Results in Control and Optimization 17 (2024) 100495

Contents lists available at ScienceDirect

Results in Control and Optimization

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

Numerical benchmarking of dual decomposition-based optimization algorithms for distributed model predictive control

Vassilios Yfantis [a,*], Achim Wagner [b], Martin Ruskowski [a,b]

[a] Chair of Machine Tools and Control Systems, Department of Mechanical and Process Engineering, TU Kaiserslautern, Gottlieb-Daimler-Straße 42, 67663 Kaiserslautern, Germany

[b] Innovative Factory Systems, German Research Center for Artificial Intelligence, Trippstadter Str. 122, 67663 Kaiserslautern, Germany

ARTICLE INFO

Keywords:
Distributed model predictive control
Dual decomposition
Subgradient method
ADMM
Bundle method
Quasi-Newton

ABSTRACT

This paper presents a benchmark study of dual decomposition-based distributed optimization algorithms applied to constraint-coupled model predictive control problems. These problems can be interpreted as multiple subsystems which are coupled through constraints on the availability of shared limited resources. In a dual decomposition-based framework the production and consumption of these resources can be coordinated by iteratively computing their prices and sharing them with the involved subsystems. Following a brief introduction to model predictive control different architectures and communication topologies for a distributed setting are presented. After decomposing the system-wide control problem into multiple subproblems by introducing dual variables, several distributed optimization algorithms, including the recently proposed quasi-Newton dual ascent algorithm, are discussed. Furthermore, an epigraph formulation of the bundle cuts as well as a line search strategy are proposed for the quasi-Newton dual ascent algorithm, which increase its numerical robustness and speed up its convergence compared to the previously used trust region. Finally, the quasi-Newton dual ascent algorithm is compared to the subgradient method, the bundle trust method and the alternating direction method of multipliers for a large number of benchmark problems. The used benchmark problems are publicly available on GitHub.

1. Introduction

The increase in complexity of industrial systems has necessitated the use of advanced control methods. Decentral control architectures, consisting of a large number of decoupled PID control loops, have been used in practice for many years [1]. These methods often assume weak or non-existent coupling between the control loops of the system-wide control problem. However, the strive for a more efficient operation of large-scale systems has led to a tighter integration of their individual components, reducing the robustness of decentralized control schemes. In addition, modern systems, and thus their associated control systems, are tasked with satisfying multiple operational and safety-related constraints while simultaneously optimizing several performance metrics. Model predictive control (MPC) is an optimization-based control method whereby a model of the plant is used to predict its future behavior depending on its current state and the computed control inputs. By stating the control problem as an optimization problem the system's performance metrics can be minimized or maximized directly while the involved constraints can be explicitly accounted for. Nowadays problems with hundreds or even thousands of variables can be solved efficiently, in large part due to the substantial performance increase of the used optimization solvers. However, some situations still necessitate the use of a more

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

https://doi.org/10.1016/j.rico.2024.100495
Received 13 December 2023; Accepted 9 November 2024
Available online 19 November 2024

decentralized control approach. One example are large-scale systems that consist of multiple autonomous subsystems, each with their own objectives and constraints. Even though these subsystems can take autonomous decisions, they might still be coupled, e.g., by streams of energy or materials. In this case system-wide constraints must still be satisfied, even when the individual control problems are solved in a decentralized manner. A reason for such a control structure can be confidentiality between the subsystems, e.g., if they belong to different legal entities and are not allowed or willing to disclose information on their dynamics and local constraints. Distributed optimization can address this issue by coordinating the controllers of the different subsystems, leading to distributed model predictive control (DMPC) problems.

A class of distributed optimization algorithms suited for the coordination of resource-coupled DMPC problems are dual decomposition based-methods. These methods are also often referred to as price-based coordination, whereby the introduced dual variables are interpreted as prices for shared limited resources and can be used to steer the subsystems towards a feasible solution with respect to the system-wide constraints. The contribution of this paper is twofold: First, the recently proposed quasi-Newton dual ascent algorithm [2,3] is extended by reformulating its used cutting planes into linear constraints via an epigraph formulation. Additionally, a constrained line search update step is introduced, which speeds up the convergence in the vicinity of the optimum. Second, a large number of DMPC benchmark problem instances were generated and made publicly available on GitHub. Even though several papers have conducted benchmark studies on DMPC algorithms, many of them either only consider isolated problem instance [4–6] or do not make the benchmark instances publicly available [7]. The remainder of this paper is structured as follows: Section 2 introduces the notation used throughout this paper as well as a general formulation of the considered class of MPC problems. Section 3 discusses DMPC, including a review of different control architectures. Section 4 deals with dual decomposition-based DMPC. After illustrating how the central MPC problem can be decomposed via the use of dual variables, several coordination algorithms are discussed, namely, the subgradient method, the bundle trust method, the alternating direction method of multipliers, and the quasi-Newton dual ascent. Section 5 first describes the generation of the DMPC benchmark problems, followed by a discussion of the benchmark results. The paper is concluded in Section 6.

2. Preliminaries

In this paper we denote vectors by bold lowercase letters ($\mathbf{x}$) and matrices by bold uppercase letters ($\mathbf{A}$). In MPC time is usually discretized into equal time intervals with a sampling time T_s, i.e., into time points $t_k = t_0 + k \cdot T_s$, where t_0 is the current time. The notation $\mathbf{x}^k$ and $\mathbf{u}^k$ refers to the states and inputs of the plant at time t_k respectively. The value of a variable $\mathbf{z}$ in iteration t of a dual decomposition-based algorithm is denoted by $\mathbf{z}^{(t)}$. The Euclidean norm is denoted by $\|\cdot\|_2$.

MPC algorithms usually employ a discrete-time model to predict the evolution of the plant's states $\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}}$ depending on the current states and control inputs $\mathbf{u} \in \mathbb{R}^{n_\mathbf{u}}$. A typical MPC optimization problem is given by Eq. (1),

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

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

$$\mathbf{x}^0 = \bar{\mathbf{x}}(t_0), \tag{1c}$$

$$\mathbf{x}^k \in \mathcal{X} \subset \mathbb{R}^{n_\mathbf{x}}, \; k = 0, \dots, N_p, \tag{1d}$$

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

Constraint (1b) represents a discrete-time model used to predict the state of the plant $\mathbf{x}^{k+1}$ at the next sampling time from the current states $\mathbf{x}^k$ and the used control input $\mathbf{u}^k$ with $\mathbf{A} \in \mathbb{R}^{n_\mathbf{x} \times n_\mathbf{x}}$ and $\mathbf{B} \in \mathbb{R}^{n_\mathbf{x} \times n_\mathbf{u}}$. The future states and inputs are computed over a prediction horizon N_p. Note that the control inputs must not necessarily be computed over the entire prediction horizon, especially if solving the MPC optimization problem is computationally expensive. Instead, the control inputs can be computed over a control horizon $N_c < N_p$ and then be kept constant until the end of the prediction horizon, i.e., $\mathbf{u}^{N_c} = \mathbf{u}^{N_c+1} = \cdots = \mathbf{u}^{N_p-1}$. In this work $N_p = N_c$ is assumed. Constraint (1c) represents the initial conditions of the plant, whereby $\bar{\mathbf{x}}(t_0)$ denotes the measured states at time t_0. Depending on the plant the actual states might not be measurable. In these cases, state estimation techniques might be used to estimate the plant's states from the measured output. However, state estimation is outside the scope of this paper. Necoara et al. provide an overview of state estimation in a distributed setting [8]. A main advantage of MPC compared to other control approaches is that constraints on the states and inputs can be explicitly considered within the optimization problem [9]. These constraints are summarized by (1d) and (1e) respectively.

At each time step an input trajectory is computed over a prediction horizon N_p by solving the open-loop optimal control problem (1). A common objective in MPC is to track a given reference trajectory over the prediction horizon while also minimizing the control inputs, i.e.,

$$J(\mathbf{x}^k, \mathbf{u}^k) = (\mathbf{x}^k - \mathbf{x}^{\text{ref},k})^T \mathbf{H}_\mathbf{x} (\mathbf{x}^k - \mathbf{x}^{\text{ref},k}) + \mathbf{u}^{k,T} \mathbf{H}_\mathbf{u} \mathbf{u}^k, \tag{2}$$

where $\mathbf{H}_\mathbf{x} \in \mathbb{R}^{n_\mathbf{x} \times n_\mathbf{x}}$ and $\mathbf{H}_\mathbf{u} \in \mathbb{R}^{n_\mathbf{u} \times n_\mathbf{u}}$ are symmetric positive (semi-) definite weighing matrices. Eq. (2) is often referred to as stage costs, while $J^f(\mathbf{x}^{N_p})$ is referred to as terminal cost. A common terminal cost is the deviation of the states from their reference at the end of the prediction horizon, i.e.,

$$J^f(\mathbf{x}^{N_p}) = (\mathbf{x}^{N_p} - \mathbf{x}^{\text{ref},N_p})^T \mathbf{H}_\mathbf{x}^f (\mathbf{x}^{N_p} - \mathbf{x}^{\text{ref},N_p}). \tag{3}$$

The design of the terminal cost plays an important role regarding the control performance and stability [9]. These aspects are also outside the scope of this paper. Another main appeal of MPC is that control objectives beyond classical reference tracking can be incorporated into the optimization problem [10]. For instance, the economic performance of the process can be directly optimized, giving rise to economic MPC (EMPC). An extensive review of EMPC is provided by Ellis et al. [11].

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

3. Distributed model predictive control

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

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

$$\min_{\mathbf{x}^{0:N_p}, \mathbf{u}^{0:N_p-1}} \sum_{i\in\mathcal{I}} \left[J_i^f(\mathbf{x}_i^{N_p}) + \sum_{k=0}^{N_p-1} J_i(\mathbf{x}_i^k, \mathbf{u}_i^k) \right], \tag{4a}$$

$$\text{s. t. } \mathbf{x}_i^{k+1} = \sum_{j\in\mathcal{I}} \mathbf{A}_{ij}\mathbf{x}_j^k + \mathbf{B}_{ij}\mathbf{u}_j^k, \ \forall i \in \mathcal{I}, \ k = 0, \dots, N_p - 1, \tag{4b}$$

$$\mathbf{x}_i^0 = \tilde{\mathbf{x}}(t_0), \ \forall i \in \mathcal{I}, \tag{4c}$$

$$\mathbf{x}_i^k \in \mathcal{X}_i \subset \mathbb{R}^{n_{\mathbf{x}_i}}, \ \forall i \in \mathcal{I}, k = 0, \dots, N_p, \tag{4d}$$

$$\mathbf{u}_i^k \in \mathcal{U}_i \subset \mathbb{R}^{n_{\mathbf{u}_i}}, \ \forall i \in \mathcal{I}, k = 0, \dots, N_p - 1, \tag{4e}$$

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

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

$$\min_{\mathbf{x}^{0:N_p}, \mathbf{u}^{0:N_p-1}} \sum_{i\in\mathcal{I}} \left[J_i^f(\mathbf{x}_i^{N_p}) + \sum_{k=0}^{N_p-1} J_i(\mathbf{x}_i^k, \mathbf{u}_i^k) \right], \tag{5a}$$

$$\text{s. t. } \mathbf{x}_i^{k+1} = \mathbf{A}_i\mathbf{x}_i^k + \mathbf{B}_i\mathbf{u}_i^k, \ \forall i \in \mathcal{I}, k = 0, \dots, N_p - 1, \tag{5b}$$

$$\mathbf{x}_i^0 = \tilde{\mathbf{x}}(t_0), \ \forall i \in \mathcal{I}, \tag{5c}$$

$$\mathbf{x}_i^k \in \mathcal{X}_i \subset \mathbb{R}^{n_{\mathbf{x}_i}}, \ \forall i \in \mathcal{I}, k = 0, \dots, N_p, \tag{5d}$$

$$\mathbf{u}_i^k \in \mathcal{U}_i \subset \mathbb{R}^{n_{\mathbf{u}_i}}, \ \forall i \in \mathcal{I}, \ k = 0, \dots, N_p - 1, \tag{5e}$$

$$\sum_{i\in\mathcal{I}} \mathbf{R}_i\mathbf{u}_i^k \le \mathbf{r}_{\max}^k, \ k = 0, \dots, N_p - 1. \tag{5f}$$

The dynamics of each subsystem (5b) are decoupled. The coupling of the subsystem is represented by the system-wide constraints (5f), which can be interpreted as constraints on the availability of shared limited resources. The matrices $\mathbf{R}_i \in \mathbb{R}^{n_\mathbf{r} \times n_{\mathbf{u}_i}}$ map the subsystems inputs to the corresponding resource consumption or production, while $\mathbf{r}_{\max}^k \in \mathbb{R}^{n_\mathbf{r}}$ denotes the maximum availability of resources. Note that in the resource constraints, a positive sign indicates consumption while a negative sign indicates the production of a resource.

