratic, model is fitted to previously collected data. To this end, a set of data points,

$$\mathcal{D}^{(t)} = \{(\boldsymbol{\lambda}^{(j)}, \hat{\psi}(\boldsymbol{\lambda}^{(j)}))|\ 1 \leq j \leq t\} \tag{54}$$

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 paper is a quadratic function of the form

$$\begin{aligned} &\psi^{(t)}(\boldsymbol{\lambda}) := \frac{1}{2}\boldsymbol{\lambda}^T \mathbf{Q}^{(t)} \boldsymbol{\lambda} + \mathbf{q}^{(t),T} \boldsymbol{\lambda} + q_0^{(t)}, \\ &\mathbf{Q}^{(t)} \in S\mathbb{R}^{n_\mathfrak{b} \times n_\mathfrak{b}}, \mathbf{q}^{(t)} \in \mathbb{R}^{n_\mathfrak{b}}, q_0^{(t)} \in \mathbb{R}. \end{aligned} \tag{55}$$

The parameters of the quadratic model (55) can be computed as the solution of the regression problem

$$\mathbf{Q}^{(t)}, \mathbf{q}^{(t)}, q_0^{(t)} = \arg\min_{\mathbf{Q},\mathbf{q},q_0} \sum_{j \in \mathcal{J}^{(t)}} \|\psi^{(t)}(\boldsymbol{\lambda}^{(j)}) - \hat{\psi}(\boldsymbol{\lambda}^{(j)})\|_2^2. \tag{56}$$

In order to obtain the solution (56) in a closed form, eq. (55) can be rewritten as

$$\psi^{(t)}(\boldsymbol{\lambda}) = \sum_{l=1}^{n_\mathfrak{b}} \sum_{j=1}^{n_\mathfrak{b}} [\mathbf{Q}^{(t)}]_{l,j} [\boldsymbol{\lambda}]_l [\boldsymbol{\lambda}]_j + \sum_{l=1}^{n_\mathfrak{b}} [\mathbf{q}^{(t)}]_l [\boldsymbol{\lambda}]_l + q_0^{(t)}. \tag{57}$$

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

$$\mathbf{p}^{(t),T} := ([\mathbf{Q}^{(t)}]_{1,1}, \dots, [\mathbf{Q}^{(t)}]_{1,n_{\mathbf{b}}^{(t)}}, [\mathbf{Q}^{(t)}]_{2,2}, \dots, [\mathbf{Q}^{(t)}]_{n_{\mathbf{b}},n_{\mathbf{b}}}, [\mathbf{q}^{(t)}]_1, \dots, [\mathbf{q}^{(t)}]_{n_{\mathbf{b}}}, q_0^{(t)}). \tag{58}$$

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

$$\hat{\boldsymbol{\psi}}^{(t)} := (\hat{\psi}(\boldsymbol{\lambda}^{(1)}), \dots, \hat{\psi}(\boldsymbol{\lambda}^{(n_j^{(t)})}))^T \tag{59}$$

the vector of observed values of the approximated function. Then, by defining the Vandermonde-matrix [64]

$$\mathbf{M}^{(t)} := \left(\mathbf{l}_1^{\circ 2}, \mathbf{l}_1 \circ \mathbf{l}_2, \dots, \mathbf{l}_1 \circ \mathbf{l}_{n_{\mathbf{b}}}, \mathbf{l}_2^{\circ 2}, \dots, \mathbf{l}_{n_{\mathbf{b}}}^{\circ 2}, \mathbf{l}_1 \dots, \mathbf{l}_{n_{\mathbf{b}}}, \mathbf{1}\right), \tag{60}$$

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

$$\mathbf{p}^{(t)} = \left(\mathbf{M}^{(t),T}\mathbf{M}^{(t)}\right)^{-1}\mathbf{M}^{(t)}\hat{\boldsymbol{\psi}}^{(t)}. \tag{61}$$

Note that $\boldsymbol{\lambda}^{(1)}$ in equations (59) and (60) denotes the first dual variables in the regression set (54), not the dual variables in the first iteration of the dual decomposition-based algorithm.

In order to perform a quadratic approximation at least

$$n_{\text{reg,min}} := (n_{\mathbf{b}} + 1)(n_{\mathbf{b}} + 2)/2 \tag{62}$$

data points are necessary, i.e., $n_j^{(t)} \geq n_{\text{reg,min}}$, since $\mathbf{p}^{(t)} \in \mathbb{R}^{n_{\text{reg,min}}}$ [16,64]. This shows that the regression-based 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., $\mathcal{J}^{(t)} \subseteq \{1, \dots, t\}$, plays an important role in the performance of the algorithms and is discussed in the next section.

5.1.2. Regression data selection strategy

The selection of suitable points for the quadratic approximation has been studied extensively in the context of derivative-free optimization [16]. The following criteria are usually considered [26,69]:

- spread,
- distance,
- number of points,
- age.

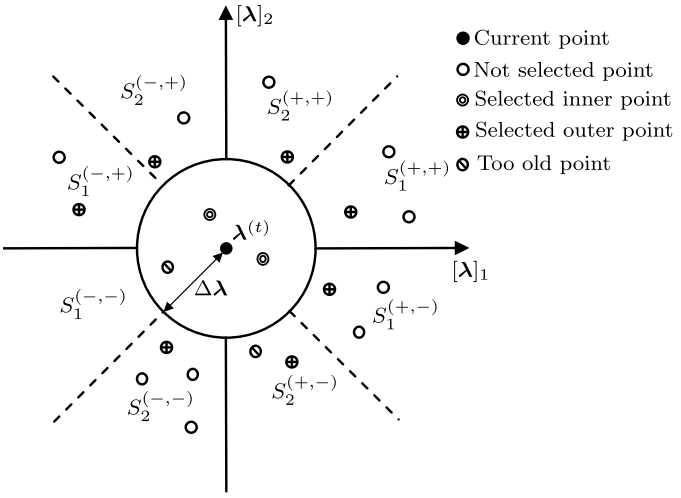


Fig. 3. Illustration of the regression data selection using the nearest axis point separation algorithm.

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 in order to keep the approximation local. A minimum number of points (62) has to be used for the quadratic approximation, but 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 [26,66].

Different algorithms have been proposed for data selection in the context of quadratic approximation. Two different algorithms from Wenzel et al. [66] and Gao et al. [26] were compared in [70]. Throughout this work the nearest axis point separation (NAPS) algorithm from [66] is used as it yields comparable results to the selection algorithm proposed in [26] 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 [64,65,69]. 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 in order to stabilize the approximation. The algorithm is illustrated in Fig. 3 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 $\boldsymbol{\Lambda} := (\boldsymbol{\lambda}^{(0)}, \ldots, \boldsymbol{\lambda}^{(t)})$. These points are divided into inner points $\boldsymbol{\Lambda}_I$ and outer point $\boldsymbol{\Lambda}_O$. A point is classified as an inner point if it 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 $\mathbb{R}^{n_\text{b}}$ is then divided into segments according to their sign configuration

(in reference to $\boldsymbol{\lambda}^{(t)}$) and their nearest axis. For instance, in Fig. 3 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

$$\mathcal{D}^{(t)} = \{(\boldsymbol{\lambda}^{(j)}, \hat{\psi}(\boldsymbol{\lambda}^{(j)}))|\ j \in \mathcal{J}^{(t)}\}, \tag{63}$$

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 [69]).

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.
$\mathcal{J}^{(t)} \leftarrow \mathcal{J}^{(t)} \cup \{j \in \{1, \ldots, t\}|\ \boldsymbol{\lambda}^{(j)} \in \boldsymbol{\Lambda}_I\}$ ▷ 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 $\mathcal{S}_l^{(\cdot)} \in \{\mathcal{S}_1^{(\cdot)}, \ldots, \mathcal{S}_{n_\mathbf{b}}^{(\cdot)}\}$ **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 $\boldsymbol{\Lambda}^{(t)}$, $\hat{\boldsymbol{\psi}}^{(t)}$, $\mathcal{J}^{(t)}$

5.1.3. Covariance-based step size constraint

Wenzel and Engell [65] proposed a covariance-based 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,

$$\mathbf{C}^{(t)} = \text{cov}(\boldsymbol{\Lambda}^{(t)}). \tag{64}$$

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. Wenzel and Engell [65] 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,

$$\mathbf{C}^{(t)} = \mathbf{U}^{(t)}\mathbf{\Sigma}^{(t)}\mathbf{V}^{(t),T},\ \mathbf{\Sigma}^{(t)} = \text{diag}(\sigma_l^{(t)}), \tag{65}$$

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

$$\hat{\sigma}_l^{(t)} := \max\{\underline{s}_l, \min\{\sigma_l^{(t)}, \overline{s}_l\}\}, \tag{66}$$

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,

$$\hat{\mathbf{C}}^{(t)} = \mathbf{U}^{(t)}\hat{\mathbf{\Sigma}}^{(t)}\mathbf{V}^{(t),T},\ \hat{\mathbf{\Sigma}}^{(t)} = \text{diag}(\hat{\sigma}_l^{(t)}). \tag{67}$$

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

$$\mathcal{E}(\mathbf{\Lambda}^{(t)}) := \{\boldsymbol{\lambda} \in \mathbb{R}^{n_{\mathbf{b}}} | (\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)})^T \hat{\mathbf{C}}^{(t),-1}(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)}) \leq (\gamma^{(t)})^2\}. \tag{68}$$

Wenzel et al. [69] propose to update $\gamma^{(t)}$ according to

$$\gamma^{(t)} = \max\{\log \|\mathbf{w}_p(\boldsymbol{\lambda}^{(t)})\|_2, \underline{\gamma}\}, \tag{69}$$

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.

5.1.4. The QAC algorithm

The quadratic approximation coordination (QAC) algorithm was first proposed in [67]. It was motivated by the distributed optimization of quadratic programs (QPs) without local constraints

$$\min_{\mathbf{x}_1,\dots,\mathbf{x}_{N_s}} \sum_{i\in\mathcal{I}} \frac{1}{2}\mathbf{x}_i^T\mathbf{H}_i\mathbf{x}_i + \mathbf{c}_i^T\mathbf{x}_i, \tag{70a}$$

$$\text{s. t.} \sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i = \mathbf{b}, \tag{70b}$$

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

$$\min_x \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{c}^T\mathbf{x} \tag{71a}$$

$$\text{s. t. } \mathbf{A}\mathbf{x} = \mathbf{b} \tag{71b}$$

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 (71) is

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{c}^T\mathbf{x} + \boldsymbol{\lambda}^T(\mathbf{A}\mathbf{x} - \mathbf{b}). \tag{72}$$

Since the matrices $\mathbf{H}_i$ are symmetric and positive definite, i.e., problem (71) 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 [9]:

$$\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) \overset{!}{=} \mathbf{0} \Rightarrow \mathbf{x}^*(\boldsymbol{\lambda}) = -\mathbf{H}^{-1}(\mathbf{c} + \mathbf{A}^T\boldsymbol{\lambda}). \tag{73}$$

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

$$\mathbf{w}_p(\boldsymbol{\lambda}) = \mathbf{A}\mathbf{x}^*(\boldsymbol{\lambda}) - \mathbf{b} = -\mathbf{A}\mathbf{H}^{-1}(\mathbf{c} + \mathbf{A}^T\boldsymbol{\lambda}) - \mathbf{b}. \tag{74}$$

Computing the squared Euclidean norm of the primal residual leads to

$$\begin{aligned}\|\mathbf{w}_p(\boldsymbol{\lambda})\|_2^2 = \frac{1}{2}\boldsymbol{\lambda}^T \underbrace{2\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^T\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^T}_{=:\hat{\mathbf{Q}}}\boldsymbol{\lambda} + \\ \underbrace{2(\mathbf{c}^T\mathbf{H}^{-1}\mathbf{A}^T + \mathbf{b}^T)\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^T}_{=:\hat{\mathbf{q}}^T}\boldsymbol{\lambda} + \\ \underbrace{(\mathbf{c}^T\mathbf{H}^{-1}\mathbf{A}^T + \mathbf{b}^T)(\mathbf{A}\mathbf{H}^{-1}\mathbf{c} + \mathbf{b})}_{=:\hat{q}_0}.\end{aligned} \tag{75}$$

Thus the squared Euclidean norm of the primal residual is a quadratic function of the dual variables [69]

$$\|\mathbf{w}_p(\boldsymbol{\lambda})\|_2^2 = \frac{1}{2}\boldsymbol{\lambda}^T\hat{\mathbf{Q}}\boldsymbol{\lambda} + \hat{\mathbf{q}}^T\boldsymbol{\lambda} + \hat{\mathbf{q}}_0. \tag{76}$$

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

$$\hat{\psi}(\boldsymbol{\lambda}) = \|\mathbf{w}_p(\boldsymbol{\lambda})\|_2^2 = \left\|\sum_{i\in\mathcal{I}}\mathbf{A}_i\mathbf{x}_i(\boldsymbol{\lambda}) - \mathbf{b}\right\|_2^2. \tag{77}$$

The surrogate function is a quadratic function

$$r^{(t)}(\boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{\lambda}^T\mathbf{Q}^{(t)}\boldsymbol{\lambda} + \mathbf{q}^{(t),T}\boldsymbol{\lambda} + q_0^{(t)} \tag{78}$$

with the parameters

$$\mathbf{Q}^{(t)}, \mathbf{q}^{(t)}, q_0^{(t)} = \arg\min_{\mathbf{Q},\mathbf{q},q_0} \sum_{j\in\mathcal{J}^{(t)}} \left\| r^{(t)}(\boldsymbol{\lambda}^{(j)}) - \|\mathbf{w}_p(\boldsymbol{\lambda}^{(j)})\|_2^2 \right\|_2^2. \tag{79}$$

The regression data

$$\mathcal{D}^{(t)} = \{(\boldsymbol{\lambda}^{(j)}, \|\mathbf{w}_p(\boldsymbol{\lambda}^{(j)}))\|_2^2 |\ j \in \mathcal{J}^{(t)}\}, \tag{80}$$

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

$$\boldsymbol{\lambda}^{(t+1)} = \arg\min_{\boldsymbol{\lambda}\in\mathbb{R}^{n_\mathbf{b}}} r^{(t)}(\boldsymbol{\lambda}), \tag{81a}$$

$$\text{s.t. } \boldsymbol{\lambda} \in \mathcal{E}(\boldsymbol{\Lambda}^{(t)}), \tag{81b}$$

$$\boldsymbol{\lambda} \geq \mathbf{0}. \tag{81c}$$

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 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 is able to 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 also be 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 (70). Wenzel et al. [69] 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 [69]. It was shown in the example in Section 3 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

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 (1b) 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 (1b) 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
    $\alpha^{(t)} \leftarrow \alpha^{(0)} / \max\{\|\mathbf{w}_p^{(0)}\|_2, \ldots, \|\mathbf{w}_p^{(t)}\|_2\}$
    if Constraints (1b) are inequalities **then**
      $\boldsymbol{\lambda}^{(t+1)} \leftarrow [\boldsymbol{\lambda}^{(t)} + \alpha^{(t)} \mathbf{g}(\boldsymbol{\lambda}^{(t)})]^+$
    else if Constraints (1b) are equalities **then**
      $\boldsymbol{\lambda}^{(t+1)} \leftarrow \boldsymbol{\lambda}^{(t)} + \alpha^{(t)} \mathbf{g}(\boldsymbol{\lambda}^{(t)})$
    end if
  else ▷ Perform QAC Updates
    $\mathcal{D}^{(t)} \leftarrow \text{NAPS}(\boldsymbol{\Lambda}, \|\mathbf{w}_p(\boldsymbol{\Lambda})\|_2^2, \tau, \Delta\boldsymbol{\lambda})$
    $\mathcal{E}(\boldsymbol{\Lambda}^{(t)}) \leftarrow \text{ComputeEllipsoid}(\boldsymbol{\Lambda}^{(t)}, \underline{s}_i, \overline{s}_i, \underline{\gamma})$
    $r^{(t)}(\boldsymbol{\lambda}) \leftarrow \text{Regression}(\mathcal{D}^{(t)})$
    if Constraints (1b) are inequalities **then**
      $\boldsymbol{\lambda}^{(t+1)} \leftarrow \arg\min_{\boldsymbol{\lambda} \in \mathcal{E}(\boldsymbol{\Lambda}^{(t)})} r^{(t)}(\boldsymbol{\lambda})$, s.t. $\boldsymbol{\lambda} \geq \mathbf{0}$
    else if Constraints (1b) are equalities **then**
      $\boldsymbol{\lambda}^{(t+1)} \leftarrow \arg\min_{\boldsymbol{\lambda} \in \mathcal{E}(\boldsymbol{\Lambda}^{(t)})} r^{(t)}(\boldsymbol{\lambda})$
    end if
  end if
  $\mathbf{w}_d^{(t)} \leftarrow \boldsymbol{\lambda}^{(t+1)} - \boldsymbol{\lambda}^{(t)}$
until $(\|\mathbf{w}_p^{(t)}\|_2 \leq \epsilon_p \wedge \|\mathbf{w}_d^{(t)}\|_2 \leq \epsilon_d) \vee (t \geq t_{\max})$
return $\boldsymbol{\lambda}^{(t)}$

its efficiency. The issue of the changing sets of active constraints was addressed in [69] by employing a fallback strategy, if an insensitivity of the primal residual was detected. However, the numerical experiments described in Section 6 showed that the QAC algorithm actually performed better without the fallback strategy. Therefore its discussion is omitted at this point.