This paper focuses on the second type of DMPC problems (5), which can be solved within a dual decomposition-based distributed optimization framework. Note that DMPC problems of the form (4) can also be solved by using dual decomposition, which will be briefly discussed in Section 4.3.

3.1. DMPC architectures

Apart from the type of coupling between the subsystems, the employed DMPC method strongly depends on the allowed communication. Several DMPC architectures are reviewed by, e.g., Scattolini [12] and Christofides et al. [13], some of which are depicted in Fig. 1. For the sake of simplicity, the influence of the output of a DMPC controller on other subsystems is omitted at this point, i.e., $\mathbf{B}_{ij} = \mathbf{0}$ for $i \neq j$ (cf. Eq. (4b)). Fig. 1(a) depicts a setting in which the coupling of the subsystems is not explicitly considered by the DMPC controllers, i.e., there is no communication between them. This is often referred to as decentralized MPC. In this setting, the coupling between the subsystems can be regarded as an external disturbance that can be compensated by the individual controllers. If the coupling between subsystems is strong, decentralized control might exhibit poor performance. Furthermore, decentralized DMPC is not suitable for constraint-coupled problems as the satisfaction of system-wide constraints

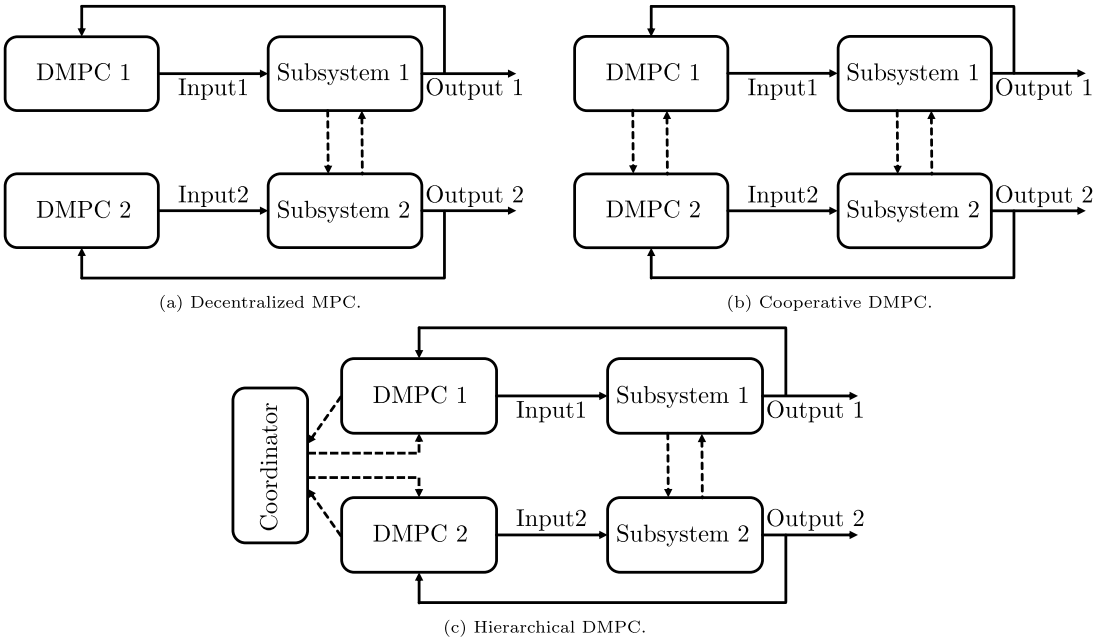


Fig. 1. Different distributed model predictive control structures for two subsystems with two controllers.
Source: Adapted from [12].

cannot be guaranteed without some amount of information exchange. Christophides et al. provide an overview of decentralized MPC [13].

If coupling between subsystems must be accounted for explicitly by the DMPC controllers some information must be exchanged. Fig. 1(b) depicts a setting in which direct communication between the DMPC controllers is allowed. In this setting, the controllers can exchange information like predicted state and input trajectories, objective values, constraints, etc. DMPC algorithms relying on this architecture can be classified according to the number of communication rounds per MPC iteration, i.e., how often information is exchanged before a control action is applied to the subsystems. Camponogara et al. provide a general overview of both types of communication [14]. A main criterion for which information exchange frequency should be employed is the computational cost of the solution of a single DMPC problem. While DMPC with multiple communication rounds, often also referred to as agent negotiation [15], usually results in better performance and under certain conditions enables the convergence to the optimal solution of the system-wide MPC problem, it is usually computationally more expensive than single communication DMPC since the subsystems have to solve multiple optimization problems in each iteration. If short sampling times must be considered, performing multiple communication rounds is usually not feasible. This is the case, e.g., in the control of robotic manipulators with collision constraints, where only the predicted state trajectory is communicated to the neighboring DMPC controllers [16].

Finally, Fig. 1(c) depicts a setting where no direct communication between the DMPC controllers is allowed. This leads to a hierarchical control structure where the individual controllers share information with a coordinator. The coordinator aggregates the responses and communicates price signals to the DMPC controllers. This setting is considered in dual decomposition-based DMPC, where the coordinator communicates the dual variables to the subsystems. Hierarchical DMPC is especially suitable for constraint-coupled problems where direct information exchange between the controllers is not intended, e.g., due to confidentiality reasons. Furthermore, a hierarchical DMPC approach always requires multiple communication rounds between the subsystems and the coordinator. Thus, this approach is only suitable for systems with slow dynamics and large sampling times, where the underlying MPC problems can be solved multiple times within one sampling period until a feasible solution is found, e.g., for chemical plants or power systems.

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

3.2. DMPC literature review

Distributed control for large-scale systems has been an active field of research since the 1970s [17]. Extensive reviews on DMPC are provided by, e.g., Scattolini [12], Necoara et al. [8], Chrisofides et al. [13], Negenborn and Maestre [18] and Müller and Allgöwer [19].

Stewart et al. present a cooperative DMPC algorithm that can be interpreted as a suboptimal system-wide MPC, in turn proving stability [20]. Maestre et al. propose a cooperative DMPC algorithm for two agents with two communication rounds per sampling period [21]. They also propose a cooperative DMPC scheme with multiple agents whereby in each communication round a random

set of agents proposes a control action, which is accepted by their neighbors if it improves the overall control objective [15]. Zheng et al. present a DMPC algorithm for a network of interacting systems with direct communication between the controllers [22].

Dual decomposition-based DMPC has been applied to systems coupled in their dynamics and constraints. Giselsson and Rantzer employ dual decomposition for systems coupled in their dynamics and present a stopping criterion that guarantees bounded suboptimality and asymptotic stability [23]. Gieselsson et al. also present an accelerated gradient method for dual decomposition-based DMPC where the objective function includes $\mathcal{L}_1$-norm penalty terms [24]. In [25] Giselsson and Rantzer present a stopping criterion based on adaptive constraint tightening that allows for an early termination of dual decomposition-based DMPC problems for dynamically coupled systems. Köhler et al. examine recursive feasibility for premature termination of a dual decomposition-based distributed optimization algorithm [26]. Doan et al. employ dual decomposition and use primal averaging and constraint tightening to ensure convergence of DMPC problems with both coupled dynamics and constraints [27].

Biegel et al. show how MPC can be used for congestion management in an energy grid comprised of intelligent consumers with energy storage capabilities [28]. This problem is extended into a DMPC setting in [29]. Biegel et al. also show how dual decomposition-based DMPC can be generally applied to constraint-coupled systems [30]. Pflaum et al. compare primal and dual decomposition-based DMPC for the distributed control in smart districts [31]. In [32] they examine the scalability of dual decomposition-based DMPC using a bundle method based on the number of subsystems, the number of constraints, and the maximum size of the bundle.

Razzanelli et al. use DMPC for the management of the energy flow within a network of microgrids using a subgradient method [33]. Eser et al. use the alternating direction method of multipliers (ADMM) to distributedly control building energy systems [34]. Maxeiner and Engell use ADMM for DMPC of semi-batch reactors coupled in their inputs through shared limited resources [35]. This work is extended in [5], where the subgradient method, ADMM, and the augmented Lagrangian-based alternating direction inexact Newton method (ALADIN) are extensively compared for these types of problems. Houska and Shi provide a tutorial on how ALADIN can be employed for DMPC [36]. Yfantis et al. apply a quadratic approximation-based algorithm to DMPC problems with shared limited resources, comparing its performance to the subgradient method [37].

Conte et al. study computational aspects of dual decomposition-based DMPC using an accelerated gradient method and ADMM [7]. More specifically they examine the influence of the coupling strength, stability, the initial state, and the network size and topology on the computational performance of the DMPC. Stomberg et al. compare the performance of six algorithms for DMPC of dynamically coupled systems, namely the subgradient method, an accelerated subgradient method, ADMM, a distributed active set algorithm, an essentially decentralized interior point method, and Jacobi iterations [6].

Other proximal algorithms, other than ADMM, can also be used for DMPC. Halvgaard et al. use the Douglas-Rachford splitting method for the coordination of smart energy systems [38]. An extensive collection of DMPC algorithms and applications can be found in the textbook by Maestre and Negenborn [39].

4. Dual decomposition-based distributed model predictive control

This section deals with dual decomposition-based DMPC. After decomposing a system-wide constraint-coupled MPC problem by introducing dual variables, the corresponding dual problem is formulated, which can be solved in a distributed manner. Several algorithms for the solution of the dual problem are presented, including the quasi-Newton dual ascent algorithm. Finally, the use of dual decomposition is demonstrated for dynamically coupled DMPC problems for the sake of completeness.