5.1.5. Quadratically approximated dual ascent

As discussed in the previous section, approximating the primal residual as a quadratic function suffers from a number of 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.,

$$\hat{\psi}(\boldsymbol{\lambda}) = d(\boldsymbol{\lambda}) = \min_{\mathbf{x}_i \in \mathcal{X}_i,\ \forall i \in \mathcal{I}} \sum_{i \in \mathcal{I}} \mathcal{L}_i(\mathbf{x}_i, \boldsymbol{\lambda}) - \boldsymbol{\lambda}^T \mathbf{b}, \tag{82}$$

which is always concave. For the special case of distributed QPs (70) it is also easy to show that the dual function is quadratic. By inserting the optimal values of the primal variables $\mathbf{x}^*(\boldsymbol{\lambda})$ (73) into the Lagrange function (72) the dual function computes to

$$\begin{aligned} d(\boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{\lambda}^T \underbrace{(-\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^T)}_{=:\tilde{\mathbf{Q}}}\boldsymbol{\lambda} + \\ \underbrace{(-\mathbf{c}^T\mathbf{H}^{-1}\mathbf{A}^T - \mathbf{b}^T)}_{=:\tilde{\mathbf{q}}^T}\boldsymbol{\lambda} + \\ \underbrace{(-\frac{1}{2}\mathbf{c}^T\mathbf{H}^{-1}\mathbf{c})}_{\tilde{q}_0}. \end{aligned} \tag{83}$$

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

$$d_Q^{(t)}(\boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{\lambda}^T\mathbf{Q}^{(t)}\boldsymbol{\lambda} + \mathbf{q}^{(t),T}\boldsymbol{\lambda} + q_0^{(t)} \tag{84}$$

with the parameters

$$\mathbf{Q}^{(t)}, \mathbf{q}^{(t)}, q_0^{(t)} = \arg\min_{\mathbf{Q},\mathbf{q},q_0} \sum_{j\in\mathcal{J}^{(t)}} \|d_Q^{(t)}(\boldsymbol{\lambda}^{(j)}) - d(\boldsymbol{\lambda}^{(j)})\|_2^2. \tag{85}$$

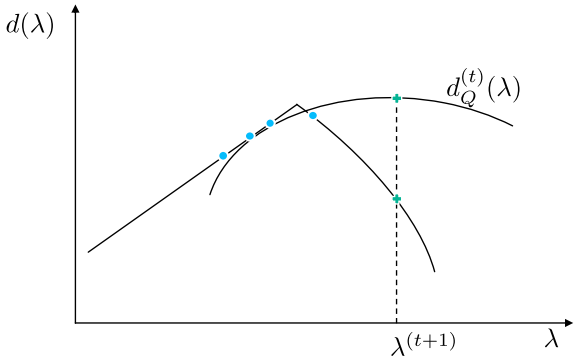
The regression data

$$\mathcal{D}^{(t)} = \{(\boldsymbol{\lambda}^{(j)}, d(\boldsymbol{\lambda}^{(j)}))|\ j \in \mathcal{J}^{(t)}\}, \tag{86}$$

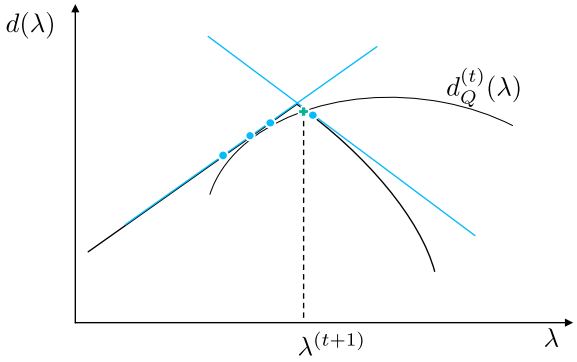
is selected using the NAPS algorithm. Once the quadratic approximation has been performed, the dual variables can be updated by maximizing the approximated dual function. The update step of the dual variables can be interpreted as an ascent step for the dual function using a quadratic approximation. 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 the information about the violation of the system-wide constraints, the QADA algorithm additionally requires the information about the contributions of the subproblems to the dual function,

$$d_i(\boldsymbol{\lambda}) = \min_{\mathbf{x}_i\in\mathcal{X}_i} \mathcal{L}_i(\mathbf{x}_i, \lambda). \tag{87}$$



(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.

Fig. 4. Illustration of the effect of bundle cuts.

This essentially means that the QADA algorithm collects the same information as bundle methods (30), 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.

5.1.6. Bundle cuts

The core idea of the QADA algorithm is that a quadratic surrogate model of the dual function is computed and optimized in order 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. 4a. 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 is-

sue, the collected subgradients can be used to formulate cutting planes. According to the definition of the subgradient (22) the following relation holds between the dual function $d(\boldsymbol{\lambda})$ and a subgradient $\mathbf{g}(\boldsymbol{\lambda}^{(j)})$ at a point $\boldsymbol{\lambda}^{(j)}$:

$$d(\boldsymbol{\lambda}) \leq d(\boldsymbol{\lambda}^{(j)}) + \mathbf{g}^T(\boldsymbol{\lambda}^{(j)})(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(j)}). \tag{88}$$

This implies that a quadratic approximation of the dual function is not valid, if it does not satisfy condition (88). Therefore, the collected subgradients are used to formulate additional constraints on the updated dual variables $\boldsymbol{\lambda}^{(t+1)}$, referred to as bundle cuts in the following:

$$d_Q^{(t)}(\boldsymbol{\lambda}^{(t+1)}) \leq d(\boldsymbol{\lambda}^{(j)}) + \mathbf{g}^T(\boldsymbol{\lambda}^{(j)})(\boldsymbol{\lambda}^{(t+1)} - \boldsymbol{\lambda}^{(j)}), \quad \forall j \in \{t - \tau + 1, \dots, t\}. \tag{89}$$

The bundle cuts are formulated by using the data points that are not older than the age parameter τ. Fig. 4b 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. (89) 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. In the following, the bundle cuts are summarized as

$$\begin{aligned}\mathcal{BC}^{(t)} = \{\boldsymbol{\lambda} \in \mathbb{R}^{n_\mathbf{b}} | d_Q^{(t)}(\boldsymbol{\lambda}) \leq d(\boldsymbol{\lambda}^{(j)}) + \mathbf{g}^T(\boldsymbol{\lambda}^{(j)})(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(j)}),\\ \forall j \in \{t - \tau + 1, \dots, t\}\}.\end{aligned} \tag{90}$$

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,

$$\boldsymbol{\lambda}^{(t+1)} = \arg\max_{\boldsymbol{\lambda} \in \mathbb{R}^{n_\mathbf{b}}} d_Q^{(t)}(\boldsymbol{\lambda}), \tag{91a}$$

$$\text{s.t. } \boldsymbol{\lambda} \in \mathcal{E}(\boldsymbol{\Lambda}^{(t)}) \cap \mathcal{BC}^{(t)}, \tag{91b}$$

$$\boldsymbol{\lambda} \geq \mathbf{0}. \tag{91c}$$

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 5). However, since the QADA algorithm also uses the bundle information, BTM could also be used for the initial sampling. Steps 5–9 are performed in parallel by the subproblems while steps 10–37 are performed by the coordinator.

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, $t_{\max}$

$t \leftarrow 0$
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 (1b) 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 (1b) 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
    $\alpha^{(t)} \leftarrow \alpha^{(0)} / \max\{\|\mathbf{w}_p^{(0)}\|_2, \ldots, \|\mathbf{w}_p^{(t)}\|_2\}$
    if Constraints (1b) are inequalities **then**
      $\boldsymbol{\lambda}^{(t+1)} \leftarrow [\boldsymbol{\lambda}^{(t)} + \alpha^{(t)}\mathbf{g}(\boldsymbol{\lambda}^{(t)})]^+$
    else if Constraints (1b) are equalities **then**
      $\boldsymbol{\lambda}^{(t+1)} \leftarrow \boldsymbol{\lambda}^{(t)} + \alpha^{(t)}\mathbf{g}(\boldsymbol{\lambda}^{(t)})$
    end if
  else ▷ Perform QADA Updates
    $\mathcal{D}^{(t)} \leftarrow \text{NAPS}(\boldsymbol{\Lambda}, d(\boldsymbol{\Lambda}), \tau, \Delta\boldsymbol{\lambda})$
    $\mathcal{E}(\boldsymbol{\Lambda}^{(t)}) \leftarrow \text{ComputeEllipsoid}(\boldsymbol{\Lambda}^{(t)}, \underline{s}_i, \overline{s}_i, \underline{\gamma})$
    $d_Q^{(t)}(\boldsymbol{\lambda}) \leftarrow \text{Regression}(\mathcal{D}^{(t)})$
    $\mathcal{F}^{(t)} \leftarrow \mathcal{E}(\boldsymbol{\Lambda}^{(t)}) \cap \mathcal{BC}^{(t)}$
    if Constraints (1b) are inequalities **then**
      $\boldsymbol{\lambda}^{(t+1)} \leftarrow \text{argmax}_{\boldsymbol{\lambda} \in \mathcal{F}^{(t)}} d_Q^{(t)}(\boldsymbol{\lambda}),\ \text{s.t. } \boldsymbol{\lambda} \geq \mathbf{0}$
    else if Constraints (1b) are equalities **then**
      $\boldsymbol{\lambda}^{(t+1)} \leftarrow \text{argmax}_{\boldsymbol{\lambda} \in \mathcal{F}^{(t)}} d_Q^{(t)}(\boldsymbol{\lambda})$
    end if
  end if
  $\mathbf{w}_d^{(t)} \leftarrow \boldsymbol{\lambda}^{(t+1)} - \boldsymbol{\lambda}^{(t)}$
until $(\|\mathbf{w}_p^{(t)}\|_2 \leq \epsilon_p \wedge \|\mathbf{w}_d^{(t)}\|_2 \leq \epsilon_d) \vee (t \geq t_{\max})$
return $\boldsymbol{\lambda}^{(t)}$

5.1.7. Summary of regression-based algorithms

Fig. 5 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 is communicated to the coordinator. If not enough iterations have been performed, the dual variables are updated using the subgradient method (23b). 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 constraints, the dual variables are updated by optimizing the surrogate function.

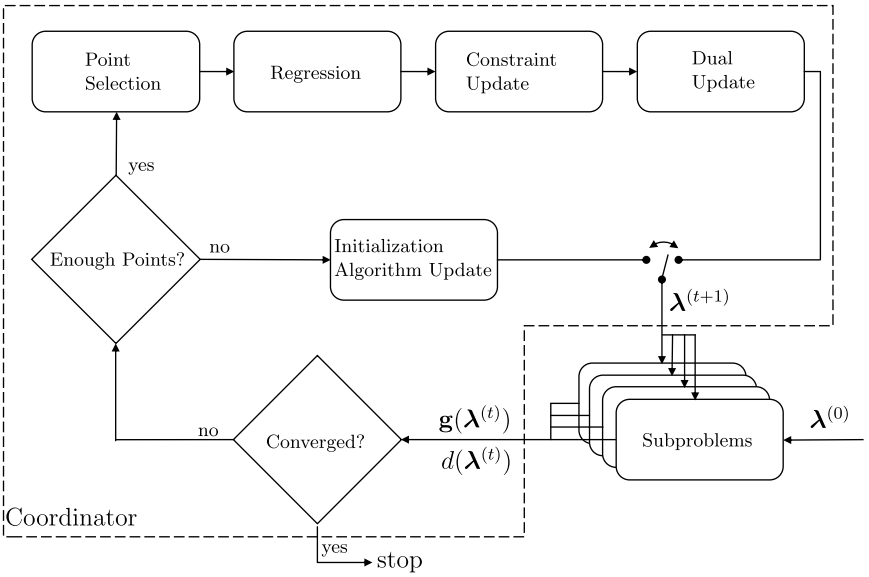


Fig. 5. Flowchart of the regression-based coordination algorithms (adapted from [69]).

5.2. Quasi-Newton dual ascent

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 paper the same principle is applied to the dual optimization problem (4). If no individual constraints (1c) 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. The proposed quasi-Newton dual ascent (QNDA) algorithm is described in the following. A decentralized quasi-Newton algorithm was presented in [18] and [19]. There the subproblems use the curvature of their own 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 paper. Furthermore, no individual constraints are considered, which would result in a nonsmoothness of the dual problem. In contrast, the algorithm presented in this 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,

$$d(\boldsymbol{\lambda}) \approx \frac{1}{2}(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)})^T \nabla^2 d(\boldsymbol{\lambda}^{(t)})(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)}) + \nabla^T d(\boldsymbol{\lambda}^{(t)})(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)}) + d(\boldsymbol{\lambda}^{(t)}). \tag{92}$$

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 (92) 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:

$$d_B^{(t)}(\boldsymbol{\lambda}) = \frac{1}{2}(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)})^T \mathbf{B}^{(t)}(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)}) + \mathbf{g}^T(\boldsymbol{\lambda}^{(k)})(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)}) + d(\boldsymbol{\lambda}^{(t)}), \tag{93}$$

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 of an optimization problem based on previous data. Thus, the surrogate function $d_B(\boldsymbol{\lambda})$ is obtained through the solution of an optimization problem (52). 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,

$$\mathbf{s}^{(t)} := \boldsymbol{\lambda}^{(t)} - \boldsymbol{\lambda}^{(t-1)} \tag{94}$$

and of the subgradients

$$\mathbf{y}^{(t)} = \mathbf{g}(\boldsymbol{\lambda}^{(t)}) - \mathbf{g}(\boldsymbol{\lambda}^{(t-1)}) \tag{95}$$

are defined. The approximated Hessian is then updated according to [51],

$$\mathbf{B}^{(t)} = \arg \min_{\mathbf{B} \in S\mathbb{R}^{n_\mathbf{b} \times n_\mathbf{b}}} \|\mathbf{B} - \mathbf{B}^{(t-1)}\|_F \tag{96a}$$

$$\text{s.t. } \mathbf{B}^{(-1)}\mathbf{y}^{(t)} = \mathbf{s}^{(t)}, \tag{96b}$$

were $\|\cdot\|_F$ denotes the (weighted) Frobenius norm. The approximated Hessian has to be symmetric, since the actual Hessian is also always symmetric. Constraint (96b) is called the secant condition and captures the local curvature of the objective function. The solution of (96) can be written in a closed form as [51]

$$\mathbf{B}^{(t)} = \mathbf{B}^{(t-1)} + \frac{\mathbf{y}^{(t)}\mathbf{y}^{(t),T}}{\mathbf{y}^{(t),T}\mathbf{s}^{(t)}} - \frac{\mathbf{B}^{(t-1)}\mathbf{s}^{(t)}\mathbf{s}^{(t),T}\mathbf{B}^{(t-1),T}}{\mathbf{s}^{(t),T}\mathbf{B}^{(t-1)}\mathbf{s}^{(t)}}. \tag{97}$$

Eq. (97) is the well-known BFGS-update scheme. The surrogate function $d_B^{(t)}(\boldsymbol{\lambda})$ is a smooth approximation of the dual function. In order 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

$$\boldsymbol{\lambda}^{(t+1)} = \arg\max_{\boldsymbol{\lambda}\in\mathbb{R}^{n_\mathbf{b}}} d_B^{(t)}(\boldsymbol{\lambda}), \tag{98a}$$

$$\text{s.t. } \|\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)}\|_2^2 \leq \alpha^{(t)}, \tag{98b}$$

$$\boldsymbol{\lambda} \in \mathcal{BC}^{(t)}, \tag{98c}$$

$$\boldsymbol{\lambda} \geq \mathbf{0}. \tag{98d}$$

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 5–9 are performed in parallel by the subproblems while steps 10–36 are performed by the coordinator.

5.3. Discussion of the convergence properties of the QADA and QNDA algorithms

In this section, we provide a preliminary analysis of the convergence properties of the QADA and QNDA algorithms 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 section. In the real implementation and parameterization, the algorithms include heuristic components which can only be validated by tests for well-designed benchmark problems.

First, we consider the case of distributed quadratic programs without individual constraints,

$$\min_{\mathbf{x}_1,\ldots,\mathbf{x}_{N_s}} \sum_{i\in\mathcal{I}} \frac{1}{2}\mathbf{x}_i^T\mathbf{H}_i\mathbf{x}_i + \mathbf{c}_i^T\mathbf{x}_i, \tag{70a}$$

$$\text{s.t. } \sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i = \mathbf{b}. \tag{70b}$$

As was shown in Section 5.1.5, the dual function of Problem (70) is

$$d(\boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{\lambda}^T(-\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^T)\boldsymbol{\lambda} + (-\mathbf{c}^T\mathbf{H}^{-1}\mathbf{A}^T - \mathbf{b}^T)\boldsymbol{\lambda} + \left(-\frac{1}{2}\mathbf{c}^T\mathbf{H}^{-1}\mathbf{c}\right), \tag{83}$$

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

$$\mathbf{g}(\boldsymbol{\lambda}^{(t)}) = \mathbf{A}\mathbf{x}^{(t+1)} - \mathbf{b}, \tag{100}$$

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 (1b) 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 (1b) are equalities **then**
    $\mathbf{w}_p^{(t)} \leftarrow \mathbf{g}(\boldsymbol{\lambda}^{(t)})$
  $\alpha^{(t)} \leftarrow \alpha^{(0)} / \max\{\|\mathbf{w}_p^{(0)}\|_2, \ldots, \|\mathbf{w}_p^{(t)}\|_2\}$
  if $j = 1$ **then** ▷ Perform SG update in first iteration.
    if Constraints (1b) are inequalities **then**
      $\boldsymbol{\lambda}^{(t+1)} \leftarrow [\boldsymbol{\lambda}^{(t)} + \alpha^{(t)}\mathbf{g}(\boldsymbol{\lambda}^{(t)})]^+$
    else if Constraints (1b) are equalities **then**
      $\boldsymbol{\lambda}^{(t+1)} \leftarrow \boldsymbol{\lambda}^{(t)} + \alpha^{(t)}\mathbf{g}(\boldsymbol{\lambda}^{(t)})$
    end if
  else ▷ Perform QNDA Updates
    $\mathbf{y}^{(t)} \leftarrow \mathbf{g}(\boldsymbol{\lambda}^{(t)}) - \mathbf{g}(\boldsymbol{\lambda}^{(t-1)})$
    $\mathbf{B}^{(t)} \leftarrow \mathrm{BFGS}(\mathbf{B}^{(t-1)}, \mathbf{y}^{(t)}, \mathbf{s}^{(t)})$
    $\mathcal{F}^{(t)} \leftarrow \{\boldsymbol{\lambda} \in \mathbb{R}^{n_{\mathbf{B}}} \mid \|\boldsymbol{\lambda} - \boldsymbol{\lambda}^t\|_2^2 \leq \alpha^{(t)}\} \cap \mathcal{BC}^{(t)}$
    if Constraints (1b) are inequalities **then**
      $\boldsymbol{\lambda}^{(t+1)} \leftarrow \mathrm{argmin}_{\boldsymbol{\lambda} \in \mathcal{F}^{(t)}} d_B^{(t)}(\boldsymbol{\lambda}),\ \text{s.t. } \boldsymbol{\lambda} \geq \mathbf{0}$
    else if Constraints (1b) are equalities **then**
      $\boldsymbol{\lambda}^{(t+1)} \leftarrow \mathrm{argmin}_{\boldsymbol{\lambda} \in \mathcal{F}^{(t)}} \tilde{d}_B^{(t)}(\boldsymbol{\lambda})$
    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 $(\|\mathbf{w}_p^{(t)}\|_2 \leq \epsilon_p \wedge \|\mathbf{w}_d^{(t)}\|_2 \leq \epsilon_d) \vee (t \geq t_{\max})$
return $\boldsymbol{\lambda}^{(t)}$

with

$$\mathbf{x}^{(t+1)} = \arg\min_{\mathbf{x} \in \mathbb{R}^{n_{\mathbf{x}}}} \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{c}^T\mathbf{x} + \boldsymbol{\lambda}^{(t),T}(\mathbf{A}\mathbf{x} - \mathbf{b}) \tag{101}$$

$$= -\mathbf{H}^{-1}(\mathbf{c} + \mathbf{A}^T\boldsymbol{\lambda}^{(t)}). \tag{73}$$

Inserting (73) into (100) yields

$$\mathbf{g}(\boldsymbol{\lambda}^{(t)}) = -\mathbf{A}\mathbf{H}^{-1}(\mathbf{c} + \mathbf{A}^T\boldsymbol{\lambda}) + \mathbf{b} = \nabla d(\boldsymbol{\lambda}^{(t)}), \tag{102}$$

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 a steepest ascent method with a step size $\alpha^{(t)}$ and a search direction $\mathbf{s}^{(t)} = \nabla d(\boldsymbol{\lambda}^{(t)})$,

$$\boldsymbol{\lambda}^{(t)} = \boldsymbol{\lambda}^{(t)} + \alpha^{(t)}\mathbf{s}^{(t)}. \tag{103}$$

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

$$\mathbf{s}^{(t)} = \mathbf{B}^{(t),-1}\nabla d(\boldsymbol{\lambda}^{(t)}), \tag{104}$$

where $\mathbf{B}^{(t)}$ denotes the approximation of the Hessian computed by (97). 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, the 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 [51],

$$d(\boldsymbol{\lambda}^{(t)} + \alpha^{(t)}\mathbf{s}^{(t)}) \geq d(\boldsymbol{\lambda}^{(t)}) + \beta_1\alpha^{(t)}\nabla^T d(\boldsymbol{\lambda}^{(t)})\mathbf{s}^{(t)}, \tag{105a}$$

$$\nabla^T d(\boldsymbol{\lambda}^{(t)} + \alpha^{(t)}\mathbf{s}^{(t)})\mathbf{s}^{(t)} \geq \beta_2\nabla^T d(\boldsymbol{\lambda}^{(t)})\mathbf{s}^{(t)}, \tag{105b}$$

$$\beta_1 \in (0,1),\ \beta_2 \in (\beta_1, 1). \tag{105c}$$

However, the problem with finding a suitable step size through conditions (105) 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 hyper-parameters. 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 (70) the dual function is quadratic, but unknown to the coordinator. For enough data points, assuming a well-conditioned Vandermonde matrix $\mathbf{M}^{(t)}$ (60) 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 without individual constrains Wenzel et al. [69] showed that the primal residual for QAC is piecewise 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 a 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 towards the global optimum of the dual function.

Similar arguments can be made for more general problems, as 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 nonsmoothness 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,

$$\hat{d}^{(t)}(\boldsymbol{\lambda}) := \min_{j \in \mathcal{J}^{(t)}} \{d(\boldsymbol{\lambda}^{(j)}) + \mathbf{g}^T(\boldsymbol{\lambda}^{(j)})(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(j)})\}. \tag{31}$$

It has been shown that the cutting plane model will exactly match the concave nonsmooth dual function, as t goes to ∞ [7], i.e.,

$$\lim_{t \to \infty} \hat{d}^{(t)}(\boldsymbol{\lambda}) = d(\boldsymbol{\lambda}), \tag{106}$$

if all previously collected points are kept in the bundle. This is the basis of the proof of the (theoretical) convergence of bundle methods. 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 (98) can be reformulated as

$$\boldsymbol{\lambda}^{(t+1)} = \arg \max_{\boldsymbol{\lambda} \in \mathbb{R}^{n_\mathbf{b}}} d_B^{(t)}(\boldsymbol{\lambda}), \tag{107a}$$

$$\text{s.t. } \|\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)}\|_2^2 \leq \alpha^{(t)}, \tag{107b}$$

$$d_B^{(t)}(\boldsymbol{\lambda}) \leq \hat{d}^{(t)}(\boldsymbol{\lambda}), \tag{107c}$$

$$\boldsymbol{\lambda} \geq \mathbf{0}. \tag{107d}$$

As t tends to ∞, the constraints (107c) prevent the algorithm from moving in a 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 capacity. 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 hyper-parameter. If sufficient data is kept in the bundle good performance can be observed in practice.

In this paper 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 function, i.e., convergence of the dual variables. For convex problems, the optimal primal solution can be obtained at the dual optimum, since strong duality holds. 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. [60] propose a 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 paper and discussed in more detail in Section 6.2. Feasibility is proven for mixed-integer linear programming problems. 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 actually defined as the infimum of the Lagrange function for a value of the dual variables (3). 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.

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 (39) 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 [8]. The practical efficiency of the new proposed algorithms in comparison to the benchmark algorithms, both in terms of number of required iterations and solution accuracy, is demonstrated in the next section.

6. Computational results

In this section, the performance of the proposed new QADA and QNDA algorithms is compared to the subgradient method, BTM, ADMM and to 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).

All algorithms and subproblems were implemented in the programming language Julia [5] using the optimization toolbox JuMP [17]. All affine, QP and MIQP subproblems were solved using the commercial solver Gurobi [29], while general convex problems were solved using the interior-point solver IPOPT [61]. 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 were performed on a standard Laptop PC (Intel(R) Core(TM) i5-6200U CPU @ 2.30 GHz, 8 GB RAM).

In order to assess the efficiency of the different algorithms the computation time required for the solution of a distributed optimization problem is computed as [55]

$$T_{\text{comp}} = N_{\text{iter}} \cdot T_{\text{comm}} + \sum_{t=1}^{N_{\text{iter}}} (T_{\text{update}}^{(t)} + \max_{i \in \mathcal{I}} T_{\text{sub},i}^{(t)}), \tag{108}$$

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 iteration 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 are terminated if the Euclidean norms of the primal and dual residuals lie below a threshold ϵ_p and ϵ_d respectively, or when the maximum number of iterations $t_{\max}$ is reached.

6.1. Distributed QPs

A large number of distributed QP benchmark problems were defined in [65] and [69] with the following structure:

$$\min_{\mathbf{x}_1,\ldots,\mathbf{x}_{N_s}} \sum_{i \in \mathcal{I}} \frac{1}{2} \mathbf{x}_i^T \mathbf{H}_i \mathbf{x}_i + \mathbf{c}_i^T \mathbf{x}_i, \tag{109a}$$

$$\text{s. t.} \sum_{i \in \mathcal{I}} \mathbf{A}_i \mathbf{x}_i = 0, \tag{109b}$$

$$\mathbf{x}_i^{\text{LB}} \leq \mathbf{x}_i \leq \mathbf{x}_i^{\text{UB}},\ \forall i \in \mathcal{I}, \tag{109c}$$

with $\mathbf{x}_i \in \mathbb{R}^{n_{\mathbf{x}_i}}$. The system-wide constraints (109b) can be interpreted as a resource network balance, were 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 matches.

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

$$\mathbf{H}_i = \mathbf{M}_i^T \mathbf{M}_i,\ \mathbf{M}_i \in \mathbb{R}^{n_{\mathbf{x}_i} \times n_{\mathbf{x}_i}}, \tag{110}$$

where the elements of $\mathbf{M}_i$ were drawn from a normal distribution $[\mathbf{M}_i]_{l,j} \in \mathcal{N}(\mu = 0, \sigma = 1)$. 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 $\mathcal{U}_c(1, 2)$. Afterwards, they were altered such that their sign is flipped or they are set to zero through the uniform discrete distribution $\mathcal{U}_d[-1, 0, 1]$,

$$\mathbf{A}_i = \mathbf{B}_i \circ \mathbf{C}_i \in \mathbb{R}^{n_{\mathbf{b}} \times n_{\mathbf{x}_i}}, \qquad [\mathbf{B}_i]_{l,j} \in \mathcal{U}_c(1, 2), [\mathbf{C}_i]_{l,j} \in \mathcal{U}_d[-1, 0, 1], \tag{111}$$

where $\circ$ denotes element-wise multiplication. If a row in $\mathbf{A}_i$ only contained zeros, a correction step was performed that creates at least one nonzero entry. Box constraints (109c) were used as individual constraints for each subproblem. In all cases $[\mathbf{x}_i^{\mathrm{LB}}]_l = -10$ and $[\mathbf{x}_i^{\mathrm{UB}}]_l = 10$.

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 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 subsystems and number of coupling constraints/variables $(N_s, n_{\mathbf{b}})$. In the following, the notation $\mathrm{QP}^{(R)}_{(N_s, n_{\mathbf{b}})}$ is used, where R indicates the number of the problem instance. For example, $\mathrm{QP}^{(7)}_{(256,2)}$ refers to the seventh problem instance containing 256 constrained quadratic programs, each with 2 variables, connected through 2 coupling constraints. In [65] and [69] $n_{\mathbf{x}_i} \in \{2, \ldots, 5\}$ was considered, resulting in 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.

6.1.1. Parameter settings for distributed QPs

For the subgradient method (SG) the initial step size parameters was set to $\alpha^{(0)} = 2 \times 10^{-3}$ and then varied according to (28). 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 (50) with $\tau_{\text{decr}} = 1.25$, $\tau_{\text{incr}} = 1.5$ and $\mu = 10$. The parameters for the regression-based algorithms were mostly chosen as in [69]. 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 (69), 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

$$\|\mathbf{w}_p(\boldsymbol{\lambda}^{(t)})\|_2 \leq \epsilon_b \cdot \|\mathbf{w}_p(\boldsymbol{\lambda}^{(0)})\|_2. \tag{112}$$

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, in order to find the parameters that result in the most converged benchmark problems. All parameters are summarized in Table A.5 in the appendix.

6.1.2. 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 Table 1 and in Fig. 6. A more extensive summary is given in Table D.7 in the appendix. The results show that the subgradient method performs poorly for the considered benchmarks. Only a small fraction of the problems

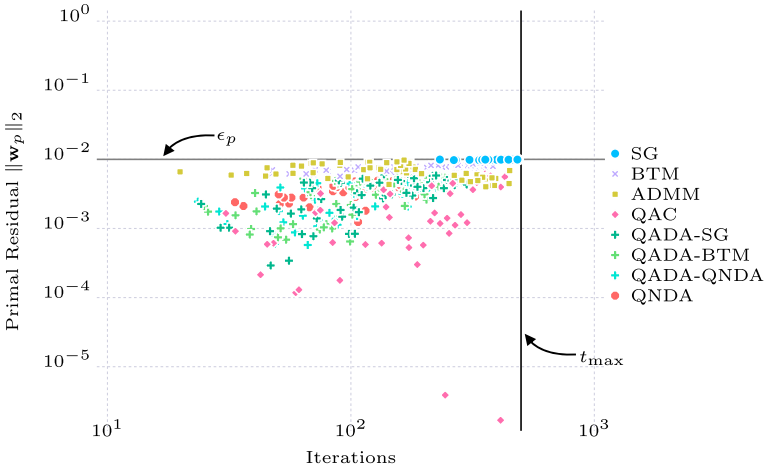
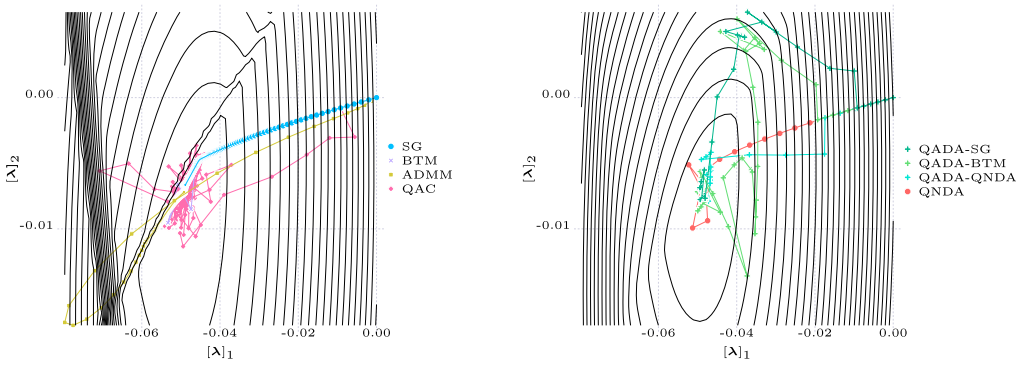


Fig. 6. Mean values of the primal residuals upon convergence for the distributed quadratic programs. Each data point represents the mean values of the converged problem instances for a pair N_s and $n_{\mathbf{b}}$ (cf. Table D.7).

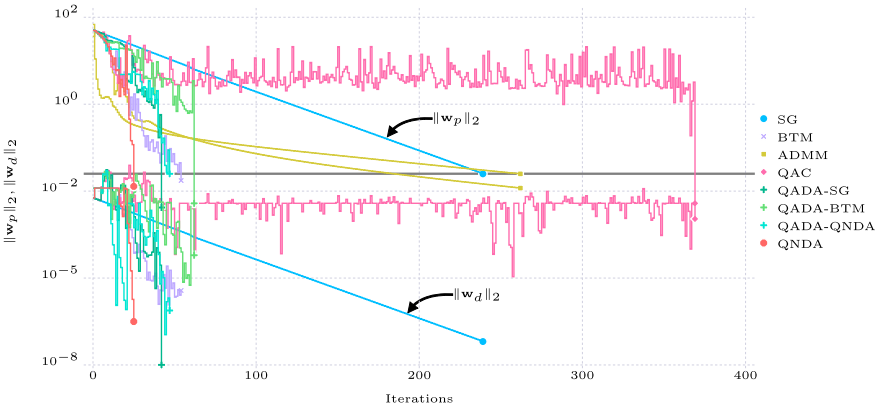
Table 1
Summary of the results for the distributed optimization of QPs (mean values of the converged instances only), $\overline{t}$: mean number of iterations until convergence, $\overline{T_{\text{comp}}}$: mean computation time of converged runs (in s), $\overline{\|\mathbf{w}_p\|_2}$: mean primal residual of converged runs ($\times 10^{-3}$), $\%_c$: percentage of converged runs within $t_{\max}$ iterations.

Algorithm	$\overline{t}$	$\overline{T_{\text{comp}}}$	$\overline{\|\mathbf{w}_p\|_2}$	$\%_c$
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**

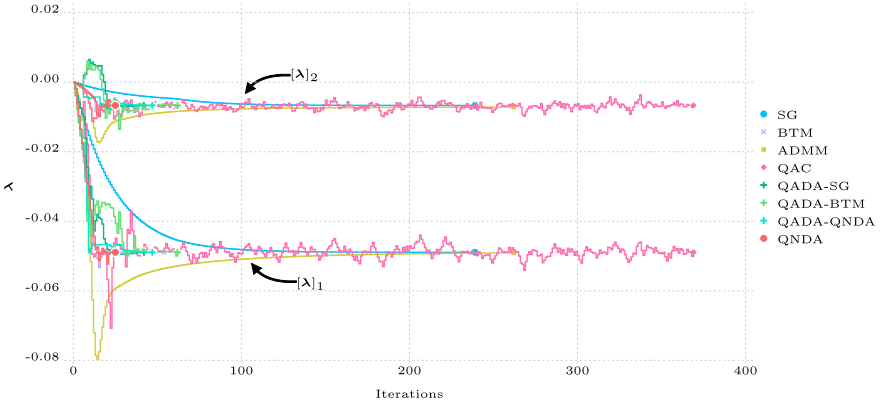
is solved within the allowed number of iterations. Additionally, the problems that do converge require a large number of iterations and long computation times. BTM and ADMM are significantly more robust, being able to solve most of the benchmark problems. While BTM solves more problems, ADMM requires 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 benchmark problems. Note however, that all other algorithms except the subgradient method use more information on the subproblems and that ADMM enforces that the subsystem problems 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 almost all benchmark problems. Interestingly, while QADA requires less iterations



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

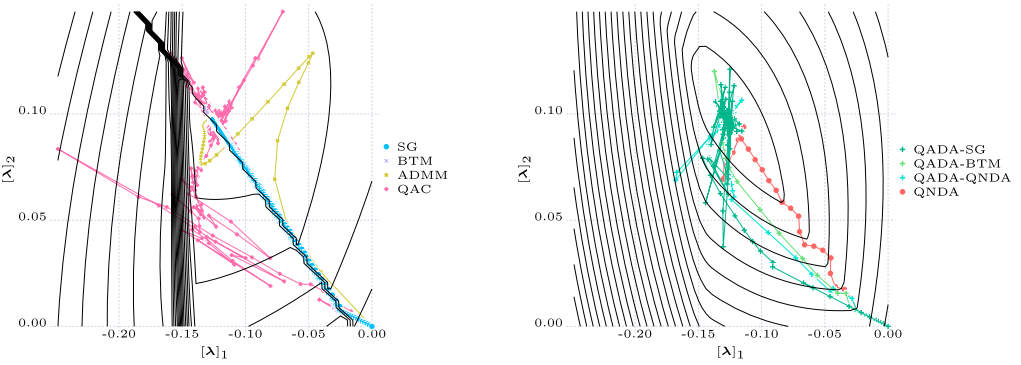


(b) Evolution of the primal and dual residuals.