4.1. Dual decomposition of constraint-coupled DMPC problems

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

$$\mathcal{L}(\mathbf{x}^{0:N_p}, \mathbf{u}^{0:N_p-1}, \lambda^{0:N_p-1}) = \sum_{i\in\mathcal{I}} \underbrace{\left[J_i^f(\mathbf{x}_i^{N_p}) + \sum_{k=0}^{N_p-1} [J_i(\mathbf{x}_i^k, \mathbf{u}_i^k) + \lambda^{k,T} \mathbf{R}_i \mathbf{u}_i^k] \right]}_{=: \mathcal{L}_i(\mathbf{x}_i^{0:N_p}, \mathbf{u}_i^{0:N_p-1}, \lambda^{0:N_p-1})} - \sum_{k=0}^{N_p-1} \lambda^{k,T} \mathbf{r}_{\max}^k \tag{6}$$

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

$$\mathcal{L}(\mathbf{x}^{0:N_p}, \mathbf{u}^{0:N_p-1}, \lambda^{0:N_p-1}) = \sum_{i\in\mathcal{I}} \mathcal{L}_i(\mathbf{x}_i^{0:N_p}, \mathbf{u}_i^{0:N_p-1}, \lambda^{0:N_p-1}) - \sum_{k=0}^{N_p-1} \lambda^{k,T} \mathbf{r}_{\max}^k \tag{7}$$

Thus the dual function

$$d(\lambda^{0:N_p-1}) := \min_{\mathbf{x}^{0:N_p}, \mathbf{u}^{0:N_p-1}} \mathcal{L}(\mathbf{x}^{0:N_p}, \mathbf{u}^{0:N_p-1}, \lambda^{0:N_p-1}), \tag{8}$$
$$\text{s t. (5b)–(5e)}$$

can be evaluated by solving the individual DMPC problems

$$\min_{\mathbf{x}_i^{0:N_p}, \mathbf{u}_i^{0:N_p-1}} \mathcal{L}_i(\mathbf{x}_i^{0:N_p}, \mathbf{u}_i^{0:N_p-1}, \lambda^{0:N_p-1}), \tag{9a}$$

$$\text{s t. } \mathbf{x}_i^{k+1} = \mathbf{A}_i \mathbf{x}_i^k + \mathbf{B}_i \mathbf{u}_i^k,$$
$$k = 0, \dots, N_p - 1, \tag{9b}$$
$$\mathbf{x}_i^0 = \tilde{\mathbf{x}}(t_0), \tag{9c}$$
$$\mathbf{x}_i^k \in \mathcal{X}_i \subset \mathbb{R}^{n_{\mathbf{x}_i}},\ k = 0, \dots, N_p, \tag{9d}$$
$$\mathbf{u}_i^k \in \mathcal{U}_i \subset \mathbb{R}^{n_{\mathbf{u}_i}},\ k = 0, \dots, N_p - 1 \tag{9e}$$

for each subsystem i in a distributed manner.

Dual decomposition-based optimization relies on the solution of the dual optimization problem

$$\max_{\lambda^{0:N_p-1}} d(\lambda^{0:N_p-1}), \tag{10a}$$
$$\text{s t. } \lambda^k \geq \mathbf{0},\ k = 0, \dots, N_p - 1, \tag{10b}$$

which is amendable to distributed computations due to the separability of the dual function. Constraint (10b) stems from the Karush–Kuhn–Tucker conditions and the coupling constraints being inequalities [40].

The dual function and the corresponding dual optimization problem exhibit some important properties. First, the optimal objective value of the dual problem (10) always provides a lower bound on the objective of problem (5), which is called the primal problem in this context. This property is referred to as weak duality. If the primal problem is convex, and if some additional regularity conditions hold, the optimal objective values of the primal and the dual problem are equal, which is referred to as strong duality [40]. Furthermore, an optimal primal solution is obtained when the subproblems (9) are solved with the optimal value of the dual variables. Second, the dual function is always concave, regardless of whether or not the primal problem is convex or not. However, it is not always smooth, i.e., a gradient of the dual function does not exist for every value of the dual variables [3]. Thus, the dual problem (10) is a nonsmooth convex optimization problem. The following section reviews algorithms that are used to solve the nonsmooth dual problem.

4.2. Dual decomposition-based algorithms

This section presents the dual decomposition-based distributed optimization algorithms used throughout this paper. After a brief description of the subgradient method, the bundle trust method and the alternating direction method of multipliers, the recently proposed quasi-Newton dual ascent algorithm [2,3] is presented in more detail. Compared to [3] the algorithm is extended by the reformulation of the update problem via an epigraph formulation and by a line search update step in the vicinity of the optimum.

4.2.1. Subgradient method

The simplest algorithm for solving the nonsmooth dual problem is the subgradient method [41]. The subgradient is a generalization of the gradient for nonsmooth functions. For the sake of brevity we denote the column vector of the dual variables over the prediction horizon as

$$\lambda_{N_p} := [\lambda^{0,T}, \dots, \lambda^{N_p-1,T}]^T \in \mathbb{R}^{n_{n_{\mathbf{r}} \cdot (N_p-1)}} \tag{11}$$

A subgradient of the dual function (8) in iteration t can be computed by evaluating the constraints on the shared limited resources (5f), i.e.,

$$\mathbf{g}(\lambda^{0:N_p-1,(t)}) := \begin{pmatrix} \sum_{i \in \mathcal{I}} \mathbf{R}_i \mathbf{u}_i^{0,(t+1)} - \mathbf{r}_{\max}^0 \\ \vdots \\ \sum_{i \in \mathcal{I}} \mathbf{R}_i \mathbf{u}_i^{N_p-1,(t+1)} - \mathbf{r}_{\max}^{N_p-1} \end{pmatrix}, \tag{12}$$

where $\mathbf{u}_i^{k,(t+1)}$ are the control inputs computed by the subsystems by solving problem (9) in iteration t of the distributed optimization algorithm. In the subgradient method the dual variables can then be updated in each iteration in the direction of the subgradient,

$$\lambda_{N_p}^{(t+1)} = [\lambda_{N_p}^{(t)} + \alpha^{(t)} \mathbf{g}(\lambda_{N_p}^{(t)})]^+, \tag{13}$$

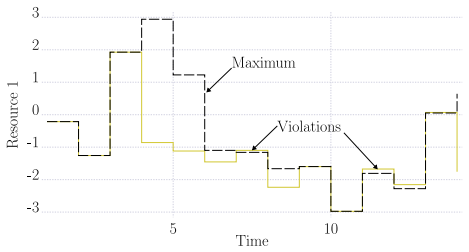
where $[\cdot]^+$ denotes the projection onto the positive orthant. The step size parameter $\alpha^{(t)}$ is usually adapted over the course of the iterations. The algorithm is terminated when the norm of the primal residual

$$[\mathbf{w}_p^{(t)}]_l := \max \left\{ \left[\mathbf{g}(\lambda_{N_p}^{(t)}) \right]_l, 0 \right\},\ l = 1, \dots, n_{\mathbf{b}} \tag{14}$$

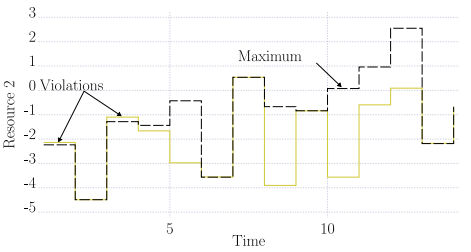
and the dual residual

$$\mathbf{w}_d^{(t)} := \lambda_{N_p}^{(t+1)} - \lambda_{N_p}^{(t)}, \tag{15}$$

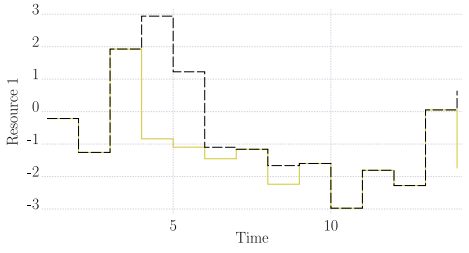
lie below a predefined threshold, i.e., when the primal problem is feasible and the dual variables have converged to a stationary value.



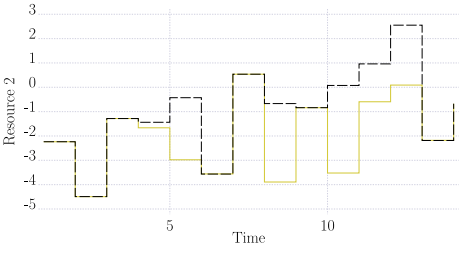
(a) Utilization of resource 1 with update strategy (22b).



(b) Utilization of resource 2 with update strategy (22b).



(c) Utilization of resource 1 with update strategy (23).



(d) Utilization of resource 2 with update strategy (23).

Fig. 2. Comparison of the resource utilization upon convergence of the ADMM algorithm for DMPC benchmark problem with different update strategies of the auxiliary variables. The plots show the aggregated resource utilization for a DMPC problem with $N_s = 20, n_{\mathbf{x}} = 3, n_{\mathbf{u}} = 2, n_{\mathbf{u}} = 2, N_p = 15$. The constraints on the shared resources are violated when using the update strategy (22b).

4.2.2. Bundle trust method

The subgradient method only uses the current subgradient to update the dual variables. Bundle methods are a class of generally more efficient nonsmooth optimization algorithms, whereby subgradients from multiple previous iterations are used to update the dual variables [42]. To this end the data

$$\mathcal{B}^{(t)} = \{(\lambda_{N_p}^{(l)}, \mathbf{g}(\lambda_{N_p}^{(l)}), d(\lambda_{N_p}^{(l)})) \in \mathbb{R}^{n_\lambda} \times \mathbb{R}^{n_\lambda} \times \mathbb{R} |\ l = t-\tau+1, \dots, t\} \tag{16}$$

is stored in each iteration, where $n_\lambda = n_{\mathbf{r}} \cdot (N_p - 1)$ denotes the number of dual variables. $\mathcal{B}^{(t)}$ is referred to as a bundle and it contains the dual variables, subgradients, and values of the dual function from previous iterations. Since storing all information from all previous iterations might cause memory issues, only data from the previous τ iterations is used.

The collected bundle information can then be used to construct a piece-wise linear over-approximation of the dual function, a so-called cutting plane model, which is then optimized. Since the cutting plane model is still nonsmooth, an epigraph formulation is used to compute a search direction for the dual variables, subject to a trust region constraint, leading to a bundle trust method (BTM)

$$\max_{v \in \mathbb{R},\, \mathbf{s} \in \mathbb{R}^{n_\lambda}} \ v, \tag{17a}$$

$$\text{s. t. } \|\mathbf{s}\|_2^2 \le \alpha^{(t)}, \tag{17b}$$

$$\mathbf{g}^T(\lambda_{N_p}^{(l)})\mathbf{s} - \beta^{(l,t)} \ge v, \ \forall l \in \{t-\tau+1, \dots, t\}, \tag{17c}$$

$$\lambda_{N_p}^{(t)} + \mathbf{s} \ge \mathbf{0}. \tag{17d}$$

with the linearization error

$$\beta^{(l,t)} = d(\lambda_{N_p}^{(t)}) - d(\lambda_{N_p}^{(l)}) - \mathbf{g}^T(\lambda_{N_p}^{(l)})(\lambda_{N_p}^{(t)} - \lambda_{N_p}^{(l)}), \ \forall l \in \{t-\tau+1, \dots, t\}. \tag{18}$$

After computing a direction the dual variables are updated according to

$$\lambda_{N_p}^{(t+1)} = \lambda_{N_p}^{(t)} + \mathbf{s}^{(t)}. \tag{19}$$

4.2.3. Alternating direction method of multipliers

The alternating direction method of multipliers (ADMM) employs regularization terms that convexify the response surface of the dual function [43]. The setting of multiple subsystems coupled via shared limited resources can be handled by the optimal exchange version of ADMM [44]. In this an augmented Lagrange function is defined by introducing auxiliary variables $\mathbf{z}_i^k$ for each subsystem,

$$\mathcal{L}_{\rho,i}(\mathbf{x}_i^{0:N_p}, \mathbf{u}_i^{0:N_p-1}, \lambda^{0:N_p-1}, \mathbf{z}_i^{0:N_p-1}) =$$
$$J_i^f(\mathbf{x}_i^{N_p}) + \sum_{k=0}^{N_p-1} \left[J_i(\mathbf{x}_i^k, \mathbf{u}_i^k) + \lambda^{k,T}(\mathbf{R}_i\mathbf{u}_i^k - \mathbf{z}_i^k)\right] + \frac{\rho}{2}\sum_{k=0}^{N_p-1} \|\mathbf{R}_i\mathbf{u}_i^k - \mathbf{z}_i^k\|_2^2. \tag{20}$$