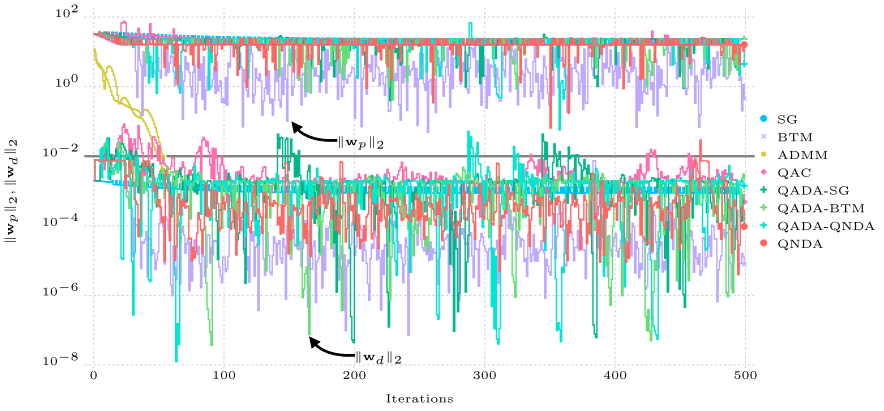


(c) Evolution of the dual varialbes.

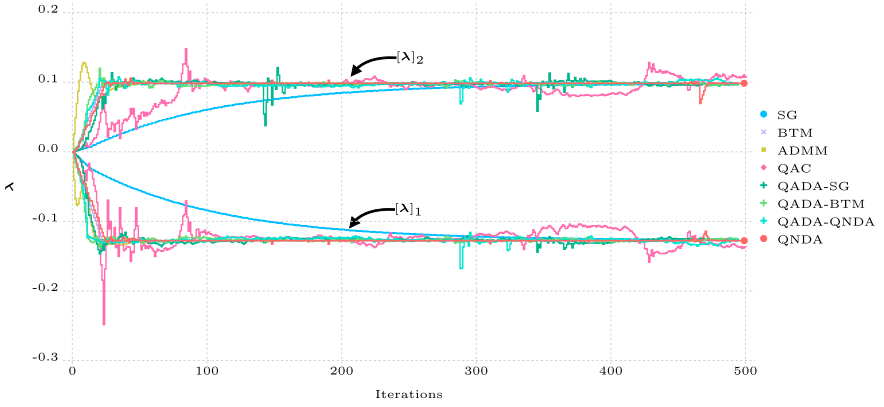
Fig. 7. Results of the distributed optimization of problem $\mathrm{QP}^{(7)}_{(256,2)}$.



(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.

Fig. 8. Results of the distributed optimization of problem $\mathrm{QP}^{(17)}_{(16,2)}$.

to converge, QNDA requires less computation time. This is due to the fact that 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, 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. 7 shows the results for the distributed optimization of benchmark problem $\mathrm{QP}^{(7)}_{(256,2)}$. The contour plots in Fig. 7a 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 cause 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 are able to handle the nonsmoothness of the dual function in most cases, this does not apply to all benchmark problems. Fig. 8 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 [69] only convex QPs were considered. In this paper the computational results are extended by also considering distributed MIQPs. The benchmark problems have the following structure:

$$\min_{\mathbf{x}_1,\ldots,\mathbf{x}_{N_s}} \sum_{i\in\mathcal{I}} \frac{1}{2}\mathbf{x}_i^T\mathbf{H}_i\mathbf{x}_i + \mathbf{c}_i^T\mathbf{x}_i, \tag{113a}$$

$$\text{s.t.} \sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i \leq \mathbf{b}, \tag{113b}$$

$$\mathbf{D}_i\mathbf{x}_i \leq \mathbf{d}_i,\ \forall i \in \mathcal{I}, \tag{113c}$$

$$\mathbf{x}_i^{\text{LB}} \leq \mathbf{x}_i \leq \mathbf{x}_i^{\text{UB}},\ \forall i \in \mathcal{I}, \tag{113d}$$

$$\mathbf{x}_i \in \mathbb{R}^{n^c_{\mathbf{x}_i}} \times \mathbb{Z}^{n^d_{\mathbf{x}_i}}, \forall i \in \mathcal{I}, \tag{113e}$$

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 $\mathbf{H}_i$, $\mathbf{c}_i$, $\mathbf{A}_i$, $\mathbf{x}_i^{\text{LB}}$ and $\mathbf{x}_i^{\text{UB}}$ were generated in the same way as for the distributed QPs. The elements for the individual constraints (113c) 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 (113b) $\mathbf{b}$ 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 were the solutions of the subproblems are completely decoupled, i.e., were $\boldsymbol{\lambda} = \mathbf{0}$ results in a feasible solution. This trivial solution is avoided by tightening the system-wide constraints. Once all subproblems were generated, the decoupled subproblems

$$\min_{\mathbf{x}_i} \frac{1}{2}\mathbf{x}_i^T\mathbf{H}_i\mathbf{x}_i + \mathbf{c}_i^T\mathbf{x}_i, \tag{114a}$$

$$\text{s.t.}\ \mathbf{D}_i\mathbf{x}_i \leq \mathbf{d}_i,\ \forall i \in \mathcal{I}, \tag{114b}$$

$$\mathbf{x}_i^{\text{LB}} \leq \mathbf{x}_i \leq \mathbf{x}_i^{\text{UB}},\ \forall i \in \mathcal{I}, \tag{114c}$$

$$\mathbf{x}_i \in \mathbb{R}^{n^c_{\mathbf{x}_i}} \times \mathbb{Z}^{n^d_{\mathbf{x}_i}}, \forall i \in \mathcal{I}, \tag{114d}$$

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

$$[\mathbf{b}]_l = [\mathbf{b}]_l - (1 + [\boldsymbol{\beta}]_l) \left\| \sum_{i\in\mathcal{I}} \mathbf{A}_i\tilde{\mathbf{x}}_i^* \right\|_2, \tag{115}$$

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 were considered, i.e., problems with $N_s \gg n_{\mathbf{b}}$ [12,60]. The number and size of the subproblems were varied as follows:

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

All subproblems contain the same number of variables $n_{\mathbf{x}} = n_{\mathbf{x}_i}$ 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 (113e) 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. [60] 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,

$$\sum_{i\in\mathcal{I}} \mathbf{A}_i \mathbf{x}_i \leq \overline{\mathbf{b}}, \tag{116a}$$

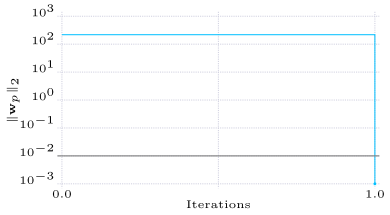
$$\overline{\mathbf{b}} = \mathbf{b} - \boldsymbol{\zeta}, \tag{116b}$$

$$[\boldsymbol{\zeta}]_l = n_{\mathbf{b}} \cdot \max_{i\in\mathcal{I}} \left\{ \max_{\mathbf{x}_i\in\mathcal{X}_i} [\mathbf{A}_i]_{l,:}\mathbf{x}_i - \min_{\mathbf{x}_i\in\mathcal{X}_i} [\mathbf{A}_i]_{l,:}\mathbf{x}_i \right\}, \tag{116c}$$

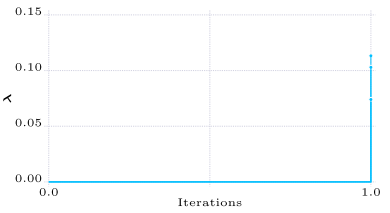
where $[\mathbf{A}_i]_{l,:}$ denotes the lth row of the matrix $\mathbf{A}_i$. The same contraction was used in this paper. Therefore, in a first step all subproblems have to solve the two inner optimization problems in (116c) and communicate the results to the coordinator. The coordinator then collects all responses and tightens the coupling constraints. It is important to note, that the contracted right-hand side $\bar{\mathbf{b}}$ 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 (26) and nonnegativity constraints are imposed on the dual variables.

6.2.2. Parameter settings for distributed MIQPs

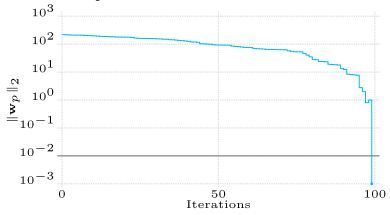
As noted earlier, the coupling constraints (113b) are inequalities for the MIQP benchmark problems. In general, it is easier to find a feasible solution for inequality constrained problems using dual decomposition-based distributed optimization than for equality constrained ones. By selecting larger 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. 9 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. However,



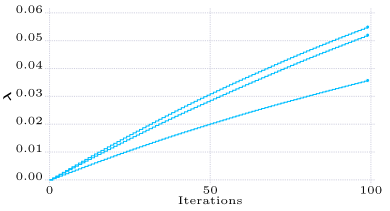
(a) Evolution of the primal and dual residuals for $\alpha^{(0)} = 5 \times 10^{-2}$.



(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 $\alpha^{(0)} = 3 \times 10^{-4}$.

Fig. 9. 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 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. (117)). This is due to the obtained values of the dual variables. As seen in Fig. 9b and 9d 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 in order to obtain better primal solutions.

The step size 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, in order to find the parameters that result in the most converged benchmark problems. All parameters are summarized in Table A.5 in the appendix.

As discussed in Sec. 6.2.1, the distributed optimization algorithms actually try to solve a primal problem with the contracted right-hand side $\overline{\mathbf{b}}$. However, the algorithms are terminated prematurely, i.e., when the original problem is feasible. In this case the dual variables might not have converged to a stationary value, as they are updated on the basis of the tightened problem. Therefore, only the primal residual is used as a convergence criterion for the MIQP benchmark problems in order 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.

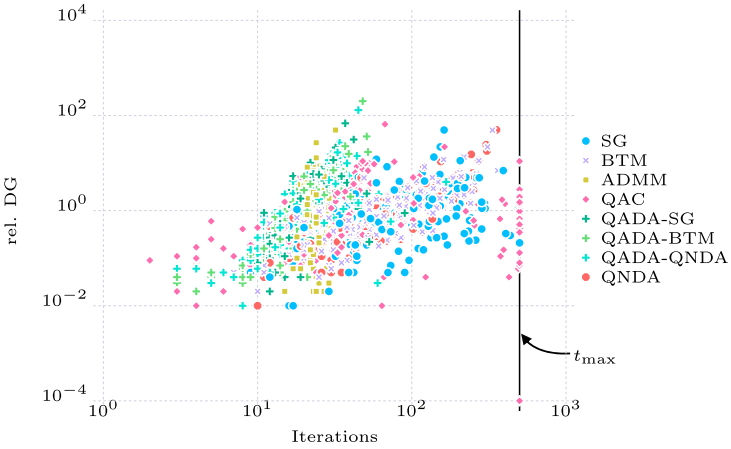


Fig. 10. Relative duality gaps (rel. DG) of the MIQP benchmark problems upon termination for the examined algorithms. The value of the rel. DG for not converged runs has no meaning, as it corresponds to an infeasible primal solution (cf. Table D.8).

Table 2
Summary of the results for the coordination of MIQPs (mean values of the converged instances only), $\overline{t}$: 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.

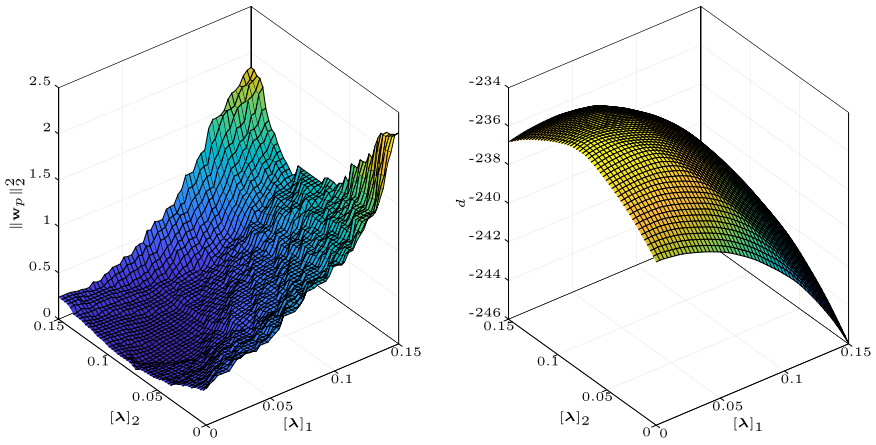
Algorithm	$\overline{t}$	$\overline{T_{\text{comp}}}$	rel. DG	$\%_c$
SG	86.69	69.88	**1.66**	99.5
BTM	80.52	65.41	**1.66**	**100**
ADMM	25.06	21.16	2.22	**100**
QAC	59.40	52.83	2.13	86.0
QADA-SG	**19.37**	**18.37**	2.54	**100**
QADA-BTM	20.78	18.62	3.53	**100**
QADA-QNDA	22.20	22.76	2.86	**100**
QNDA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

6.2.3. Results for distributed MIQPs

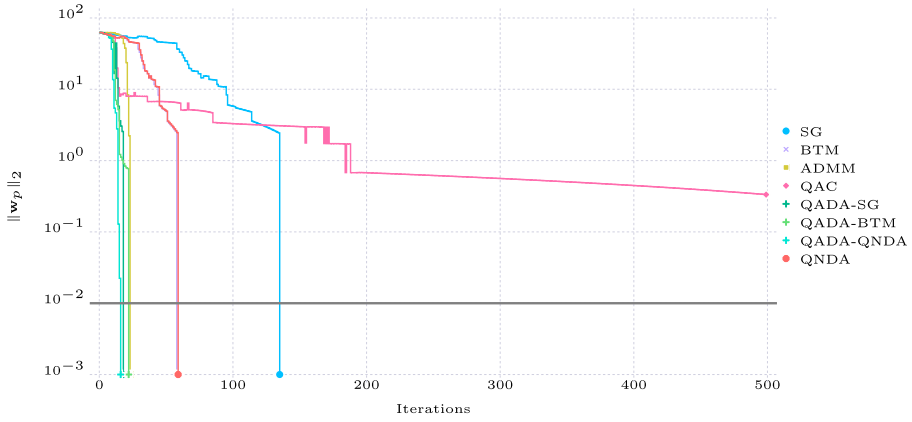
The MIQP benchmark problems were solved using the same algorithms as for the QP problems. The results are illustrated in Fig. 10 and summarized in Table 2. Instead of depicting the primal residual (which is 0 for converged runs), Fig. 10 shows the relative duality gap,

$$\text{rel. DG} = 100 \cdot \frac{\sum_{i\in\mathcal{I}} f_i(\mathbf{x}_i^*(\boldsymbol{\lambda})) - d(\boldsymbol{\lambda})}{\sum_{i\in\mathcal{I}} f_i(\mathbf{x}_i^*(\boldsymbol{\lambda}))} \tag{117}$$

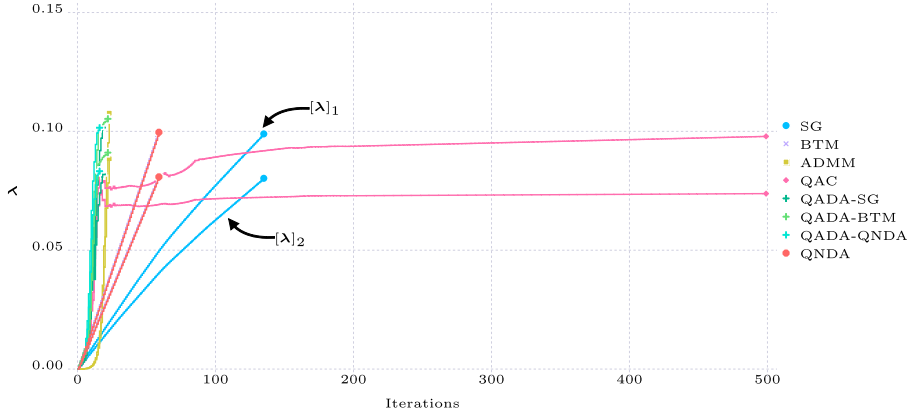
for all benchmark problems, i.e., the relative difference between the objective value of a feasible primal solution obtained for a value of the dual variables $\boldsymbol{\lambda}$ and the corresponding value of the dual function. As weak duality still holds, the value of the dual function provides a lower bound on the global optimum of the primal problem. Thus,



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

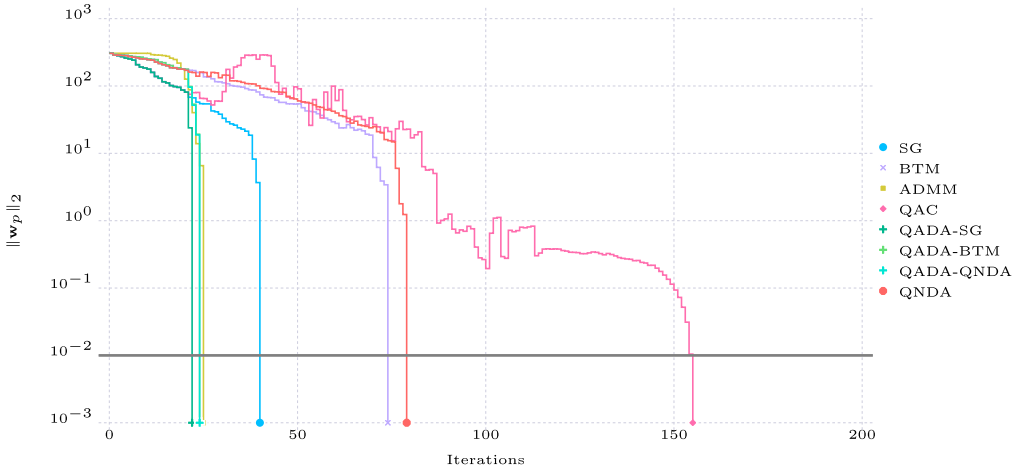


(b) Evolution of the primal and dual residuals.

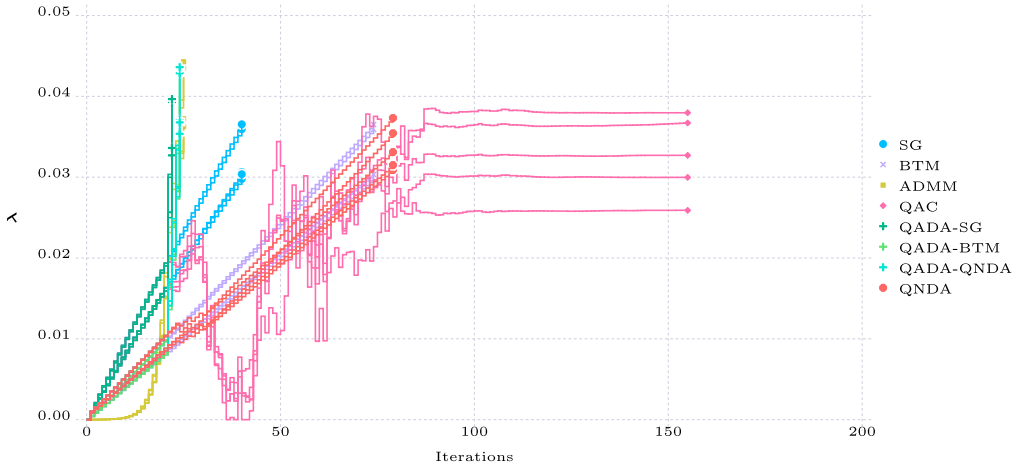


(c) Evolution of the dual varialbes.

Fig. 11. Results of the distributed optimization of problem $\text{MIQP}^{(7)}_{(100,2)}$.



(a) Evolution of primal and dual residuals.



(b) Evolution of the dual varialbes.

Fig. 12. Results of the distributed optimization of problem $\text{MIQP}^{(9)}_{(500,5)}$.

the relative DG 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 a poor performance of the QAC algorithm. One such instance is shown in Fig. 11 for benchmark problem $\text{MIQP}^{(7)}_{(100,2)}$. The surface plots in Fig. 11a show that the primal residual $\|\mathbf{w}_p\|_2^2$, which is approximated by QAC,

Table 3
Computational results for all MIQP benchmark problems with $N_s = 500$ and $n_{\mathbf{x}} = 5$. The central solution shows the relative integrality gap and the computation time while the distributed optimization algorithms show the relative duality gap and the computation time.

	Central		SG		BTM	
	rel. IG [%]	T_{comp} [s]	rel. DG [%]	T_{comp} [s]	rel. DG [%]	T_{comp} [s]
$\text{MIQP}^{(1)}_{(500,5)}$	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	
	rel. DG [%]	T_{comp} [s]	rel. DG [%]	T_{comp} [s]	rel. DG [%]	T_{comp} [s]
$\text{MIQP}^{(1)}_{(500,5)}$	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	
	rel. DG [%]	T_{comp} [s]	rel. DG [%]	T_{comp} [s]	rel. DG [%]	T_{comp} [s]
$\text{MIQP}^{(1)}_{(500,5)}$	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**

is nonsmooth and nonconvex. In comparison, the dual function is always concave and the nonsmoothness is less profound. Therefore, QADA and QNDA are able to compute a better smooth approximation and to 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 [60]. This also holds for the nonsmoothness of the response surfaces [69]. Thus, the performance of the approximation-based algorithms tends to improve for larger problems. An example is shown in Fig. 12 for benchmark problem $\text{MIQP}^{(9)}_{(500,5)}$ where all algorithms converge to a feasible solution. The evolution of the primal residual (Fig. 12a) and the dual variables (Fig. 12b) indicate that QAC tends to oscillate, leading to a slower convergence. Fig. 12 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 Sec. 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 intractable. 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 algorithm, 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. In order to assess the quality of the solutions obtained through 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 Table 3. Remarkably, Gurobi was 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 DG is usually better than the relative IG 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 samping 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

$$\min_{\mathbf{x}_1,\ldots,\mathbf{x}_{N_s}} \sum_{i\in\mathcal{I}} f_i(\mathbf{x}_i), \tag{118a}$$

$$\text{s. t.} \sum_{i\in\mathcal{I}} \mathbf{A}_i\mathbf{x}_i = \mathbf{0}, \tag{118b}$$

$$\mathbf{x}_i^T\mathbf{G}_i\mathbf{x}_i \leq p_i^2,\ \forall i\in\mathcal{I} \tag{118c}$$

$$\mathbf{x}_i^{\text{LB}} \leq \mathbf{x}_i \leq \mathbf{x}_i^{\text{UB}},\ \forall i\in\mathcal{I}, \tag{118d}$$

are considered. The objective functions $f_i(\mathbf{x}_i)$ are all convex functions (inside the feasible set of (118)) and are summarized in Table B.6 in the appendix alongside the bounds (118d) 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 (118b) were drawn from the 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 (118c), 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 $n_{\mathbf{x}} = n_{\mathbf{x}_i}$ and the number of system-wide constraints is equal to the number of variables of each subproblem, $n_{\mathbf{b}} = n_{\mathbf{x}}$. 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 (118b) 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 Table A.5). All parameters were set by trial and error, in order to find the parameters that result in the most converged

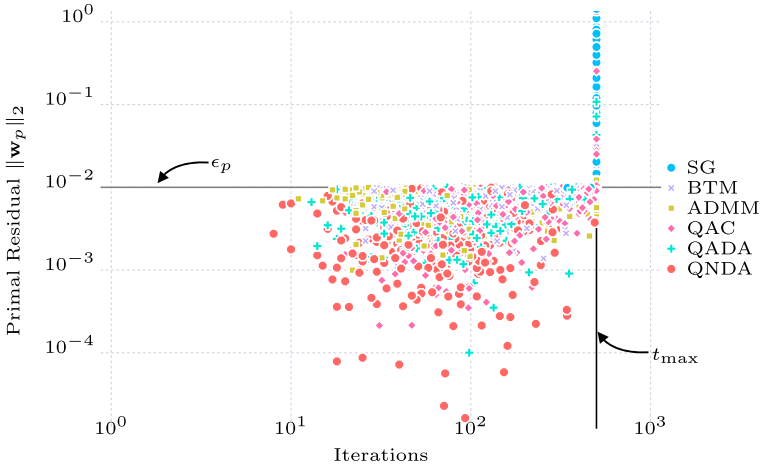


Fig. 13. Values of the primal residuals upon termination for the distributed convex programs. Each data point represents an algorithm applied to a benchmark problem (cf. Table D.9).

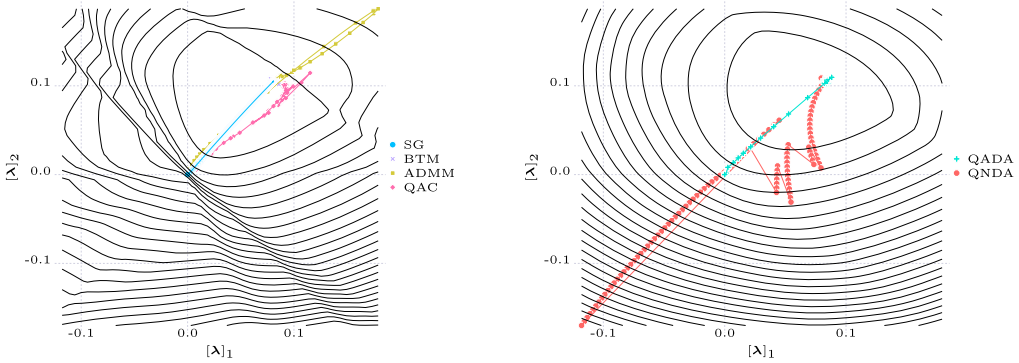
Table 4
Summary of the results for the coordination of convex problems (mean values of the converged instances only), $\overline{t}$: mean number of iterations until convergence, $\overline{T_{\text{comp}}}$: mean computation time of converged runs (in s), $\overline{\|w_p\|_2}$: mean primal residual of converged runs ($\times 10^{-3}$), $\%_c$: percentage of converged runs within $t_{\max}$ iterations.

Algorithm	$\overline{t}$	$\overline{T_{\text{comp}}}$	$\overline{\|w_p\|_2}$	$\%_c$
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**

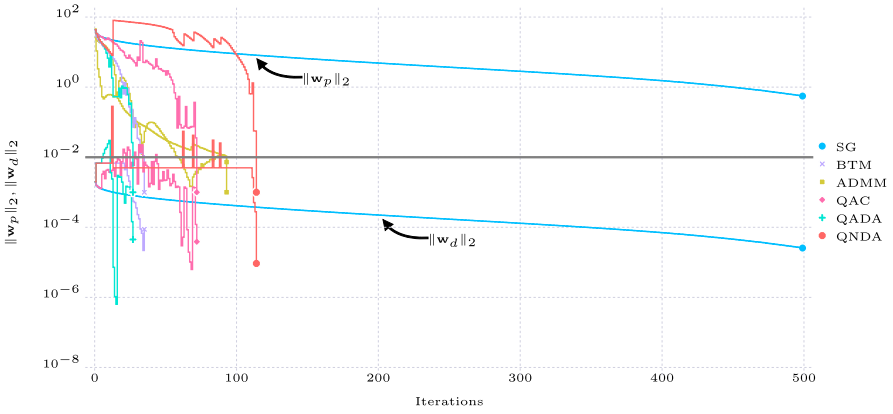
benchmark problems. As the previous results showed that the performance of QADA is not heavily influenced by the algorithm used for the initial sampling phase, only the initialization with QNDA is considered in the following. For the sake of brevity this is denoted by QADA instead of QADA-QNDA.

6.3.2. Results for distributed convex problems

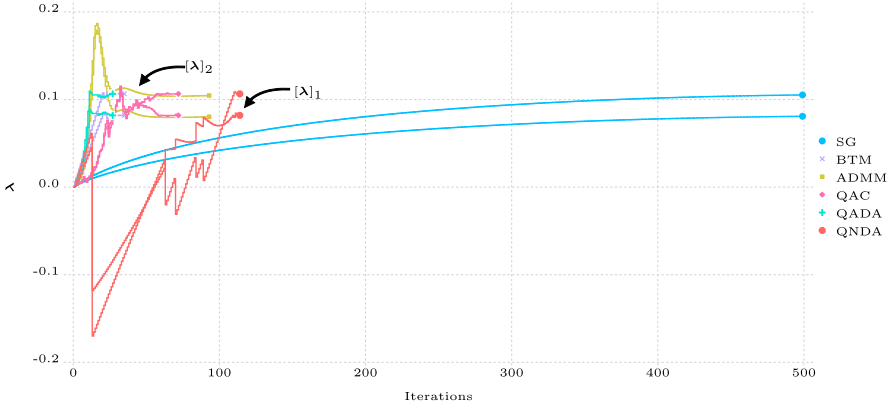
The results for the distributed optimization of the convex benchmark problems are illustrated in Fig. 13 and summarized in Table 4. The mean computation time for a single update step of the primal and dual variables ($\overline{T_{\text{comp}}}/\overline{t}$) increases compared to the distributed QPs. This is due to the fact that the solution of the general convex subproblems is computationally more expensive than that of the QPs. This is also due to the fact that 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 [42]. Table 4 shows that QNDA achieves the largest num-



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

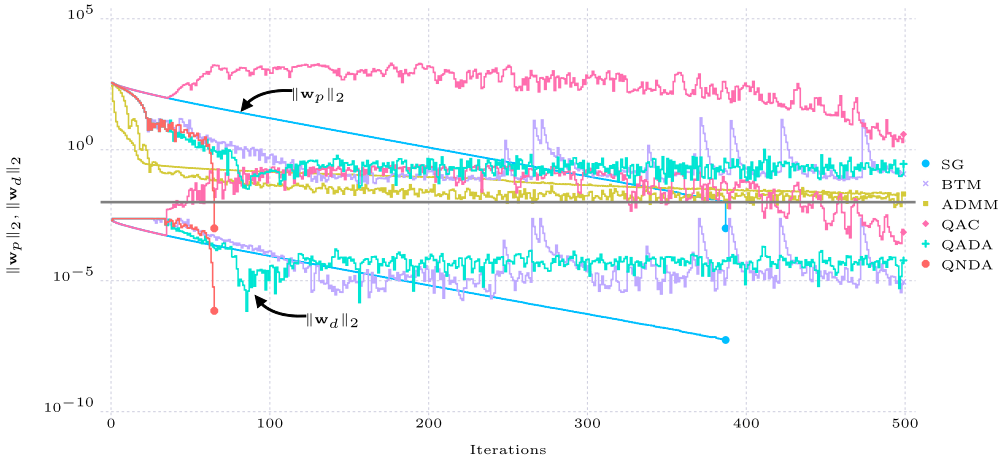


(b) Evolution of the primal and dual residuals.

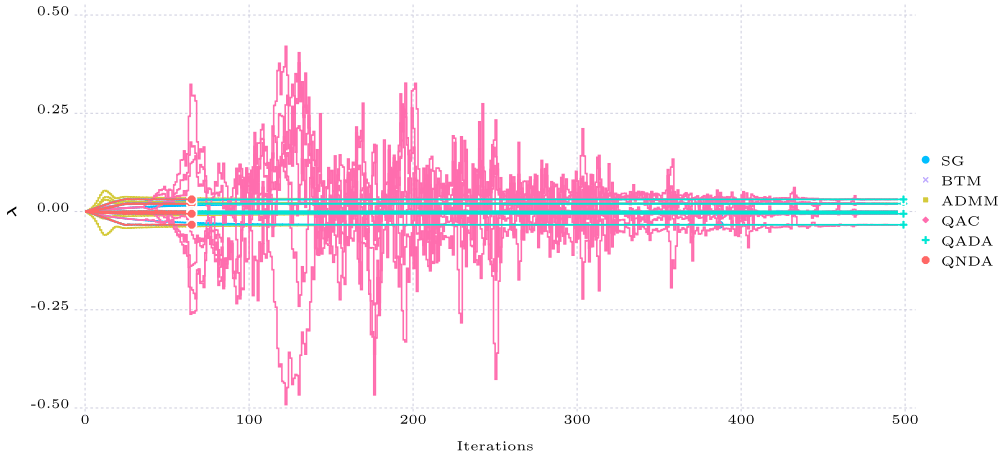


(c) Evolution of the dual varialbes.

Fig. 14. Results of the distributed optimization of problem Conv $^{(6)}_{(64,2)}$.



(a) Evolution of primal and dual residuals.



(b) Evolution of the dual varialbes.

Fig. 15. Results of the distributed optimization of problem $\text{Conv}^{(4)}_{(256,7)}$.

ber of converged benchmark problems and requires the least number of iterations and the second least computation time. In comparison, QADA performs rather poorly, only outperforming the subgradient method. Interestingly, QADA performs better for relatively few subproblems and system-wide constraints. One such instance is depicted in Fig. 14 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. 14a and 14c also show that ADMM overshoots in the beginning, which might indicate that a less aggressive tuning could lead to a better convergence.

The QNDA algorithm outperforms all other algorithms for large benchmark problems. One such instance is depicted in Fig. 15 for problem $\text{Conv}^{(4)}_{(256,7)}$. No algorithm converges,

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 actually find the optimum. This is also illustrated in Fig. 15b, where it can be seen that all algorithms terminate close to the optimal dual variables. This behavior is indicative for 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 few iterations.

7. Summary and outlook

In this paper, two new efficient dual decomposition-based distributed optimization algorithms were presented. 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 in order 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 benchmark problems showed the efficiency of the proposed algorithms. A remarkable result in our view is that dual decomposition-based distributed optimization algorithms, and especially QADA, can be used to speed up the solution of mixed-integer programs were a centralized solution does not converge in reasonable amounts of time. While the QADA algorithm showed superior performance for distributed MIQPs, the QNDA algorithm outperformed the other algorithms for general distributed convex problems. For distributed QPs the QADA and QNDA algorithms showed a similar performance, outperforming other algorithms.