Convergence of the ADMM algorithm can be proven for constant values of the regularization parameter ρ. However, adapting the parameter over the course of the iterations usually works better in practice. In this paper, the adaptation strategy reported in [45,46] is employed:

$$\rho^{(t+1)} = \begin{cases} \tau^{\text{incr}}\rho^{(t)}, & \text{if } \|\mathbf{w}_p^{(t)}\|_2 > \mu\|\mathbf{w}_d^{(t)}\|_2, \\ \rho^{(t)}/\tau^{\text{decr}}, & \text{if } \|\mathbf{w}_d^{(t)}\|_2 > \mu\|\mathbf{w}_p^{(t)}\|_2, \\ \rho^{(t)}, & \text{otherwise.} \end{cases} \tag{21}$$

The parameters $\mu, \tau^{\text{incr}}, \tau^{\text{decr}} > 1$ are tuning parameters. The dual and auxiliary variables can then be updated according to

$$\mathbf{z}_i^{k,(t+1)} = \mathbf{R}_i\mathbf{u}_i^{(k,t+1)} - \frac{1}{N_s}\sum_{i\in\mathcal{I}}\left(\mathbf{R}_i\mathbf{u}_i^{(t+1)} - \mathbf{r}_{\max}^k\right),\ k = 0,\dots,N_p-1, \tag{22a}$$

$$\lambda^{k,(t+1)} = \left[\lambda^{k,(t)} + \rho^{(t)}\frac{1}{N_s}\sum_{i\in\mathcal{I}}\left(\mathbf{R}_i\mathbf{u}_i^{(t+1)} - \mathbf{r}_{\max}^k\right)\right]^+,\ k = 0,\dots,N_p-1. \tag{22b}$$

ADMM tends to push the subsystems towards primal feasibility due to the added regularization term. For inequality-constrained problems, this can lead to a situation where the system-wide constraints are satisfied while the regularization term does not vanish. However, the regularization term should vanish if the optimal dual variables are found, i.e., the dual variables found using ADMM should result in a primal feasible solution for the unaugmented Lagrange function. Maxeiner [47] proposed to replace the update of the auxiliary variables (22b) by an optimization problem,

$$\mathbf{z}^{(t+1)} = \underset{\mathbf{z}}{\text{argmin}} \sum_{k=0}^{N_p-1}\left\|\sum_{i\in\mathcal{I}}\mathbf{R}_i\mathbf{u}_i^{k,(t)} - \mathbf{z}_i^k\right\|_2^2, \tag{23a}$$

$$\text{s. t. } \sum_{i\in\mathcal{I}}\mathbf{z}_i^k \le \mathbf{r}_{\max}^k,\ k = 0,\dots,N_p-1. \tag{23b}$$

Fig. 2 illustrates the effect of the update of the auxiliary variables. When using the update strategy (22b) ADMM converges to a value of the dual variables where optimizing the augmented Lagrange function results in a feasible resource utilization. However, when the same dual variables are used to optimize the unaugmented Lagrange function, i.e., without the regularization term, the constraints on the shared limited resources are violated. Thus, ADMM does not converge to the optimal dual variables. This case is shown for two resources in Figs. 2(a) and 2(b). When the auxiliary variables are updated according to (23) ADMM converges to a dual optimal solution, i.e., a feasible solution for both the augmented and unaugmented Lagrange function. This is depicted in Figs. 2(c) and 2(d). In the following the auxiliary variables are updated using (23).

4.2.4. Quasi-Newton dual ascent

Recently Yfantis et al. proposed the quasi-Newton dual ascent (QNDA) algorithm [2,3]. It is based on a quadratic approximation of the dual function

$$d_B^{(t)}(\lambda_{N_p}) = \frac{1}{2}(\lambda_{N_p} - \lambda_{N_p}^{(t)})^T\mathbf{B}^{(t)}(\lambda_{N_p} - \lambda_{N_p}^{(t)}) + \mathbf{g}^T(\lambda_{N_p}^{(t)})(\lambda_{N_p} - \lambda_{N_p}^{(t)}) + d(\lambda_{N_p}^{(t)}), \tag{24}$$

which is motivated by the fact that the dual function is always concave. The idea is similar to Newton methods. However, since the dual function is nonsmooth no gradients or Hessians can be used in the approximation. Instead, subgradients are used and the approximated Hessian is updated in each iteration via the Broyden–Goldfarb–Shanno-Fletcher (BFGS) scheme,

$$\mathbf{B}^{(t)} = \mathbf{B}^{(t-1)} + \frac{\mathbf{y}^{(t)}\mathbf{y}^{(t),T}}{\mathbf{y}^{(t),T}\mathbf{s}^{(t)}} - \frac{\mathbf{B}^{(t-1)}\mathbf{s}^{(t)}\mathbf{s}^{(t),T}\mathbf{B}^{(t-1),T}}{\mathbf{s}^{(t),T}\mathbf{B}^{(t-1)}\mathbf{s}^{(t)}}, \tag{25}$$

where

$$\mathbf{s}^{(t)} = \lambda_{N_p}^{(t)} - \lambda_{N_p}^{(t-1)} \tag{26}$$

is the variation of the dual variables and

$$\mathbf{y}^{(t)} = \mathbf{g}(\lambda_{N_p}^{(t)}) - \mathbf{g}(\lambda_{N_p}^{(t-1)}) \tag{27}$$

is the variation of the subgradients. The approximated dual function is then optimized to update the dual variables.

The approximation $d_B^{(t)}(\lambda_{N_p})$ is a smooth quadratic function while the actual dual is nonsmooth. This can result in significant approximation errors, especially at or near points of nondifferentiability. To overcome this issue bundle information (16) can be used. In the case of BTM the bundle is used to build a piece-wise linear over-approximation of the dual function. In contrast, in the QNDA algorithm it is used to construct cutting planes which are added as constraints to the update problem. A subgradient is a normal vector to a supporting hyperplane of the dual function. Therefore, the constraint

$$d_B^{(t)}(\lambda_{N_p}^{(t+1)}) \le d(\lambda_{N_p}^{(l)}) + \mathbf{g}^T(\lambda_{N_p}^{(l)})(\lambda_{N_p}^{(t+1)} - \lambda_{N_p}^{(l)}),\ \forall l \in \{t-\tau+1,\dots,t\}. \tag{28}$$

is added to the update problem. This prevents the update from leaving the range of validity of the approximation. Constraints (28) are referred to as bundle cuts and summarized as

$$\mathcal{BC}_{d_B}^{(t)} = \{\lambda_{N_p} \in \mathbb{R}^{n_\lambda} \mid d_B^{(t)}(\lambda_{N_p}) \le d(\lambda_{N_p}^{(l)}) + \mathbf{g}^T(\lambda_{N_p}^{(l)})(\lambda_{N_p} - \lambda_{N_p}^{(l)}),\ \forall l \in \{t-\tau+1,\dots,t\}\}. \tag{29}$$

in the following. Yfantis et al. proposed to update the dual variables in each iteration by solving the optimization problem [2,3]

$$\lambda_{N_p}^{(t+1)} = \arg\max_{\lambda_{N_p}} d_B^{(t)}(\lambda_{N_p}), \tag{30a}$$

$$\text{s. t. } \|\lambda_{N_p} - \lambda_{N_p}^{(t)}\|_2^2 \leq \alpha^{(t)}, \tag{30b}$$

$$\lambda_{N_p} \in \mathcal{BC}_{d_B}^{(t)}, \tag{30c}$$

$$\lambda_{N_p} \geq \mathbf{0}. \tag{30d}$$

Constraint (30b) represents the same trust region that is used in the BTM algorithm (17b). The bundle cuts (30c) sometimes result in a slow convergence during the initial iterations of the algorithm. Therefore, they are only enforced within a certain distance to the optimum depending on a user-defined threshold

$$\|\mathbf{w}_p(\lambda_{N_p}^{(t)})\|_2 \leq \epsilon_b \cdot \|\mathbf{w}_p(\lambda_{N_p}^{(0)})\|_2. \tag{31}$$

Problem (30) is a quadratically constrained quadratic program (QCQP). Numerical tests showed that it is more efficient to reformulate the bundle cuts via an epigraph formulation:

$$\lambda_{N_p}^{(t+1)} = \arg\max_{\lambda_{N_p}, v} d_B^{(t)}(\lambda_{N_p}), \tag{32a}$$

$$\text{s. t. } \|\lambda_{N_p} - \lambda_{N_p}^{(t)}\|_2^2 \leq \alpha^{(t)}, \tag{32b}$$

$$d_B^{(t)}(\lambda_{N_p}) \leq v \tag{32c}$$

$$v \leq d(\lambda_{N_p}^{(l)}) + \mathbf{g}^T(\lambda_{N_p}^{(l)})(\lambda_{N_p}^{(t+1)} - \lambda_{N_p}^{(l)}),\ \forall l \in \{t-\tau+1, \dots, t\}, \tag{32d}$$

$$\lambda_{N_p} \geq \mathbf{0}. \tag{32e}$$

The newly introduced variable v is an over-estimator of the approximated dual function $d_B^{(t)}(\lambda_{N_p})$ (32c). The bundle cuts can then be formulated as linear constraints (32d), which leads to increased numerical robustness and faster convergence.

Preliminary numerical tests additionally showed that the QNDA algorithm tends to converge quickly to the vicinity of the optimum for the DMPC problems. However, it tends to oscillate near the optimum resulting in poor overall performance. To avoid this issue the update strategy of the dual variables is switched to a constrained line search instead of using a trust region within a certain distance to the optimum,

$$\mathbf{s}^{(t)} = \arg\max_{\mathbf{s} \in \mathbb{R}^{n_\lambda}, \hat{\alpha} \in \mathbb{R}} \hat{\alpha}, \tag{33a}$$

$$\text{s. t. } \lambda^{(t)} + \mathbf{s} \in \mathcal{BC}_{d_B}^{(t)}, \tag{33b}$$

$$\mathbf{s} = \hat{\alpha} \nabla d_B^{(t)}(\lambda^{(t)}), \tag{33c}$$

$$\hat{\alpha} \leq \alpha^{(t)}, \tag{33d}$$

$$\lambda^{(t+1)} = \left[\lambda^{(t)} + \mathbf{s}^{(t)}\right]^+. \tag{33e}$$

The update (33) essentially describes an update step in the direction of the gradient of the approximated dual function. Constraint (33b) ensures that the bundle cuts are satisfied, whereby the formulation (32c)–(32d) is employed. Constraint (33c) gives the search direction. The step size is limited by constraint (33d), which replaces the trust region constraint. The dual variables are then updated in the computed search direction and projected onto the positive orthant due to the nonnegativity constraints on the dual variables. The update steps (33) are used if

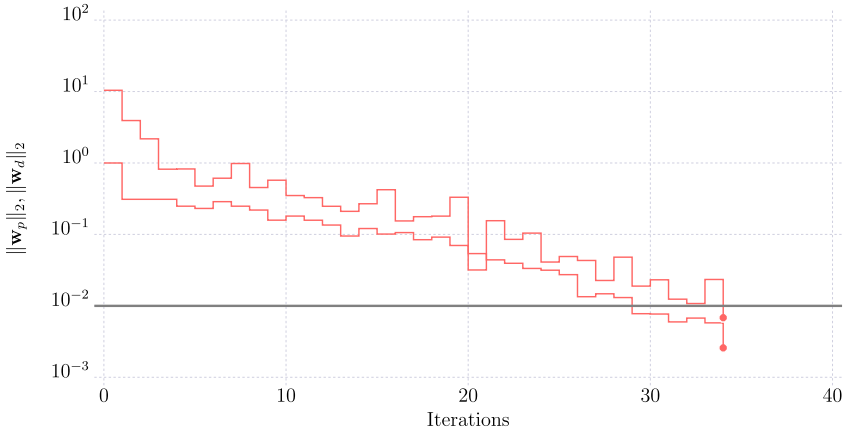
$$\|\mathbf{w}_p(\lambda^{(t)})\|_2 \leq \epsilon_l \cdot \|\mathbf{w}_p(\lambda^{(0)})\|_2, \tag{34}$$

where ϵ_l has to be sufficiently small to ensure that the algorithm has reached the vicinity of the optimal dual variables.