When comparing the algorithms, it must not be forgotten that they use different amounts of information that is exchanged between the coordinator and the local optimizations. The subgradient method and QAC only need information on the residual of the system-wide constraints, the bundle method, QADA and QNDA additionally need the values of the dual function while ADMM requires a modification of the local problems and the communication of auxiliary variables. Depending on the application, this exchange of additional information and the modification of the local problems may be problematic or not.

Future research will progress in two directions, algorithmic improvements and applications. In terms of the performance of the QADA algorithm, the solution of the regression problem is a key component. If this problem is not well conditioned, the approximation can fail and lead to wrong update steps. Therefore, a preconditioning strategy of the regression problem could improve the subsequent update steps. Wenzel et al. [70] proposed a method to evaluate the quality of regression data sets based on their Λ-poisedness. The selection of the regression data could be performed by directly optimizing this criterion. The quality criterion could also be used to perform exploratory moves, in order to enhance the approximation data [25]. Furthermore, the update steps of the dual variables require the solution of an optimization problem. For badly conditioned or badly selected data the update problems can become nonconvex in some cases for both algorithms. Guaranteeing the convexity of the update problem would significantly impact the computational performance. This could be achieved by posing the problem of computing the surrogate function as a semi-definite programming problem. The aforementioned algorithmic improvements could also enhance the performance of the QAC algorithm. The results for the distributed convex problems showed that the QADA algorithm converges faster to the vicinity of the optimum. However, the QNDA algorithm is more effective in actually finding the optimum once it reached its vicinity. The strengths of both algorithms could be combined by alternating between different update strategies for the dual variables. Finally, in this paper no assumptions where made on the topology of the network of subproblems. The network topology is reflected in the matrices $\mathbf{A}_i$. Explicitly considering the sparsity structure of the system-wide constraints could lead to the elimination of the need for a central coordinator and the application of the algorithms to network optimization.

In terms of applications, the QAC algorithm was developed in the context of market-like coordination of coupled production plants with shared resources [68]. Dual decomposition-based distributed optimization can also be applied in other areas. Model predictive control (MPC) has been a major field of research in distributed optimization [15]. Dual decomposition-based distributed MPC could be applied for systems with slow dynamics and long sampling times where a certain degree of privacy between the subsystems is required [6]. Another potential application for dual decomposition-based distributed optimization is demand-side management (DSM), where multiple energy consumers are connected to a common grid [34]. This is an example of autonomous subsystems sharing a limited resource (energy). DSM problems are usually posed as mixed-integer problems, which results in additional challenges for distributed optimization, as discussed in this paper. Privacy also plays an important role in machine learning applications when training data is stored in a decentralized fashion and cannot be shared. Dual decomposition can for instance be used for the distributed training of support vector machines [21] (convex problems) or for distributed clustering [45] (integer problems) by training individual models and coupling their parameters through consensus constraints. In the context of machine learning, parallelization of the

algorithms on specialized hardware like GPUs can provide further computational benefits.

CRediT authorship contribution statement

Vassilios Yfantis: Conceptualization, Methodology, Software, Design and evaluation of experiments, Writing - Original Draft, Visualization, **Simon Wenzel**: Conceptualization, Software, Investigation, Writing - Review & Editing, **Achim Wagner**: Writing - Review & Editing, Supervision, **Martin Ruskowski**: Writing - Review & Editing, Supervision, Project administration, Funding acquisition, **Sebastian Engell**: Conceptualization, Methodology, Writing - Review & Editing, Supervision

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The authors would like to thank Nigora Gafur for providing helpful comments on the manuscript.

Appendix A. Summary of the used parameters

Table A.5 contains all parameters used for the dual decomposition-based distributed optimization algorithms in Sec. 6.

Appendix B. Convex objective functions

All objective functions used for the convex benchmark problems in Sec. 6.3 are summarized in Table B.6.

Appendix C. Benchmark problems

All benchmark problems used in Sec. 6 can be found under https://github.com/VaYf/EJCOMP_Benchmark_Problems.

Appendix D. Summaries of computational results

Tables D.7, D.8 and D.9 summarize the results for the distributed optimization of the QP, MIQP and general convex benchmark problems respectively.

Table A.5
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 tolerances	All
ϵ_d	10^{-2}	–	dual residual convergence tolerances	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 (50)	ADMM
τ_{decr}	1.25	1.25	see (50)	ADMM
μ	10	10	see (50)	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 (66)	QAC, QADA
$\overline{s}_i$	$n_{\mathbf{b}} \times 10^{-3}$	$n_{\mathbf{b}} \times 10^{-4}$	ellipsoid parameter (66)	QAC, QADA
$\underline{\gamma}$	0.1	1	ellipsoid parameter (69)	QAC, QADA

Table B.6
Used convex objective functions for the distributed convex programs.

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

Table D.7
Results for the coordination of QPs (mean values of the converged instances only), $\overline{t}$: 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.

QP	SG				ADMM				BTM			
	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
Mean	**384.74**	**9.88**	**308.36**	**16.83**	**179.55**	**6.81**	**145.54**	**82.3**	**196.95**	**7.47**	**160.07**	**95.11**
(2, 2)	488.0	9.96	390.71	2	19.88	6.6	15.92	100	90.24	5.68	72.7	100
(4, 2)	435.6	9.93	348.76	10	32.22	5.93	25.81	100	79.78	5.78	64.28	100
(4, 3)	–	–	–	0	44.62	5.73	35.75	100	146.82	6.52	118.31	100
(4, 4)	–	–	–	0	49.22	6.14	39.43	100	177.52	6.85	143.1	100
(8, 2)	335.67	9.93	268.76	6	37.32	6.25	29.9	100	65.1	6.33	52.45	100
(8, 3)	344.0	9.98	275.45	2	57.84	6.28	46.33	100	112.74	7.07	90.85	100
(8, 4)	–	–	–	0	69.96	6.41	56.05	100	167.06	6.99	134.69	100
(8, 5)	–	–	–	0	80.42	5.29	64.43	100	185.76	7.56	149.77	100
(8, 6)	–	–	–	0	97.48	5.58	78.14	100	239.9	7.45	193.63	98
(8, 7)	–	–	–	0	117.02	5.25	93.83	100	268.96	7.47	217.47	94
(8, 8)	–	–	–	0	143.42	4.34	115.02	100	320.42	7.46	260.22	86
(16, 2)	307.14	9.85	245.96	14	45.26	7.55	36.26	100	70.36	6.04	56.7	100
(16, 3)	–	–	–	0	57.78	8.22	46.29	100	103.48	6.96	83.39	100
(16, 4)	–	–	–	0	76.16	8.61	61.02	100	140.16	6.74	112.99	100
(16, 5)	–	–	–	0	90.02	8.22	72.14	100	177.34	8.02	143.0	100
(16, 6)	–	–	–	0	112.88	7.09	90.52	100	233.92	7.69	188.8	100
(16, 7)	–	–	–	0	130.1	7.56	104.37	100	246.31	7.81	199.12	96
(16, 8)	–	–	–	0	132.5	7.18	106.31	100	303.51	7.77	246.35	94
(16, 9)	–	–	–	0	162.78	6.66	130.64	100	356.37	7.77	291.76	92
(16, 10)	–	–	–	0	169.2	6.38	135.8	100	383.29	7.95	319.84	82
(32, 2)	407.57	9.9	326.43	14	62.68	7.98	50.24	100	65.5	6.16	52.8	100
(32, 3)	384.0	9.74	307.54	4	68.76	8.68	55.13	100	96.04	7.12	77.4	100
(32, 4)	–	–	–	0	70.06	8.92	56.15	100	130.08	7.12	104.88	100
(32, 5)	–	–	–	0	90.48	9.05	72.55	100	166.04	7.68	133.9	100
(32, 6)	–	–	–	0	117.22	9.12	94.06	100	207.43	8.06	167.42	98
(32, 7)	–	–	–	0	119.48	9.05	95.91	100	237.71	7.7	192.17	96
(32, 8)	–	–	–	0	140.3	8.96	112.66	100	291.64	7.82	236.7	100
(32, 9)	–	–	–	0	158.14	9.28	127.01	100	328.81	8.04	268.97	94
(32, 10)	–	–	–	0	175.53	9.09	141.04	98	352.24	8.09	292.34	92
(64, 2)	269.73	9.81	216.11	30	119.82	7.55	96.14	100	55.82	6.16	45.01	100
(64, 3)	401.6	9.91	321.79	10	167.74	7.27	134.71	100	80.36	6.73	64.82	100
(64, 4)	232.0	9.94	185.91	2	154.4	7.12	124.04	96	116.08	7.37	93.65	98
(64, 5)	462.0	9.9	370.21	2	180.18	7.74	144.85	100	155.72	7.76	125.68	100

Table D.7 (*continued*)

QP	SG				ADMM				BTM			
	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
(64, 6)	–	–	–	0	184.04	7.15	148.13	98	204.92	7.63	165.55	100
(64, 7)	–	–	–	0	160.06	8.19	128.85	100	225.84	7.91	182.7	98
(64, 8)	–	–	–	0	173.29	8.74	139.54	98	268.52	8.05	217.97	92
(64, 9)	–	–	–	0	154.56	9.21	124.49	100	297.57	8.19	243.03	98
(64, 10)	–	–	–	0	165.35	9.76	133.2	98	327.05	7.87	270.27	88
(128, 2)	336.5	9.74	269.66	60	195.11	6.21	156.95	94	47.18	6.18	38.05	100
(128, 3)	357.54	9.86	286.51	26	265.8	5.87	214.33	82	71.42	7.06	57.59	100
(128, 4)	396.5	9.89	317.75	16	297.69	6.26	240.2	78	107.2	7.74	86.48	100
(128, 5)	483.0	9.85	387.09	2	329.59	5.98	266.25	64	150.42	7.74	121.39	100
(128, 6)	–	–	–	0	363.05	4.86	293.65	42	184.9	7.95	149.3	100
(128, 7)	430.5	9.91	345.11	4	388.74	4.2	314.79	38	228.14	8.17	184.49	98
(128, 8)	–	–	–	0	393.42	5.29	318.96	24	270.06	8.21	219.05	94
(128, 9)	–	–	–	0	379.73	5.41	308.82	22	302.5	8.15	246.89	84
(128, 10)	–	–	–	0	347.78	5.51	283.17	18	328.77	8.44	271.64	86
(256, 2)	264.78	9.75	212.3	72	285.66	4.99	233.81	76	47.62	7.05	38.44	100
(256, 3)	346.23	9.87	277.61	52	309.54	4.76	254.01	56	79.18	6.99	63.92	100
(256, 4)	367.05	9.87	294.31	40	332.08	4.94	272.93	24	115.52	7.21	93.27	96
(256, 5)	413.67	9.9	331.73	18	365.67	4.16	301.43	24	161.58	8.1	130.5	96
(256, 6)	444.2	9.9	356.26	10	449.0	6.89	370.87	8	213.37	8.17	172.5	92
(256, 7)	356.5	9.92	285.94	4	446.5	4.47	369.52	4	275.6	8.2	223.0	90
(256, 8)	487.0	9.83	390.68	2	358.0	4.02	296.67	2	311.61	8.13	252.73	76
(256, 9)	483.0	9.93	388.06	2				0	317.19	8.61	258.52	62
(256, 10)	–	–	–	0	–	–	–	0	340.5	8.78	281.28	56

(*continued on next page*)

Table D.7 (*continued*)

QP	QAC				QADA-SG				QADA-BTM			
	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
Mean	**199.24**	**2.15**	**167.73**	**57.32**	**133.49**	**3.31**	**139.5**	**96.46**	**128.10**	**3.26**	**135.22**	**96.96**
(2, 2)	71.59	1.7	59.78	98	92.1	1.48	94.36	98	79.29	1.16	81.5	98
(4, 2)	59.35	0.12	49.52	98	62.96	0.83	63.07	98	54.2	0.69	54.53	100
(4, 3)	133.77	0.61	111.15	88	98.19	0.84	94.21	96	83.78	1.0	80.78	100
(4, 4)	172.55	0.73	142.92	84	103.83	0.84	98.74	96	99.24	0.65	94.21	98
(8, 2)	61.18	0.13	50.89	100	55.68	0.34	54.11	100	50.18	0.75	49.0	100
(8, 3)	114.68	0.59	95.07	88	70.14	0.77	66.6	98	74.5	0.97	71.98	96
(8, 4)	198.28	0.57	164.05	72	102.63	2.0	96.3	98	90.65	1.52	85.24	96
(8, 5)	187.36	0.3	154.64	50	107.82	1.72	99.41	100	101.38	1.58	94.36	100
(8, 6)	232.47	1.18	191.63	38	135.53	2.73	126.11	98	126.63	3.07	118.89	98
(8, 7)	267.89	1.12	220.67	18	178.96	3.0	172.32	96	166.5	2.12	161.86	96
(8, 8)	283.17	1.59	233.67	12	217.6	2.75	219.73	94	204.39	2.61	207.13	92
(16, 2)	48.09	0.61	39.89	92	46.83	0.29	44.31	96	38.9	0.98	36.99	98
(16, 3)	90.12	0.18	74.5	82	72.41	1.4	68.11	98	77.4	1.04	73.76	100
(16, 4)	173.22	0.53	142.89	64	90.72	1.85	84.46	100	87.06	2.26	80.52	100
(16, 5)	212.15	1.27	174.71	54	106.08	3.24	96.94	100	99.9	2.19	92.27	100
(16, 6)	300.0	1.21	246.79	24	147.48	3.12	137.22	100	140.04	3.15	130.28	100
(16, 7)	262.0	4.5	215.46	8	181.55	4.18	174.51	94	169.7	3.15	163.62	92
(16, 8)	329.0	4.63	271.44	4	203.02	4.0	203.03	92	191.09	4.52	189.8	92
(16, 9)	412.0	0.0	342.64	2	266.65	4.23	283.6	98	254.27	3.48	272.58	96
(16, 10)	–	–	–	0	298.77	4.33	347.32	86	287.04	4.55	331.77	90
(32, 2)	42.54	0.21	35.13	92	46.56	0.91	43.19	100	38.42	0.82	35.46	100
(32, 3)	84.18	0.63	69.35	80	62.56	1.89	58.2	100	57.86	1.16	54.2	98
(32, 4)	143.88	1.45	118.33	82	75.53	2.66	69.08	98	68.57	3.03	62.98	98
(32, 5)	248.36	1.41	204.51	56	118.35	4.32	107.45	96	105.58	3.75	96.52	96
(32, 6)	253.23	0.85	208.03	26	138.91	3.02	129.05	92	133.7	3.56	123.52	94
(32, 7)	412.5	3.98	339.07	8	169.02	3.28	160.11	96	164.98	3.43	158.62	96
(32, 8)	244.0	–	201.09	2	198.66	3.84	198.02	100	200.6	3.12	198.95	100
(32, 9)	–	–	–	0	241.02	3.83	255.18	92	229.2	4.03	243.75	90
(32, 10)	–	–	–	0	270.86	4.87	307.4	88	263.07	4.28	303.17	92
(64, 2)	45.31	0.59	41.39	96	31.62	1.02	43.44	100	33.58	1.55	44.82	100
(64, 3)	85.67	1.21	77.27	90	49.58	1.94	71.2	100	51.44	1.0	73.93	100
(64, 4)	139.49	2.05	124.65	82	69.34	3.83	93.72	94	68.62	3.43	94.95	96
(64, 5)	220.75	1.68	197.5	48	110.73	3.04	144.17	98	101.2	3.72	135.79	98
(64, 6)	359.5	4.08	320.61	32	130.98	4.59	174.69	98	121.0	4.6	160.05	98
(64, 7)	429.0	5.58	381.52	8	160.73	4.85	221.55	96	150.56	4.47	212.24	100
(64, 8)	–	–	–	0	187.64	4.01	246.8	90	183.59	4.49	257.16	92

Table D.7 (*continued*)

QP	QAC				QADA-SG				QADA-BTM			
	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
(64, 9)	–	–	–	0	211.48	4.66	223.92	96	207.84	5.44	223.83	98
(64, 10)	–	–	–	0	248.17	5.04	285.49	96	243.56	5.43	282.29	96
(128, 2)	33.55	0.91	27.63	98	29.16	1.02	26.11	100	25.9	1.75	23.48	100
(128, 3)	74.86	1.84	61.54	100	56.67	1.64	52.49	98	40.96	1.77	38.04	98
(128, 4)	144.81	3.16	119.0	84	67.94	4.62	60.51	98	68.24	3.34	62.19	100
(128, 5)	255.65	3.19	209.99	74	91.12	4.59	81.62	100	88.96	4.44	80.45	100
(128, 6)	317.08	3.64	260.34	26	120.16	4.17	109.3	98	118.29	5.03	109.91	98
(128, 7)	447.43	6.88	367.6	14	155.96	5.05	150.04	94	151.77	6.01	146.24	96
(128, 8)	–	–	–	0	176.44	5.64	174.09	96	180.35	5.16	180.84	98
(128, 9)	–	–	–	0	216.81	5.78	229.86	96	216.63	5.68	232.67	98
(128, 10)	–	–	–	0	258.61	4.93	292.52	92	254.94	4.75	294.96	94
(256, 2)	30.66	1.65	27.41	94	24.26	2.26	31.32	100	24.92	2.11	31.03	100
(256, 3)	65.47	3.01	53.86	68	44.76	3.11	40.82	98	40.41	3.26	37.57	98
(256, 4)	136.28	3.63	120.78	80	64.12	4.62	57.22	100	63.68	5.26	57.08	100
(256, 5)	214.71	4.14	189.67	68	89.23	4.97	80.8	96	86.22	4.88	77.67	98
(256, 6)	374.73	5.77	330.63	30	111.79	5.25	104.01	94	109.28	4.76	102.81	92
(256, 7)	354.0	7.29	311.04	8	136.79	6.07	130.12	94	137.23	5.16	131.69	94
(256, 8)	–	–	–	0	181.6	5.34	184.46	96	181.83	5.65	184.45	96
(256, 9)	–	–	–	0	223.89	5.85	240.29	94	228.23	6.11	248.18	94
(256, 10)	–	–	–	0	265.65	5.18	309.53	92	256.17	4.45	299.57	92

(*continued on next page*)

Table D.7 (*continued*)

QP	QADA-QNDA				QNDA			
	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
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
(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

Table D.7 (*continued*)

QP	QADA-QNDA				QNDA			
	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
Mean	**127.23**	**3.29**	**135.47**	**96.57**	**134.31**	**3.94**	**126.4**	**98.5**
(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

Table D.8
Results for the coordination of MIQPs (mean values of the converged instances only), $\overline{t}$: 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.