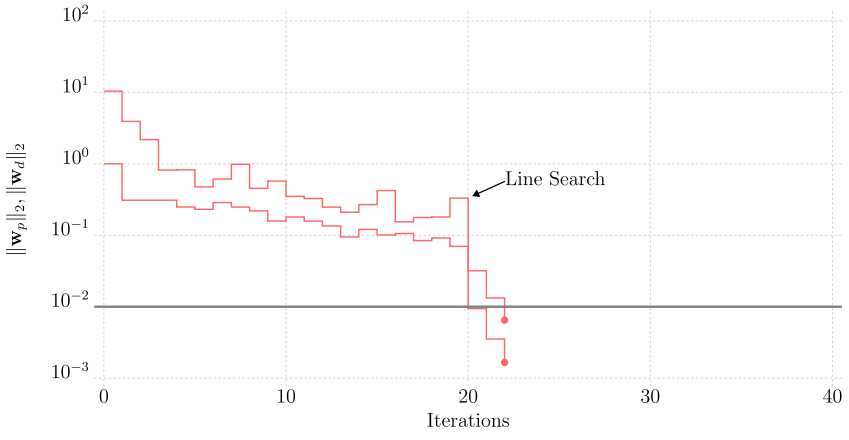
The effect of the line search strategy is illustrated in Fig. 3. The evolution of the primal and dual residuals without a line search is shown in Fig. 3(a). In comparison, the line search strategy (33) is employed in Fig. 3(b). This results in faster convergence in the vicinity of the optimum.

4.3. Dual decomposition of dynamically coupled DMPC problems

For the sake of completeness, this section briefly demonstrates how dual decomposition can be applied to distributedly solve DMPC problems with coupled dynamics (4). Also for the sake of brevity, it is assumed that the subsystems are only coupled through their states, i.e., $\mathbf{B}_{ij} = \mathbf{0}$ for $i \neq j$. First the subset $\mathcal{N}_i \subset \mathcal{I}$ is defined which contains all neighbors of subsystem i, i.e., $\mathbf{A}_{ij} \neq \mathbf{0},\ \forall j \in \mathcal{N}_i$. Each subsystem i can now be augmented by additional decision variables $\mathbf{v}_{ij}$ which represent a local copy of the states of a neighboring subsystem j. With this, the system-wide problem (4) can be reformulated as



(a) Evolution of the primal and dual residuals without the line search strategy.



(b) Evolution of the primal and dual residuals with the line search strategy.

Fig. 3. Comparison of the convergence of the QNDA algorithm for a DMPC benchmark problem with and without the line search update (33). The plots show the evolutions of the primal and dual residuals for a DMPC problem with $N_s = 20, n_{\mathbf{x}} = 3, n_{\mathbf{u}} = 2, n_{\mathbf{u}} = 2, N_p = 15$. The line search is used once the algorithm reaches the vicinity of the optimum.

$$\min_{\mathbf{x}^{0:N_p},\mathbf{u}^{0:N_p-1},\mathbf{v}^{0:N_p}} \sum_{i\in\mathcal{I}} \left[J_i^f(\mathbf{x}_i^{N_p}) + \sum_{k=0}^{N_p-1} J_i(\mathbf{x}_i^k, \mathbf{u}_i^k) \right], \tag{35a}$$

$$\text{s. t. } \mathbf{x}_i^{k+1} = \mathbf{A}_{ii}\mathbf{x}_i^k + \sum_{j\in\mathcal{N}_i} \mathbf{A}_{ij}\mathbf{v}_{ij}^k + \mathbf{B}_i\mathbf{u}_i^k, \ \forall i \in \mathcal{I},$$

$$k = 0, \dots, N_p - 1, \tag{35b}$$

$$\mathbf{x}_i^0 = \tilde{\mathbf{x}}(t_0), \ \forall i \in \mathcal{I}, \tag{35c}$$

$$\mathbf{x}_i^k \in \mathcal{X}_i \subset \mathbb{R}^{n_{\mathbf{x}_i}}, \ \forall i \in \mathcal{I}, k = 0, \dots, N_p, \tag{35d}$$

$$\mathbf{u}_i^k \in \mathcal{U}_i \subset \mathbb{R}^{n_{\mathbf{u}_i}}, \ \forall i \in \mathcal{I}, k = 0, \dots, N_p - 1, \tag{35e}$$

$$\mathbf{v}_{ij}^k - \mathbf{x}_j^k = \mathbf{0}, \ \forall i \in \mathcal{I}, j \in \mathcal{N}_i, k = 0, \dots, N_p. \tag{35f}$$

The subsystems in problem (35) are now coupled through the constraints (35f) for which dual variables $\lambda_{ij}^k, \ i \neq j$, can be introduced. The resulting individual Lagrange function for a subsystem i can then be defined as

$$\mathcal{L}_i(\mathbf{x}_i^{k:N_p}, \mathbf{u}_i^{0:N_p-1}, \mathbf{v}_{ij}^{0:N_p}, \boldsymbol{\lambda}_{ij}^{0:N_p}) = J_i^f(\mathbf{x}_i^{N_p}) + \sum_{k=0}^{N_p-1} J_i(\mathbf{x}_i^k, \mathbf{u}_i^k) + \sum_{k=0}^{N_p} \sum_{j\in\mathcal{N}_i} [\boldsymbol{\lambda}_{ij}^{k,T}\mathbf{v}_{ij}^k - \boldsymbol{\lambda}_{ji}^{k,T}\mathbf{x}_i^k]. \tag{36}$$

The system-wide MPC problem (35) can then be solved in a distributed manner by optimizing the individual Lagrange functions in parallel and coordinating the solution via the dual variables.

5. Numerical analysis for DMPC problems

In this section, several DMPC benchmark problems are solved using the subgradient method, BTM, ADMM, and QNDA. The generated benchmark problems have the structure of problem (5) with a reference tracking objective (2) and a terminal cost (3). The objective function matrices were generated randomly as symmetric positive definite matrices,

$$\mathbf{H}_{\mathbf{x}_i} = \mathbf{H}_{\mathbf{x}_i}^f = \mathbf{N}_{\mathbf{x}_i}^T \mathbf{N}_{\mathbf{x}_i} \in \mathbb{R}^{n_{\mathbf{x}_i} \times n_{\mathbf{x}_i}}, \tag{37}$$

$$\mathbf{H}_{\mathbf{u}_i} = \mathbf{N}_{\mathbf{u}_i}^T \mathbf{N}_{\mathbf{u}_i} \in \mathbb{R}^{n_{\mathbf{u}_i} \times n_{\mathbf{u}_i}}, \tag{38}$$

where the elements of $\mathbf{N}_{\mathbf{x}_i}$ and $\mathbf{N}_{\mathbf{u}_i}$ were drawn from a normal distribution $[\mathbf{N}_{\mathbf{x}_i}]_{l,j}, [\mathbf{N}_{\mathbf{u}_i}]_{l,j} \in \mathcal{N}(\mu = 0, \sigma = 1)$.

The system matrices $\mathbf{A}_i$ were generated such that the resulting subsystems are stable. A discrete-time system is stable if all eigenvalues of the matrix $\mathbf{A}_i$ lie within the unit sphere [48]. The matrices were thus computed as

$$\mathbf{A}_i = \mathbf{S}_i \mathbf{D}_i \mathbf{S}_i^{-1}, \tag{39}$$

where $\mathbf{D}_i$ is a diagonal matrix containing the eigenvalues of the matrix $\mathbf{A}_i$ drawn from a continuous uniform distribution $[\mathbf{D}_i]_{l,l} \in \mathcal{U}_c(-1, 1)$ while the elements of the matrix $\mathbf{S}_i$ were drawn from the continuous uniform distribution $\mathcal{U}_c(-3, 3)$. The elements of the input matrix $\mathbf{B}_i$ were drawn from the continuous uniform distribution $\mathcal{U}_c(-2, 2)$.

The individual constraints were generated as intersections of ellipsoids around the origin,

$$\mathcal{X}_i = \{\mathbf{x}_i \in \mathbb{R}^{n_{\mathbf{x}_i}} \mid \mathbf{x}_i^T \mathbf{G}_{\mathbf{x}_i,l} \mathbf{x}_i \leq p_{\mathbf{x}_i,l}^2,\ l = 1, \dots, n_c\}, \tag{40}$$

$$\mathcal{U}_i = \{\mathbf{u}_i \in \mathbb{R}^{n_{\mathbf{u}_i}} \mid \mathbf{u}_i^T \mathbf{G}_{\mathbf{u}_i,l} \mathbf{u}_i \leq p_{\mathbf{u}_i,l}^2,\ l = 1, \dots, n_c\}. \tag{41}$$

The matrices $\mathbf{G}_{\mathbf{x}_i,l}$ and $\mathbf{G}_{\mathbf{u}_i,l}$ were generated in the same way as the objective matrices while the right-hand sided $p_{\mathbf{x}_i,l}$ and $p_{\mathbf{u}_i,l}$ were drawn from the continuous uniform distribution $\mathcal{U}_c(1, 5)$.

Ideally the reference trajectory $\mathbf{x}_i^{\mathrm{ref},0:N_p}$ should be feasible. Therefore, the elements of the reference trajectory were drawn from the continuous uniform distribution $\mathcal{U}_c(-2, 2)$ at each time point until a feasible point was found, i.e., until $\mathbf{x}_i^{\mathrm{ref},k} \in \mathcal{X}_i$. The initial state $\tilde{\mathbf{x}}(t_0)$ was generated in the same way.

The elements of the resource matrix $\mathbf{R}_i$ were first drawn from the continuous distribution $\mathcal{U}_c(1, 2)$. Afterward, they were altered such that their sign was flipped or they were set to zero through the uniform discrete distribution $\mathcal{U}_d[-1, 0, 1]$,

$$\mathbf{R}_i = \mathbf{D}_i \circ \mathbf{C}_i \in \mathbb{R}^{n_{\mathbf{r}} \times n_{\mathbf{u}_i}},\ [\mathbf{D}_i]_{l,j} \in \mathcal{U}_c(1, 2), [\mathbf{C}_i]_{l,j} \in \mathcal{U}_d[-1, 0, 1]. \tag{42}$$

This ensures that not all subsystems are connected to each resource network and that some produce while others consume certain resources. The maximum resource utilization should be generated such that the decentralized solution, i.e., for $\lambda = \mathbf{0}$, is infeasible, while the system-wide problem is feasible. To this end, the optimal input trajectory for the system-wide problem without the resources constraints was first computed,

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

$$\text{s. t. } \mathbf{x}_i^{k+1} = \mathbf{A}_i \mathbf{x}_i^k + \mathbf{B}_i \mathbf{u}_i^k,\ \forall i \in \mathcal{I},\ k = 0, \dots, N_p - 1, \tag{43a}$$

$$\mathbf{x}_i^0 = \tilde{\mathbf{x}}(t_0),\ \forall i \in \mathcal{I}, \tag{43b}$$

$$\mathbf{x}_i^k \in \mathcal{X}_i \subset \mathbb{R}^{n_{\mathbf{x}_i}},\ \forall i \in \mathcal{I}, k = 0, \dots, N_p, \tag{43c}$$

$$\mathbf{u}_i^k \in \mathcal{U}_i \subset \mathbb{R}^{n_{\mathbf{u}_i}},\ \forall i \in \mathcal{I},$$

$$k = 0, \dots, N_p - 1. \tag{43d}$$

Using the found optimal solution the optimal unconstrained resource utilization can be computed,

$$\mathbf{r}^{k,*} := \sum_{i \in \mathcal{I}} \mathbf{R}_i \mathbf{u}_i^{k,*},\ k = 0, \dots, N_p - 1. \tag{44}$$

The maximum resource utilization was then computed by tightening the optimal unconstrained resource utilization,

$$[\mathbf{r}_{\max}^k]_l = [\mathbf{r}^{k,*}]_l - \beta_l^k |[\mathbf{r}^{k,*}]_l|,\ k = 0, \dots, N_p - 1,\ l = 1, \dots, n_{\mathbf{r}},\ \beta_l^k \in \mathcal{U}_c(0, 0.2). \tag{45}$$

After the maximum resource utilization was generated, the feasibility of the decentralized and the system-wide problem was verified.

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

Number of subproblems: $N_s \in \{5, 10, 20, 50\}$,

Table 1
Parameter settings of the distributed optimization algorithms for the DMPC benchmark problems.

	Value	Description	Algorithms
$\lambda^{(0)}$	$\mathbf{0}$	initial dual variables	All
$\alpha^{(0)}$	1	initial step size/trust region parameter	SG, BTM, QNDA
$t_{\max}$	500	maximum number of iterations	All
ϵ_p	10^{-2}	primal residual convergence tolerance	All
ϵ_d	10^{-2}	dual residual convergence tolerance	All
ϵ_b	0.6	bundle cuts threshold	QNDA
ϵ_l	10^{-2}	line search threshold	QNDA
$\rho^{(0)}$	10^{-3}	initial regularization parameter	ADMM
τ_{incr}	1.25	see (21)	ADMM
τ_{decr}	1.1	see (21)	ADMM
μ	10	see (21)	ADMM
$\mathbf{z}^{(0)}$	$\mathbf{0}$	initial auxiliary variables	ADMM
τ	150	allowed age of data points	BTM, QNDA
$\mathbf{B}^{(0)}$	$-\mathbf{I}$	Initial approximated Hessian	QNDA

Number of states: $n_{\mathbf{x}} \in \{2,3,4,5\}$,

Number of inputs: $n_{\mathbf{u}} \in \{2,3,4,5\}, n_{\mathbf{x}} \geq n_{\mathbf{u}}$

Number of resources: $n_{\mathbf{r}} \in \{2,3,4,5\}, n_{\mathbf{u}} \geq n_{\mathbf{r}}$

Prediction horizon: $N_p \in \{10,15,20\}$.

Each subsystem contains the same number of states and inputs, i.e., $n_{\mathbf{x}_i} = n_{\mathbf{x}}$, $n_{\mathbf{u}_i} = n_{\mathbf{u}}$, $\forall i \in \mathcal{I}$. The number of state and input constraints was set equal to the number of states, i.e., $n_c = n_{\mathbf{x}}$. Ten benchmark problems were generated for each combination resulting in a total of 2400 DMPC problems.

All algorithms and DMPC subproblems were implemented in the programming language Julia [49] using the optimization toolbox JuMP [50]. All DMPC subproblems were solved using the commercial solver Gurobi [51]. The update problem of BTM is guaranteed to have a linear objective with affine and convex quadratic constraints, therefore it was solved using Gurobi. The update problems of QNDA were solved using IPOPT [52]. All computations throughout were performed on a standard Laptop PC (Intel(R) Core(TM) i5-6200U CPU @ 2.30 GHz, 8 GB RAM).

To assess the efficiency of the different algorithms the computation time required for the solution of a distributed optimization problem was computed as [32]

$$T_{\text{comp}} = N_{\text{iter}} \cdot T_{\text{comm}} + \sum_{t=1}^{N_{\text{iter}}} (T_{\text{update}}^{(t)} + \max_{i \in \mathcal{I}} T_{\text{sub},i}^{(t)}), \tag{46}$$

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.

5.1. Parameter settings for DMPC problems

The parameters for the DMPC problems were set by trial and error to achieve a good compromise between robustness and rate of convergence. The initial step size parameter (SG)/ trust region parameter (BTM, QNDA) was set to $\alpha^{(0)} = 1$ and updated according to

$$\alpha^{(t)} = \frac{\alpha^{(0)}}{\max\{\|\mathbf{w}_p^{(0)}\|_2, \ldots, \|\mathbf{w}_p^{(t)}\|_2\}}. \tag{47}$$

The ADMM algorithm exhibited divergence when tuned too aggressively. Therefore the initial regularization parameter was set $\rho^{(0)} = 10^{-3}$. The tuning parameters were set to $\tau_{\text{incr}} = 1.25$, $\tau_{\text{decr}} = 1.1$ and $\mu = 10$. The bundle cuts threshold was set to $\epsilon_b = 0.6$ (cf. Eq. (31)) and the threshold for the line search (33) was set to $\epsilon_l = 10^{-2}$ (cf. Eq. (34)). The age parameter for BTM and QNDA was set to $\tau = 150$. All algorithms were initialized with $\lambda^{(0)} = \mathbf{0}$ and $\mathbf{z}_i^{(0)} = \mathbf{0}$, $\forall i \in \mathcal{I}$ (for ADMM). The approximated Hessian for the QNDA algorithm was initialized with the negative identity matrix. All algorithms were terminated after $t_{\max} = 500$ iterations or when the primal and dual residuals reached the threshold $\epsilon_p = \epsilon_d = 10^{-2}$. All parameters are summarized in Table 1.

5.2. Results for DMPC problems

The results of the distributed optimization of the DMPC benchmark problems are depicted in Fig. 4 and summarized in Table 2. The results show that the ADMM and QNDA algorithms can solve almost all DMPC benchmark problems. While ADMM can solve slightly more problems, the QNDA algorithm outperforms ADMM in terms of required iterations, computation time, and quality of

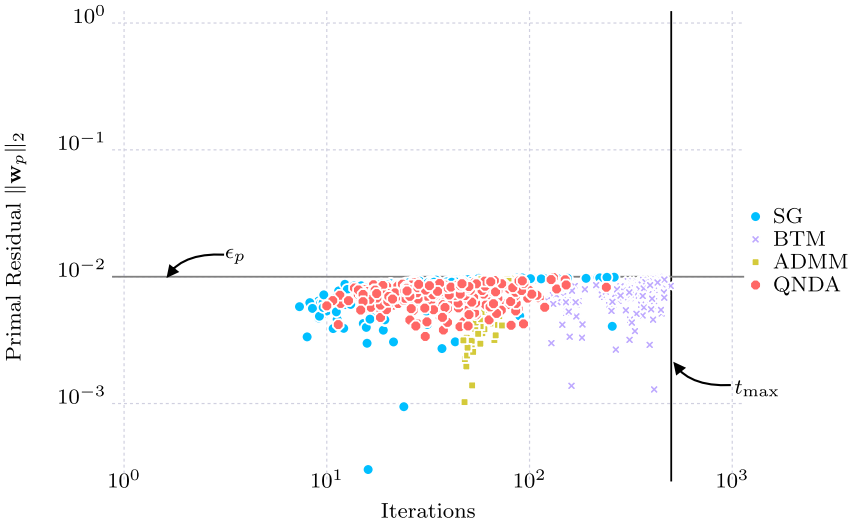


Fig. 4. Values of the primal residuals upon termination for the distributed MPC problems. Each data point represents the mean values of the converged problem instances for a tuple $(N_s, n_{\mathbf{x}}, n_{\mathbf{u}}, n_{\mathbf{r}}, N_p)$ (cf. Table B.3).

Table 2
Summary of the results for the distributed optimization of the DMPC benchmark problems (mean values of the converged instances only), $\bar{t}$: mean number of iterations until convergence, $\overline{T_{\text{comp}}}$: mean computation time of converged runs (in s), $\overline{\|\mathbf{w}_p\|_2}$: mean primal residual of converged runs ($\times 10^{-3}$), $\%_c$: percentage of converged runs within $t_{\max}$ iterations.

Algorithm	$\bar{t}$	$\overline{T_{\text{comp}}}$	$\overline{\|\mathbf{w}_p\|_2}$	$\%_c$
SG	42.83	**35.24**	8.18	80.96
BTM	273.83	228.17	7.09	57.58
ADMM	62.24	51.92	7.65	**99.21**
QNDA	**42.6**	38.31	**7.07**	98.21

found solutions. The subgradient method exhibits a similar number of required iterations for its converged runs as QNDA with a better computation time due to the less expensive update steps. However, the subgradient method solves far fewer problems than the ADMM and QNDA algorithms. BTM algorithm exhibits rather poor performance for the DMPC benchmark problems. A more detailed summary of the results is given in Table B.3 in the appendix.

The considered DMPC problems are convex. Thus, the optimal duality gap, i.e., the difference between the optimal primal and dual objectives, is zero. For nonconvex problems, such as mixed-integer programs, the optimal duality gap tends to decrease as the number of subsystems increases [53]. Additionally, the degree of nonsmoothness tends to decrease with an increase in the number of subsystems [3,54]. In the case of the considered convex DMPC problems this results in higher-quality solutions being found by the different algorithms, as illustrated in Fig. 5. Note that the convergence tolerance ϵ_p of the algorithms, i.e., the equivalent to a feasibility tolerance, is chosen relatively large (cf. Table 1). The reason is that a low feasibility tolerance for the coupling constraints is usually not necessary in practice. Furthermore, algorithms like ADMM tend to converge quickly to solutions with moderate tolerances, while requiring significantly more iterations to further improve them [44]. Fig. 5 shows that the distributed optimization algorithms tend to find solutions with tighter tolerances as the number of subsystems increases. Note that only the mean values of the runs that actually converged are depicted.

Fig. 6 shows the mean number of iterations performed by the different algorithms for different numbers of subsystems to analyze their scalability. It can be seen that in the case of the subgradient method and BTM the mean number of iterations increases as the number of subsystems increases. In contrast, the number of iterations decreases or stays relatively consistent for ADMM and QNDA. Furthermore, they require far fewer iterations, underlining their better efficiency and scalability. In contrast, Fig. 7 shows that the number of resources does not have a substantial influence on the number of iterations.

Fig. 8 shows the results for the distributed optimization of a benchmark problem with three shared resources and a prediction horizon of 15. i.e., $\lambda \in \mathbb{R}^{42}$. The subgradient method and QNDA exhibit the fastest convergence. ADMM exhibits a small oscillation but still converges quickly. The BTM algorithm exhibits significant oscillations and converges very slowly compared to the other algorithms. However, Fig. 8(b) shows that the oscillations mainly take place in the vicinity of the optimal dual variables, indicating that the resource utilization is sensitive to the change of the dual variables.