MIQP	SG				ADMM				BTM			
	$\overline{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\overline{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\overline{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$	$\%_c$
Mean	**86.69**	**1.66**	**69.88**	**99.5**	**25.06**	**2.22**	**21.16**	**100**	**80.52**	**1.66**	**65.41**	**100**
(100, 2)	58.1	0.78	46.73	100	23.4	1.11	18.95	100	29.3	0.81	23.72	100
(100, 3)	73.4	1.65	59.11	100	21.2	1.69	17.15	100	66.1	1.67	53.59	100
(100, 4)	60.7	4.81	48.85	100	23.5	5.2	19.24	100	98.7	4.71	80.04	100
(100, 5)	51.6	9.76	41.61	100	22.7	10.09	18.43	100	124.0	9.64	100.67	100
(200, 2)	113.33	0.35	90.94	90	25.6	1.08	20.8	100	57.6	0.5	46.54	100
(200, 3)	66.6	0.84	53.5	100	24.5	1.06	20.07	100	57.6	0.86	46.56	100
(200, 4)	146.5	2.9	118.08	100	23.7	3.91	19.66	100	157.5	2.98	127.89	100
(200, 5)	65.1	2.97	52.54	100	23.2	3.61	19.11	100	124.6	2.94	101.29	100
(300, 2)	129.9	0.31	104.55	100	27.0	0.88	22.31	100	44.7	0.32	36.24	100
(300, 3)	66.2	0.58	53.34	100	25.4	0.79	21.19	100	51.9	0.58	42.15	100
(300, 4)	71.2	1.25	57.54	100	24.3	1.49	20.39	100	86.0	1.23	69.97	100
(300, 5)	61.1	1.56	49.37	100	24.7	1.82	20.82	100	110.8	1.53	90.18	100
(400, 2)	89.4	0.13	72.04	100	28.4	0.59	23.93	100	34.7	0.16	28.15	100
(400, 3)	165.2	0.76	133.14	100	26.9	2.26	23.16	100	99.7	0.76	80.91	100
(400, 4)	77.0	1.06	62.28	100	25.4	1.99	21.89	100	84.2	1.06	68.55	100
(400, 5)	54.3	1.17	43.94	100	24.2	1.63	21.03	100	93.9	1.15	76.49	100
(500, 2)	170.3	0.16	137.18	100	29.9	1.17	25.84	100	55.2	0.18	44.75	100
(500, 3)	58.9	0.28	47.55	100	26.3	0.57	23.37	100	42.6	0.28	34.56	100
(500, 4)	76.4	0.63	61.81	100	25.6	1.31	22.85	100	75.4	0.63	61.44	100
(500, 5)	78.5	1.21	63.56	100	25.4	2.08	22.97	100	115.9	1.21	94.52	100

Table D.8 (*continued*)

MIQP	QAC				QADA-SG				QADA-BTM			
	$\overline{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\overline{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\overline{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$	$\%_c$
Mean	**59.4**	**2.13**	**52.83**	**86.0**	**19.37**	**2.54**	**18.37**	**100**	**20.78**	**3.53**	**18.62**	**100**
(100, 2)	63.0	0.03	56.64	20	23.4	1.59	34.3	100	17.6	0.99	26.82	100
(100, 3)	81.0	1.66	71.81	60	23.4	2.1	32.31	100	17.4	2.8	18.07	100
(100, 4)	82.14	4.49	74.82	70	19.8	5.09	17.21	100	26.1	5.58	22.19	100
(100, 5)	55.6	11.16	47.98	100	22.7	13.42	19.53	100	32.1	27.97	27.19	100
(200, 2)	101.17	0.44	92.76	60	15.2	1.01	17.43	100	12.5	0.83	11.05	100
(200, 3)	95.5	0.92	85.42	100	18.3	1.46	17.62	100	16.9	1.03	15.03	100
(200, 4)	59.22	4.91	52.42	90	23.3	4.56	19.73	100	27.0	7.36	23.07	100
(200, 5)	42.2	3.29	36.12	100	23.5	6.08	19.94	100	29.4	3.79	24.91	100
(300, 2)	89.2	0.37	80.62	100	15.6	0.4	14.61	100	10.5	0.38	9.22	100
(300, 3)	48.6	0.74	42.99	100	17.3	0.92	16.6	100	15.7	0.85	14.18	100
(300, 4)	34.89	2.26	29.85	90	18.8	2.04	15.88	100	22.1	2.55	18.53	100
(300, 5)	35.5	1.61	29.91	100	22.9	2.04	19.14	100	28.9	2.85	24.28	100
(400, 2)	65.0	0.2	59.38	80	12.5	0.19	10.87	100	12.4	0.18	12.49	100
(400, 3)	30.57	2.98	26.28	70	19.4	2.44	16.9	100	18.6	1.64	15.97	100
(400, 4)	27.6	1.81	23.37	100	18.9	1.29	15.97	100	23.3	1.66	19.6	100
(400, 5)	39.8	1.57	33.7	100	23.0	1.45	19.08	100	29.5	5.01	24.96	100
(500, 2)	51.33	0.43	46.84	90	13.0	0.56	11.32	100	12.2	0.29	10.64	100
(500, 3)	79.4	0.36	71.45	100	16.7	0.49	15.42	100	15.0	0.5	13.3	100
(500, 4)	30.3	1.6	25.99	100	16.6	1.44	13.93	100	19.2	1.53	16.23	100
(500, 5)	76.0	1.71	68.26	90	23.1	2.14	19.67	100	29.1	2.78	24.57	100

(*continued on next page*)

Table D.8 (*continued*)

MIQP	QADA-QNDA				QNDA			
	$\overline{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\overline{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$	$\%_c$
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

Table D.9
Results for the coordination of convex problems (mean values of the converged instances only), $\overline{t}$: 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.

Conv	SG				ADMM				BTM			
	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
Mean	**342.51**	**9.72**	**310.45**	**46.07**	**117.44**	**7.44**	**104.97**	**93.0**	**160.29**	**7.8**	**139.29**	**78.26**
(2, 2)	–	–	–	0	27.2	8.2	22.26	100	128.2	6.61	105.05	100
(4, 2)	–	–	–	0	25.8	6.92	21.31	100	98.0	7.62	86.08	100
(4, 3)	–	–	–	0	37.6	7.04	31.05	100	107.44	7.25	89.48	90
(4, 4)	–	–	–	0	38.6	7.34	31.69	100	148.1	6.89	123.98	100
(8, 2)	–	–	–	0	42.8	7.15	35.25	100	74.0	5.83	61.39	80
(8, 3)	–	–	–	0	51.7	8.39	42.51	100	99.4	5.88	82.6	100
(8, 4)	–	–	–	0	50.1	7.67	41.44	100	108.6	7.45	90.65	100
(8, 5)	–	–	–	0	55.0	5.93	45.93	100	103.9	7.66	86.58	100
(8, 6)	–	–	–	0	64.0	6.26	53.24	100	145.6	6.87	121.68	100
(8, 7)	–	–	–	0	83.8	5.54	69.81	100	199.5	8.29	167.84	100
(8, 8)	–	–	–	0	96.6	6.59	80.13	100	220.8	8.37	185.82	100
(16, 2)	464.0	9.92	381.03	10	35.4	7.66	29.7	100	46.9	4.8	39.15	100
(16, 3)	345.0	9.87	287.14	10	49.6	8.25	41.39	100	88.7	5.79	74.66	100
(16, 4)	–	–	–	0	66.5	7.91	55.52	100	109.7	7.73	92.0	100
(16, 5)	–	–	–	0	54.5	7.68	45.43	100	110.9	7.96	92.33	100
(16, 6)	–	–	–	0	67.8	7.55	56.55	100	172.5	8.06	144.77	100
(16, 7)	–	–	–	0	72.2	7.2	60.33	100	191.67	8.07	161.48	90
(16, 8)	–	–	–	0	77.5	7.73	65.06	100	209.3	8.34	176.79	100
(16, 9)	–	–	–	0	83.0	7.55	69.74	100	225.88	8.97	191.94	80
(16, 10)	–	–	–	0	107.5	7.17	90.01	100	333.67	9.13	286.26	60
(32, 2)	229.0	9.57	191.38	20	71.9	7.91	60.72	100	56.6	6.08	47.84	100
(32, 3)	429.5	9.94	361.66	20	51.4	8.22	43.72	100	62.9	5.82	53.21	100
(32, 4)	432.0	9.91	364.56	10	54.8	8.7	46.59	100	84.5	8.2	71.58	100
(32, 5)	–	–	–	0	50.4	8.16	42.87	100	138.3	8.1	117.08	100
(32, 6)	–	–	–	0	55.4	8.48	47.0	100	148.11	8.29	125.71	90
(32, 7)	–	–	–	0	69.7	8.5	59.0	100	239.83	8.33	203.34	60
(32, 8)	–	–	–	0	94.9	8.7	82.04	100	280.0	8.85	239.04	60
(32, 9)	–	–	–	0	88.22	8.15	74.93	90	344.0	8.89	293.6	50
(32, 10)	469.0	9.51	395.54	10	98.8	7.62	84.84	100	331.5	9.43	284.68	20
(64, 2)	332.0	9.92	281.45	20	88.0	6.59	75.52	100	68.5	6.61	58.72	100
(64, 3)	361.0	9.93	308.74	20	101.9	6.81	89.45	100	106.7	8.31	93.44	100
(64, 4)	332.0	9.92	285.71	20	62.0	8.0	53.59	100	101.56	7.6	87.82	90

(*continued on next page*)

Table D.9 (*continued*)

Conv	SG				ADMM				BTM			
	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
(64, 5)	–	–	–	0	98.2	7.56	84.45	100	138.0	8.36	119.09	80
(64, 6)	–	–	–	0	69.3	8.48	62.24	100	153.0	8.69	131.54	20
(64, 7)	411.5	9.93	353.78	20	94.5	8.53	83.29	100	231.5	8.77	201.97	40
(64, 8)	–	–	–	0	81.78	8.0	70.77	90	262.0	9.68	227.68	30
(64, 9)	–	–	–	0	106.22	8.16	91.6	90	–	–	–	0
(64, 10)	–	–	–	0	135.71	6.38	118.44	70	–	–	–	0
(128, 2)	268.5	9.84	238.85	60	172.89	5.99	153.89	90	38.7	6.5	34.32	100
(128, 3)	309.2	9.74	277.81	50	188.2	5.16	170.22	100	153.8	7.22	139.38	100
(128, 4)	268.0	8.92	252.23	30	200.1	6.92	184.32	100	140.75	8.39	128.16	40
(128, 5)	289.0	9.81	259.23	60	219.6	6.81	200.39	100	163.0	8.94	146.74	30
(128, 6)	409.0	9.94	360.74	20	238.4	7.7	211.37	100	210.0	7.81	186.94	30
(128, 7)	340.0	9.67	317.57	40	291.0	7.92	272.24	80	269.0	8.85	238.95	10
(128, 8)	323.2	9.71	289.48	50	283.88	8.26	259.66	80	–	–	–	0
(128, 9)	375.5	9.86	335.07	40	278.71	7.62	255.45	70	–	–	–	0
(128, 10)	429.0	9.4	397.06	40	185.86	8.79	167.21	70	–	–	–	0
(256, 2)	175.29	9.66	169.3	70	258.25	3.87	249.57	40	41.67	7.48	40.3	90
(256, 3)	243.38	9.63	238.09	80	441.57	6.63	427.84	70	188.5	7.9	186.54	100
(256, 4)	306.86	9.69	312.87	70	–	–	–	0	270.75	8.12	266.69	40
(256, 5)	256.7	9.46	257.95	100	453.0	7.61	437.14	10	227.5	9.8	222.26	20
(256, 6)	346.0	9.71	339.83	90	–	–	–	0	–	–	–	0
(256, 7)	339.75	9.73	332.69	80	–	–	–	0	–	–	–	0
(256, 8)	348.1	9.69	345.85	100	–	–	–	0	–	–	–	0
(256, 9)	358.62	9.7	362.9	80	–	–	–	0	–	–	–	0
(256, 10)	399.29	9.51	393.96	70	–	–	–	0	–	–	–	0

Table D.9 (*continued*)

Conv	QAC				QADA				QNDA			
	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
Mean	**207.16**	**6.55**	**186.9**	**81.52**	**123.52**	**6.95**	**149.39**	**75.1**	**97.13**	**4.58**	**114.71**	**97.14**
(2, 2)	134.0	4.87	114.82	50	132.62	5.58	129.93	80	98.12	4.06	84.55	80
(4, 2)	84.3	4.91	73.94	100	101.4	5.42	106.85	100	70.7	2.34	66.92	100
(4, 3)	146.0	6.12	127.36	90	115.7	5.37	120.47	100	132.56	3.88	127.71	90
(4, 4)	214.8	4.36	186.9	100	124.3	5.74	126.38	100	129.5	3.78	120.74	100
(8, 2)	89.88	4.42	78.25	80	122.88	5.12	128.55	80	71.12	3.54	65.38	80
(8, 3)	117.1	3.78	101.48	100	66.33	6.49	70.7	90	93.33	3.34	89.99	90
(8, 4)	157.2	5.31	136.01	100	125.1	6.73	128.81	100	90.1	3.84	87.91	100
(8, 5)	154.6	5.89	133.19	100	102.3	7.28	104.77	100	82.6	5.79	83.3	100
(8, 6)	300.0	6.57	258.93	70	121.8	6.22	127.86	100	104.7	3.7	103.46	100
(8, 7)	395.4	6.9	342.82	50	158.89	6.93	176.31	90	135.0	4.97	130.73	100
(8, 8)	333.4	6.88	286.44	50	184.6	8.11	224.27	100	142.5	4.49	143.87	100
(16, 2)	64.0	6.74	55.7	90	61.3	5.61	60.78	100	54.67	2.08	51.27	90
(16, 3)	138.1	5.51	121.15	100	80.6	5.17	86.59	100	100.3	2.55	100.26	100
(16, 4)	156.33	6.33	135.76	90	126.56	7.23	132.23	90	95.5	4.44	97.24	100
(16, 5)	178.1	8.52	153.3	100	90.6	7.46	94.26	100	82.8	4.26	81.0	100
(16, 6)	235.78	8.02	203.73	90	127.5	7.34	137.42	100	116.3	3.94	120.19	100
(16, 7)	354.75	8.14	307.39	80	144.3	7.64	161.92	100	125.8	5.37	127.2	100
(16, 8)	378.75	8.16	328.46	40	153.3	7.99	189.9	100	105.5	5.07	109.11	100
(16, 9)	439.5	7.08	382.58	20	185.78	7.94	256.69	90	137.6	3.87	151.06	100
(16, 10)	–	–	–	0	246.86	8.03	360.54	70	176.22	6.27	203.63	90
(32, 2)	55.88	3.54	49.33	80	63.3	3.82	67.01	100	66.9	2.07	64.47	100
(32, 3)	73.6	6.85	65.25	100	45.4	7.66	49.21	100	83.6	2.99	93.87	100
(32, 4)	90.5	5.96	80.05	100	82.5	8.2	88.87	100	64.2	4.81	68.41	100
(32, 5)	145.0	6.9	127.45	90	88.6	7.69	97.37	100	84.3	3.37	88.93	100
(32, 6)	222.3	7.22	194.63	100	109.1	8.44	118.63	100	96.0	4.99	100.6	100
(32, 7)	344.56	7.17	300.29	90	166.43	8.59	197.02	70	103.4	4.53	111.76	100
(32, 8)	433.67	8.96	378.71	30	130.57	7.72	160.34	70	108.2	4.32	137.21	100
(32, 9)	458.0	7.81	406.43	10	161.71	8.35	217.13	70	111.6	4.93	125.65	100
(32, 10)	–	–	–	0	150.33	7.8	200.96	30	164.1	6.41	222.04	100
(64, 2)	46.3	3.27	41.35	100	52.0	5.34	58.92	100	111.67	2.97	117.66	90
(64, 3)	64.0	5.64	57.8	90	61.11	7.37	69.24	90	73.7	4.21	89.08	100
(64, 4)	86.5	7.18	77.84	100	101.67	7.73	115.04	90	54.1	2.68	58.95	100
(64, 5)	190.5	6.76	171.24	100	198.62	7.03	228.11	80	92.5	5.01	109.2	100
(64, 6)	208.6	8.35	191.2	100	70.0	7.38	82.49	10	81.0	6.34	95.44	100
(64, 7)	371.2	9.12	339.08	100	161.75	9.4	207.02	40	115.2	5.31	163.71	100
(64, 8)	412.25	7.29	369.97	40	146.33	9.19	191.32	30	108.2	4.84	127.24	100

(*continued on next page*)