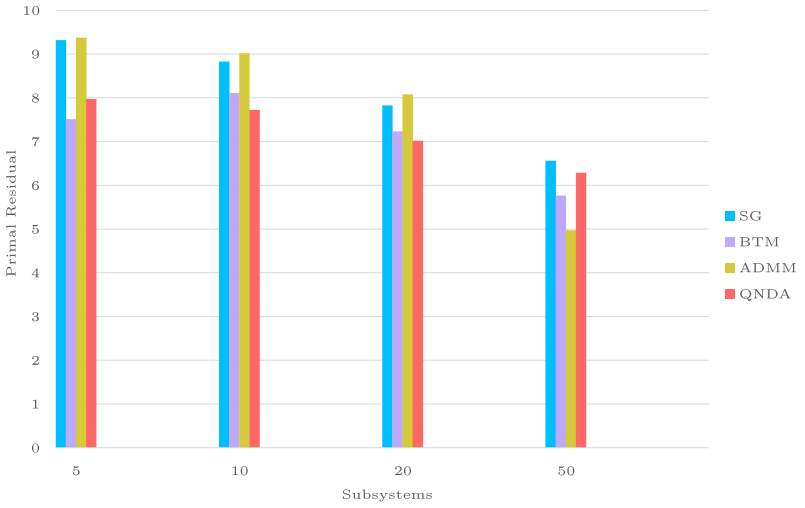


Fig. 5. Comparison of the mean primal residuals of converged runs for different numbers of subsystems.

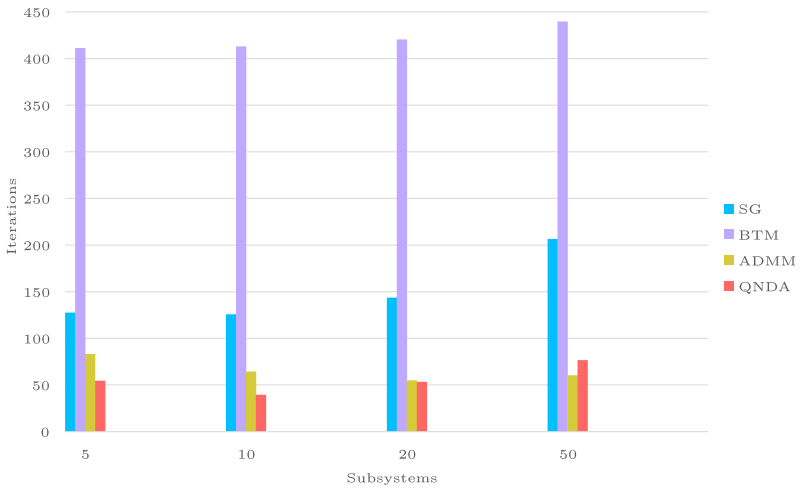


Fig. 6. Comparison of the mean number of iterations performed by the different algorithms for different numbers of subsystems.

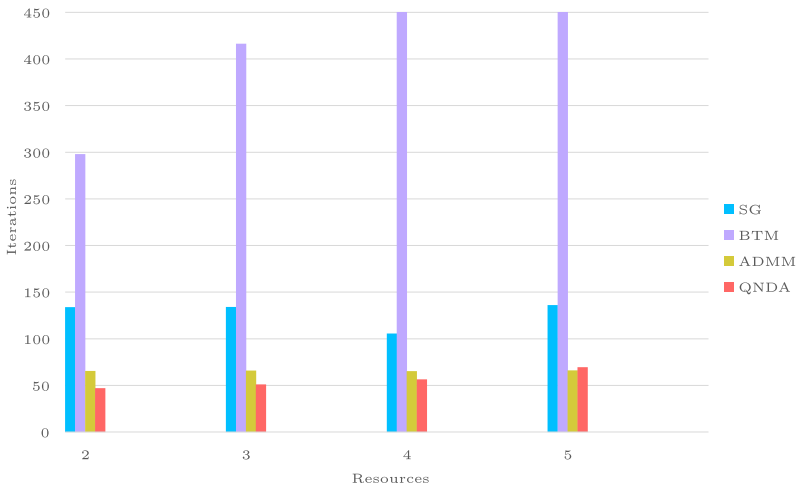
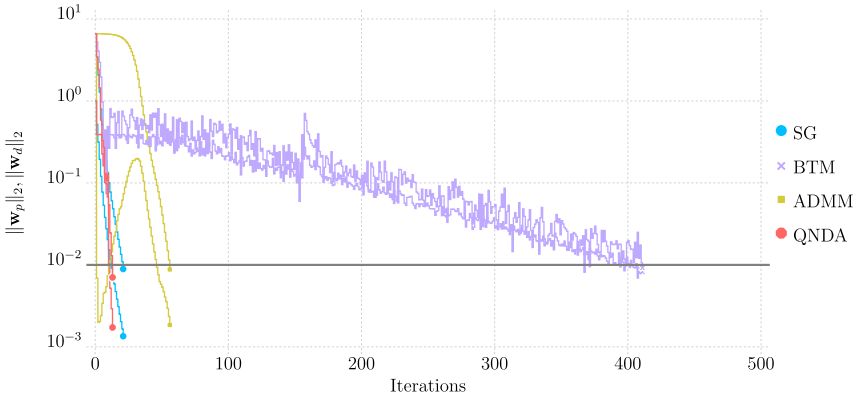
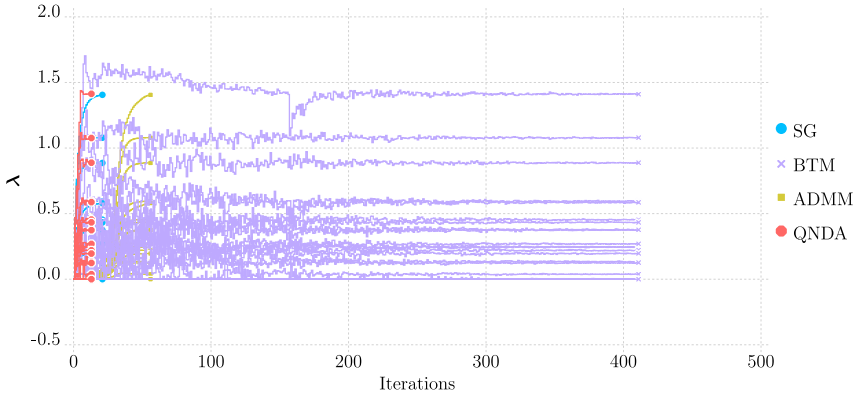


Fig. 7. Comparison of the mean number of iterations performed by the different algorithms for different numbers of resources.

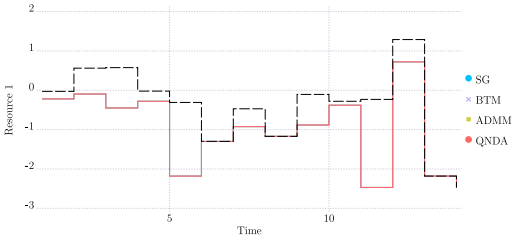


(a) Evolution of the primal and dual residuals.

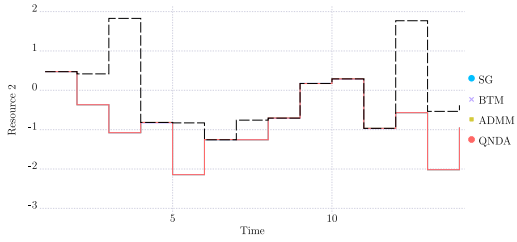


(b) Evolution of the dual variables.

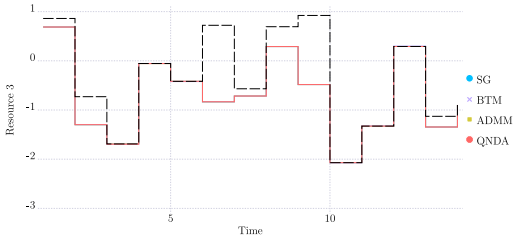
Fig. 8. Results for a DMPC benchmark problem with $N_s = 10, n_{\mathbf{x}} = 5, n_{\mathbf{u}} = 5, n_{\mathbf{r}} = 3, N_p = 15$.



(a) Utilization of resource 1.

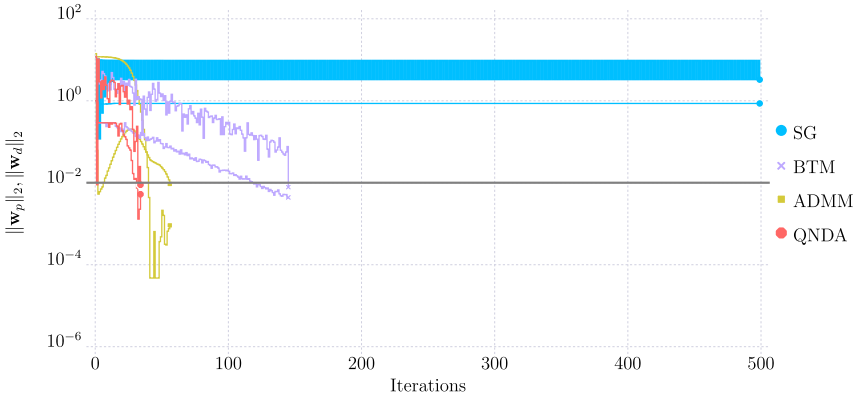


(b) Utilization of resource 2.

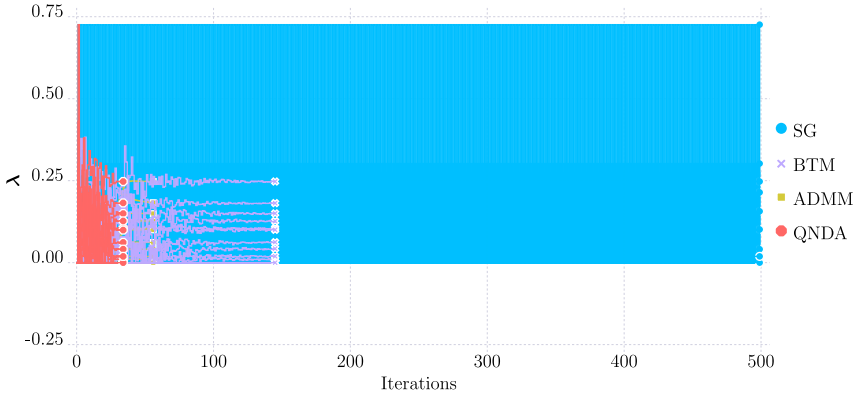


(c) Utilization of resource 3.

Fig. 9. Resource utilization upon termination for a DMPC benchmark problem with $N_s = 10, n_{\mathbf{x}} = 5, n_{\mathbf{u}} = 5, n_{\mathbf{r}} = 3, N_p = 15$.

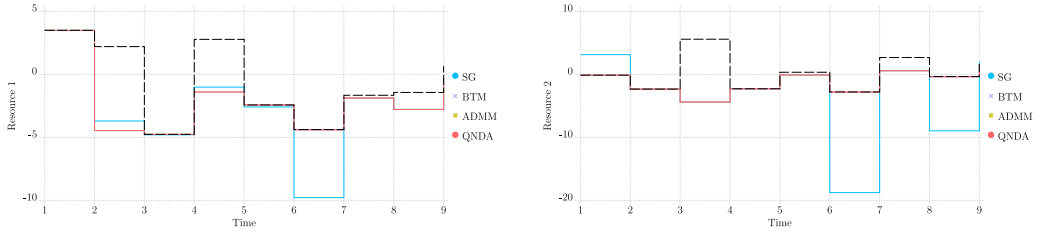


(a) Evolution of the primal and dual residuals.



(b) Evolution of the dual variables.

Fig. 10. Results for a DMPC benchmark problem with $N_s = 50, n_{\mathbf{x}} = 4, n_{\mathbf{u}} = 4, n_{\mathbf{r}} = 2, N_p = 10$.



(a) Utilization of resource 1.

(b) Utilization of resource 2.

Fig. 11. Resource utilization upon termination for a DMPC benchmark problem with $N_s = 50, n_{\mathbf{x}} = 4, n_{\mathbf{u}} = 4, n_{\mathbf{r}} = 3, N_p = 10$.