Table D.9 (*continued*)

Conv	QAC				QADA				QNDA			
	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\overline{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
$(64, 9)$	–	–	–	0	170.0	8.09	240.05	20	107.2	6.18	124.94	100
$(64, 10)$	–	–	–	0	198.0	6.57	326.19	10	121.5	5.09	149.86	100
$(128, 2)$	32.33	6.87	30.23	90	40.8	6.33	44.26	100	63.5	1.84	68.96	100
$(128, 3)$	54.5	4.59	51.73	100	67.44	7.38	81.15	90	66.9	3.59	81.81	100
$(128, 4)$	82.7	6.3	79.4	100	257.2	7.03	318.01	50	88.44	4.97	110.79	90
$(128, 5)$	126.44	7.39	118.16	90	275.67	9.13	336.91	30	62.89	4.35	70.19	90
$(128, 6)$	322.5	7.16	297.78	100	96.0	5.98	114.36	30	73.6	6.21	88.67	100
$(128, 7)$	398.25	8.76	369.4	40	104.0	0.7	133.61	10	94.8	6.06	123.52	100
$(128, 8)$	–	–	–	0	102.5	7.98	156.0	40	85.11	5.76	96.92	90
$(128, 9)$	–	–	–	0	–	–	–	0	119.1	6.2	181.47	100
$(128, 10)$	–	–	–	0	–	–	–	0	126.67	7.44	194.22	90
$(256, 2)$	33.8	4.87	34.64	100	36.8	6.5	43.9	100	46.3	3.62	60.89	100
$(256, 3)$	53.0	5.27	54.75	100	114.38	7.03	148.99	80	81.7	4.91	116.57	100
$(256, 4)$	90.89	6.78	96.88	90	170.5	8.73	227.01	40	68.8	5.36	99.93	100
$(256, 5)$	173.8	7.11	184.27	100	57.0	4.08	75.53	10	51.3	5.47	74.96	100
$(256, 6)$	388.1	8.39	400.16	100	–	–	–	0	65.56	5.74	85.24	90
$(256, 7)$	498.0	7.27	501.07	10	–	–	–	0	99.3	6.04	174.23	100
$(256, 8)$	–	–	–	0	–	–	–	0	93.7	6.46	144.96	100
$(256, 9)$	–	–	–	0	–	–	–	0	121.78	5.5	198.38	90
$(256, 10)$	–	–	–	0	–	–	–	0	137.67	4.54	257.37	90

References

[1] K.J. Arrow, L. Hurwicz, H. Uzawa, Studies in Linear and Non-linear Programming, Stanford University Press, 1958.
[2] A. Bagirov, N. Karmitsa, M.M. Mäkelä, Introduction to Nonsmooth Optimization: Theory, Practice and Software, Springer, 2014.
[3] J. Barreiro-Gomez, N. Quijano, C. Ocampo-Martinez, Constrained distributed optimization: a population dynamics approach, Automatica 69 (2016) 101–116, https://doi.org/10.1016/j.automatica.2016.02.004.
[4] D.P. Bertsekas, Nonlinear Programming, Athena Scientific, 1999.
[5] J. Bezanson, A. Edelman, S. Karpinski, V.B. Shah, Julia: a fresh approach to numerical computing, SIAM Rev. 59 (2017) 65–98, https://doi.org/10.1137/141000671.
[6] B. Biegel, J. Stoustrup, P. Andersen, Distributed MPC via dual decomposition, in: Distributed Model Predictive Control Made Easy, Springer, 2014, pp. 179–192.
[7] J.R. Birge, L. Qi, Z. Wei, Convergence analysis of some methods for minimizing a nonsmooth convex function, J. Optim. Theory Appl. 97 (1998) 357–383.
[8] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Found. Trends Mach. Learn. 3 (2011) 1–122, https://doi.org/10.1561/2200000016, https://www.nowpublishers.com/article/Details/MAL-016.
[9] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, New York, NY, USA, 2004.
[10] M.A. Bragin, P.B. Luh, B. Yan, X. Sun, A scalable solution methodology for mixed-integer linear programming problems arising in automation, IEEE Trans. Autom. Sci. Eng. 16 (2019) 531–541, https://doi.org/10.1109/TASE.2018.2835298.
[11] M.A. Bragin, P.B. Luh, J.H. Yan, N. Yu, G.A. Stern, Convergence of the surrogate Lagrangian relaxation method, J. Optim. Theory Appl. 164 (2015) 173–201, https://doi.org/10.1007/s10957-014-0561-3.
[12] A. Camisa, I. Notarnicola, G. Notarstefano, Distributed primal decomposition for large-scale MILPs, IEEE Trans. Autom. Control 67 (2021) 413–420.
[13] N. Chatzipanagiotis, D. Dentcheva, M.M. Zavlanos, An augmented Lagrangian method for distributed optimization, Math. Program. 152 (2015) 405–434.
[14] N. Chatzipanagiotis, M.M. Zavlanos, On the convergence of a distributed augmented Lagrangian method for nonconvex optimization, IEEE Trans. Autom. Control 62 (2017) 4405–4420.
[15] P.D. Christofides, R. Scattolini, D.M. de la Pena, J. Liu, Distributed model predictive control: a tutorial review and future research directions, Comput. Chem. Eng. 51 (2013) 21–41.
[16] A.R. Conn, K. Scheinberg, L.N. Vicente, Introduction to derivative-free optimization, J. Soc. Ind. Appl. Math. (2009), https://doi.org/10.1137/1.9780898718768.
[17] I. Dunning, J. Huchette, M. Lubin, JuMP: a modeling language for mathematical optimization, SIAM Rev. 59 (2017) 295–320, https://doi.org/10.1137/15M1020575.
[18] M. Eisen, A. Mokhtari, A. Ribeiro, A decentralized quasi-Newton method for dual formulations of consensus optimization, in: 2016 IEEE 55th Conference on Decision and Control (CDC), IEEE, 2016, pp. 1951–1958.
[19] M. Eisen, A. Mokhtari, A. Ribeiro, Decentralized quasi-Newton methods, IEEE Trans. Signal Process. 65 (2017) 2613–2628.
[20] H. Everett, Generalized Lagrange multiplier method for solving problems of optimum allocation of resources, Oper. Res. (1963) 399–417.
[21] P.A. Forero, A. Cano, G.B. Giannakis, Consensus-based distributed support vector machines, J. Mach. Learn. Res. 11 (2010).
[22] M. Fortin, R. Glowinski, Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems, Studies in Mathematics and Its Applications, vol. 15, North-Holland Pub. Co, Amsterdam and New York and New York, N.Y., 1983, http://www.sciencedirect.com/science/book/9780444866806.
[23] D. Gabay, B. Mercier, A dual algorithm for the solution of nonlinear variational problems via finite element approximation, Comput. Math. Appl. 2 (1976) 17–40, https://doi.org/10.1016/0898-1221(76)90003-1.
[24] C. Gambella, B. Ghaddar, J. Naoum-Sawaya, Optimization problems for machine learning: a survey, Eur. J. Oper. Res. 290 (2021) 807–828.
[25] W. Gao, R. Hernández, S. Engell, A study of explorative moves during modifier adaptation with quadratic approximation, Processes 4 (2016) 45.

[26] W. Gao, S. Wenzel, S. Engell, A reliable modifier-adaptation strategy for real-time optimization, Comput. Chem. Eng. 91 (2016) 318–328, https://doi.org/10.1016/j.compchemeng.2016.03.019.
[27] T. Goldstein, B. O'Donoghue, S. Setzer, R. Baraniuk, Fast alternating direction optimization methods, SIAM J. Imaging Sci. 7 (2014) 1588–1623, https://doi.org/10.1137/120896219.
[28] Z. Guo, G.J. Koehler, A.B. Whinston, A market-based optimization algorithm for distributed systems, Manag. Sci. 53 (2007) 1345–1358, https://doi.org/10.1287/mnsc.1060.0690.
[29] L.L.C. Gurobi Optimization, Gurobi optimizer reference manual, https://www.gurobi.com, 2021.
[30] S. Han, U. Topcu, G.J. Pappas, Differentially private distributed constrained optimization, IEEE Trans. Autom. Control 62 (2017) 50–64, https://doi.org/10.1109/TAC.2016.2541298.
[31] B.S. He, H. Yang, S.L. Wang, Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities, J. Optim. Theory Appl. 106 (2000) 337–356, https://doi.org/10.1023/A:1004603514434.
[32] H. He, D. Han, A distributed Douglas-Rachford splitting method for multi-block convex minimization problems, Adv. Comput. Math. 42 (2016) 27–53.
[33] B. Houska, J. Frasch, M. Diehl, An augmented Lagrangian based algorithm for distributed nonconvex optimization, SIAM J. Optim. 26 (2016) 1101–1127, https://doi.org/10.1137/140975991.
[34] I.Y. Joo, D.H. Choi, Distributed optimization framework for energy management of multiple smart homes with distributed energy resources, IEEE Access 5 (2017) 15551–15560.
[35] J. Konečnỳ, B. McMahan, D. Ramage, Federated optimization: distributed optimization beyond the datacenter, arXiv https://arxiv.org/abs/1511.03575, 2015.
[36] A. Kozma, C. Conte, M. Diehl, Benchmarking large-scale distributed convex quadratic programming algorithms, Optim. Methods Softw. 30 (2015) 191–214, https://doi.org/10.1080/10556788.2014.911298.
[37] Q. Le, A. Smola, S. Vishwanathan, Bundle methods for machine learning, Adv. Neural Inf. Process. Syst. 20 (2007).
[38] M. Li, Generalized Lagrange multiplier method and kkt conditions with an application to distributed optimization, IEEE Trans. Circuits Syst. II, Express Briefs 66 (2018) 252–256.
[39] T. Li, A.K. Sahu, A. Talwalkar, V. Smith, Federated learning: challenges, methods, and future directions, IEEE Signal Process. Mag. 37 (2020) 50–60.
[40] A.B. Liu, P.B. Luh, M.A. Bragin, B. Yan, Ordinal-optimization concept enabled decomposition and coordination of mixed-integer linear programming problems, IEEE Robot. Autom. Lett. 5 (2020) 5051–5058, https://doi.org/10.1109/LRA.2020.3005125.
[41] Y. Lu, M. Zhu, Privacy preserving distributed optimization using homomorphic encryption, Automatica 96 (2018) 314–325, https://doi.org/10.1016/j.automatica.2018.07.005.
[42] C. Ma, J. Konečnỳ, M. Jaggi, V. Smith, M.I. Jordan, P. Richtárik, M. Takáč, Distributed optimization with arbitrary local solvers, Optim. Methods Softw. 32 (2017) 813–848.
[43] M. Mäkelä, Survey of bundle methods for nonsmooth optimization, Optim. Methods Softw. 17 (2002) 1–29, https://doi.org/10.1080/10556780290027828.
[44] L.S. Maxeiner, S. Engell, An accelerated dual method based on analytical extrapolation for distributed quadratic optimization of large-scale production complexes, Comput. Chem. Eng. 135 (2020) 106728, https://doi.org/10.1016/j.compchemeng.2020.106728.
[45] S. Merugu, J. Ghosh, Privacy-preserving distributed clustering using generative models, in: Third IEEE International Conference on Data Mining, IEEE, 2003, pp. 211–218.
[46] I. Necoara, J. Suykens, Interior-point Lagrangian decomposition method for separable convex optimization, J. Optim. Theory Appl. 143 (2009) 567–588.
[47] A. Nedic, D.P. Bertsekas, Incremental subgradient methods for nondifferentiable optimization, SIAM J. Optim. 12 (2001) 109–138, https://doi.org/10.1137/S1052623499362111.
[48] A. Nedić, J. Liu, Distributed optimization for control, Annu. Rev. Control, Robot. Auton. Syst. 1 (2018) 77–103.
[49] J.E. Nesterov, Introductory Lectures on Convex Optimization: A Basic Course, Applied Optimization, vol. APOP 87, Kluwer Acad. Publ., Boston, Mass, 2004, http://www.loc.gov/catdir/enhancements/fy0822/2003061994-d.html.
[50] Y. Nesterov, A Method of Solving a Convex Programming Problem with Convergence Rate $\mathcal{O}(1/k^2)$, Sov. Math. Dokl., , 1983, pp. 372–376.
[51] J. Nocedal, S. Wright, Numerical Optimization, Springer Science & Business Media, 2006.
[52] I. Notarnicola, G. Notarstefano, Constraint-coupled distributed optimization: a relaxation and duality approach, IEEE Trans. Control Netw. Syst. 7 (2020) 483–492, https://doi.org/10.1109/TCNS.2019.2925267.

[53] D.P. Palomar, M. Chiang, A tutorial on decomposition methods for network utility maximization, IEEE J. Sel. Areas Commun. 24 (2006) 1439–1451, https://doi.org/10.1109/JSAC.2006.879350.
[54] N. Parikh, S. Boyd, Proximal algorithms, Foundations and trends® in Optimization 1 (2014) 127–239.
[55] P. Pflaum, M. Alamir, M.Y. Lamoudi, Scalability study for a hierarchical NMPC scheme for resource sharing problems, in: 2015 European Control Conference (ECC), IEEE, 2015, pp. 1468–1473.
[56] M. Ruskowski, A. Herget, J. Hermann, W. Motsch, P. Pahlevannejad, A. Sidorenko, S. Bergweiler, A. David, C. Plociennik, J. Popper, K. Sivalingam, A. Wagner, Production bots for production level 4, atp Magazin 62 (2020) 62–71, https://doi.org/10.17560/atp.v62i9.2505.
[57] A.M. Sampat, V.M. Zavala, Fairness measures for decision-making and conflict resolution, Optim. Eng. 20 (2019) 1249–1272, https://doi.org/10.1007/s11081-019-09452-3.
[58] S. Sun, Z. Cao, H. Zhu, J. Zhao, A survey of optimization methods from a machine learning perspective, IEEE Trans. Cybern. 50 (2019) 3668–3681.
[59] C.A. Uribe, S. Lee, A. Gasnikov, A. Nedić, A dual approach for optimal algorithms in distributed optimization over networks, in: 2020 Information Theory and Applications Workshop (ITA), IEEE, 2020, pp. 1–37.
[60] R. Vujanic, P.M. Esfahani, P.J. Goulart, S. Mariéthoz, M. Morari, A decomposition method for large scale MILPs, with performance guarantees and a power system application, Automatica 67 (2016) 144–156.
[61] A. Wächter, L.T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, Math. Program. 106 (2006) 25–57, https://doi.org/10.1007/s10107-004-0559-y.
[62] D. Walker, Walras's theories of tatonnement, J. Polit. Econ. (1987) 758–774.
[63] S.L. Wang, L.Z. Liao, Decomposition method with a variable parameter for a class of monotone variational inequality problems, J. Optim. Theory Appl. 109 (2001) 415–429, https://doi.org/10.1023/A:1017522623963.
[64] S. Wenzel, Distributed Optimization of Coupled Production Systems via Market-Like Coordination, Shaker Verlag, 2020.
[65] S. Wenzel, S. Engell, Coordination of coupled systems of systems with quadratic approximation, IFAC-PapersOnLine 52 (2019) 132–137, https://doi.org/10.1016/j.ifacol.2019.06.023.
[66] S. Wenzel, W. Gao, S. Engell, Handling disturbances in modifier adaptation with quadratic approximation, IFAC-PapersOnLine 48 (2015) 132–137, https://doi.org/10.1016/j.ifacol.2015.11.072.
[67] S. Wenzel, R. Paulen, S. Krämer, B. Beisheim, S. Engrell, Price Adjustment in Price-Based Coordination Using Quadratic Approximation, Computer Aided Chemical Engineering, vol. 38, Elsevier, 2016, pp. 193–198.
[68] S. Wenzel, R. Paulen, G. Stojanovski, S. Krämer, B. Beisheim, S. Engell, Optimal resource allocation in industrial complexes by distributed optimization and dynamic pricing, Automatisierungstechnik 64 (2016) 428–442.
[69] S. Wenzel, F. Riedl, S. Engell, An efficient hierarchical market-like coordination algorithm for coupled production systems based on quadratic approximation, Comput. Chem. Eng. 134 (2020) 106704, https://doi.org/10.1016/j.compchemeng.2019.106704.
[70] S. Wenzel, V. Yfantis, W. Gao, Comparison of regression data selection strategies for quadratic approximation in RTO, in: 27th European Symposium on Computer Aided Process Engineering, in: Computer Aided Chemical Engineering, vol. 40, Elsevier, 2017, pp. 1711–1716.
[71] B. Yang, M. Johansson, Distributed optimization and games: a tutorial overview, in: M. Morari, M. Thoma, A. Bemporad, M. Heemels, M. Johansson (Eds.), Networked Control Systems, in: Lecture Notes in Control and Information Sciences, vol. 406, Springer London, London, 2010, pp. 109–148.
[72] T. Yang, X. Yi, J. Wu, Y. Yuan, D. Wu, Z. Meng, Y. Hong, H. Wang, Z. Lin, K.H. Johansson, A survey of distributed optimization, Annu. Rev. Control 47 (2019) 278–305.
[73] V. Yfantis, 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), IEEE, 2022, pp. 31–36.
[74] M. Zargham, A. Ribeiro, A. Ozdaglar, A. Jadbabaie, Accelerated dual descent for network flow optimization, IEEE Trans. Autom. Control 59 (2014) 905–920, https://doi.org/10.1109/TAC.2013.2293221.
[75] Y. Zhang, N. Gatsis, G.B. Giannakis, Disaggregated bundle methods for distributed market clearing in power networks, in: 2013 IEEE Global Conference on Signal and Information Processing, IEEE, 2013, pp. 835–838.