Fig. 9 depicts the utilization of the shared limited resources upon the convergence of the distributed optimization algorithms for the DMPC benchmark problem in Fig. 8. All algorithms converge to the same resource utilization, i.e., to the optimal dual variables.

The results in Table 2 show that the subgradient method exhibits similar performance to QNDA for its converged runs. However, aggressive tuning is necessary to obtain this performance, which comes at the cost of robustness.

Fig. 10 shows the results for a DMPC benchmark problem with two shared resources and a prediction horizon of 10, i.e., $\lambda \in \mathbb{R}^{18}$. The subgradient method is not able to converge as it exhibits extreme oscillations due to the aggressive step size parameter. Compared to that, all other algorithms converge, with QNDA being the most efficient. Note that both the QNDA and BTM algorithms employ the same parameter as the subgradient method for their trust region and, in the case of the QNDA algorithm, for the line search updates.

Fig. 11 shows the utilization of the shared limited resources for the DMPC benchmark problem in Fig. 10. The BTM, ADMM, and QNDA algorithms converge to the optimal resource utilization. Fig. 11(b) shows that the subgradient method terminates with an infeasible utilization of resource 2.

6. Conclusion

This paper demonstrated how dual decomposition-based distributed optimization can be applied to solve distributed MPC problems that are coupled through shared limited resources. The recently proposed quasi-Newton dual ascent algorithm was extended by an epigraph formulation for the bundle cuts and by a constrained line search update step in the vicinity of the optimum. The efficiency of the algorithm was demonstrated on a large number of benchmark problems. While the ADMM algorithm was able to solve the most benchmark problems and the subgradient method showed similar performance to QNDA for its converged runs, the QNDA algorithm exhibits the best balance in terms of performance and robustness. Furthermore, the benchmarks showed that ADMM and QNDA scale well with the number of subsystems, both in terms of the quality of the found solutions and the number of required iterations. The number of resources exhibited limited impact on the number of required iterations for all algorithms.

As shown in Section 4.3 dual decomposition can also be applied to DMPC problems where the subsystems are coupled through their dynamics. The creation of a comprehensive benchmark set and the evaluation of different algorithms for these types of DMPC problems can be investigated in the future. Furthermore, all systems considered in this paper were governed by linear dynamics. As dual decomposition-based distributed optimization is an iterative hierarchical approach, i.e., many communication rounds have to be performed until the control inputs for the next time step can be applied, it is most suitable for systems with slow dynamics, which can usually be linearized without a significant loss of prediction accuracy. However, as the efficiency and robustness of nonlinear optimization solvers increase, the use of nonlinear models in MPC is also increasing. Therefore, the application of dual decomposition-based DMPC to systems with nonlinear dynamics and constraints should be further investigated. Nonlinear model predictive control (NMPC) problems are usually nonconvex, which in turn means that an optimal dual solution does not necessarily correspond to a feasible primal solution. In general, heuristics have to be used to obtain a feasible solution in the case of dual decomposition-based distributed optimization with nonconvex subproblems. Nevertheless, the value of the dual function still provides a lower bound on the objective value at the optimal primal solution. Finally, the literature is still lacking in real-life industrial-scale DMPC benchmarks, especially with a large number of subsystems. While many aspects of distributed optimization can be demonstrated in laboratory-scale benchmarks, the true impact of a coordination scheme can only be properly assessed when applied to a large-scale system.

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.

Appendix A. DMPC benchmark problems

All used benchmark problems are available at: https://github.com/VaYf/DMPC-Benchmark-Problems

Appendix B. Results of the DMPC benchmarks

The results for the DMPC benchmark problems are summarized in Table B.3.

Data availability

A link to the data used for this research is included in the manuscript.

Table B.3
Summary of the results for the distributed optimization of the DMPC benchmark problems (mean values of the converged instances only), $\bar{i}$: 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.

DMPC	SG				BTM			
	$\bar{i}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{i}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
Mean	**42.83**	**8.18**	**35.24**	**80.96**	**273.83**	**7.09**	**228.17**	**57.58**
$N_s = 5, n_r = 2$	75.96	8.92	62.18	83.33	248.51	5.8	206.64	83.0
$N_s = 5, n_r = 3$	79.18	9.36	65.1	84.44	309.54	6.93	257.53	52.22
$N_s = 5, n_r = 4$	81.44	9.38	67.24	93.33	373.43	8.34	310.17	31.11
$N_s = 5, n_r = 5$	88.44	9.62	73.32	90.0	425.67	8.98	354.84	10.0
$N_s = 10, n_r = 2$	47.38	8.63	38.91	88.67	244.71	7.19	203.75	86.67
$N_s = 10, n_r = 3$	44.09	8.87	36.38	88.89	305.44	7.93	253.95	47.22
$N_s = 10, n_r = 4$	39.2	9.05	32.4	92.22	389.08	7.9	323.86	27.78
$N_s = 10, n_r = 5$	29.04	8.76	24.14	86.67	458.0	9.42	382.54	10.0

(*continued on next page*)

Table B.3 (*continued*).

$N_s = 20, n_r = 2$	26.87	7.65	22.07	84.33	261.61	7.38	218.46	84.33
$N_s = 20, n_r = 3$	21.02	8.01	17.36	88.89	301.23	8.0	250.84	43.89
$N_s = 20, n_r = 4$	18.73	8.12	15.53	90.0	399.12	7.4	332.71	26.67
$N_s = 20, n_r = 5$	16.15	7.53	13.42	86.67	404.0	6.16	338.32	3.33
$N_s = 50, n_r = 2$	21.98	6.21	18.12	65.0	249.21	6.86	208.35	70.67
$N_s = 50, n_r = 3$	16.33	6.98	13.52	57.22	317.98	8.05	264.98	30.56
$N_s = 50, n_r = 4$	15.56	6.5	12.92	67.78	431.18	8.16	360.08	12.22
$N_s = 50, n_r = 5$	17.44	6.56	14.51	53.33	–	–	–	–
DMPC	ADMM				QNDA			
	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$	$\bar{t}$	$\overline{\|\mathbf{w}_p\|_2}$	$\overline{T_{\text{comp}}}$	$\%_c$
Mean	**62.24**	**7.65**	**51.92**	**99.21**	**42.6**	**7.07**	**38.31**	**98.21**
$N_s = 5, n_r = 2$	68.5	9.22	56.33	95.33	38.93	7.48	35.09	92.0
$N_s = 5, n_r = 3$	75.28	9.38	62.32	97.78	37.7	7.99	34.64	95.56
$N_s = 5, n_r = 4$	78.77	9.32	65.7	100.0	30.11	7.8	27.28	97.78
$N_s = 5, n_r = 5$	80.93	9.59	68.17	100.0	43.37	8.63	43.53	100.0
$N_s = 10, n_r = 2$	60.5	8.86	49.88	99.67	30.63	7.41	26.58	98.0
$N_s = 10, n_r = 3$	63.36	9.0	52.62	100.0	30.1	7.54	26.34	99.44
$N_s = 10, n_r = 4$	66.06	9.09	55.35	100.0	40.56	7.93	36.68	100.0
$N_s = 10, n_r = 5$	66.5	9.12	56.35	100.0	45.17	8.02	42.11	100.0
$N_s = 20, n_r = 2$	54.66	7.5	45.24	100.0	29.46	6.79	25.41	99.67
$N_s = 20, n_r = 3$	55.0	7.97	45.96	100.0	42.42	6.97	37.56	100.0
$N_s = 20, n_r = 4$	55.2	8.28	46.63	100.0	62.19	7.11	57.37	100.0
$N_s = 20, n_r = 5$	54.73	8.57	46.84	100.0	77.9	7.2	74.7	100.0
$N_s = 50, n_r = 2$	57.15	3.88	47.82	100.0	41.31	6.15	36.06	100.0
$N_s = 50, n_r = 3$	60.78	5.14	51.65	100.0	68.63	5.71	61.71	99.44
$N_s = 50, n_r = 4$	61.62	5.42	53.29	100.0	82.43	6.49	77.11	100.0
$N_s = 50, n_r = 5$	61.97	5.47	54.67	100.0	111.93	6.8	109.76	100.0

References

[1] Bakule L. Decentralized control: An overview. Ann Rev Control 2008;32(1):87–98.
[2] Yfantis V, Ruskowski M. 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, p. 31–6.
[3] Yfantis V, Wenzel S, Wagner A, Ruskowski M, Engell S. Hierarchical distributed optimization of constraint-coupled convex and mixed-integer programs using approximations of the dual function. EURO J Comput Optim 2023;11:100058.
[4] Alvarado I, Limon D, De La Peña DM, Maestre JM, Ridao M, Scheu H, et al. A comparative analysis of distributed MPC techniques applied to the HD-MPC four-tank benchmark. J Process Control 2011;21(5):800–15.
[5] Maxeiner LS, Engell S. Comparison of dual based optimization methods for distributed trajectory optimization of coupled semi-batch processes. Optim Eng 2020;21(3):761–802.
[6] Stomberg G, Engelmann A, Faulwasser T. A compendium of optimization algorithms for distributed linear-quadratic MPC. at-Automatisierungstechnik 2022;70(4):317–30.
[7] Conte C, Summers T, Zeilinger M, Morari M, Jones C. Computational aspects of distributed optimization in model predictive control. In: 51st IEEE conference on decision and control. CDC, IEEE; 2012, p. 6819–24.
[8] Necoara I, Nedelcu V, Dumitrache I. Parallel and distributed optimization methods for estimation and control in networks. J Process Control 2011;21(5):756–66.
[9] Mayne D, Rawlings J, Rao C, Scokaert P. Constrained model predictive control: Stability and optimality. Automatica 2000;36(6):789–814.
[10] Engell S. Feedback control for optimal process operation. J Process Control 2007;17(3):203–19.
[11] Ellis M, Durand H, Christofides P. A tutorial review of economic model predictive control methods. J Process Control 2014;24(8):1156–78.
[12] Scattolini R. Architectures for distributed and hierarchical model predictive control–a review. J Process Control 2009;19(5):723–31.
[13] Christofides P, Scattolini R, de la Pena DM, Liu J. Distributed model predictive control: A tutorial review and future research directions. Comput Chem Eng 2013;51:21–41.
[14] Camponogara E, Jia D, Krogh B, Talukdar S. Distributed model predictive control. IEEE Control Syst Mag 2002;22(1):44–52.
[15] Maestre J, de la Peña D, Camacho E, Alamo T. Distributed model predictive control based on agent negotiation. J Process Control 2011;21(5):685–97.
[16] Gafur N, Kanagalingam G, Wagner A, Ruskowski M. Dynamic collision and deadlock avoidance for multiple robotic manipulators. IEEE Access 2022.
[17] Sandell N, Varaiya P, Athans M. A survey of decentralized control methods for large scale systems. Syst Eng Power, US Dept. of Commerce 1975;334–5.
[18] Negenborn R, Maestre J. Distributed model predictive control: An overview and roadmap of future research opportunities. IEEE Control Syst Mag 2014;34(4):87–97.
[19] Müller MA, Allgöwer F. Economic and distributed model predictive control: Recent developments in optimization-based control. SICE J Control, Meas, Syst Integr 2017;10(2):39–52.
[20] Stewart B, Venkat A, Rawlings J, Wright S, Pannocchia G. Cooperative distributed model predictive control. Systems Control Lett 2010;59(8):460–9.
[21] Maestre J, de la Peña D, Camacho EF. Distributed model predictive control based on a cooperative game. Optim Control Appl Methods 2011;32(2):153–76.
[22] Zheng Y, Li S, Qiu H. Networked coordination-based distributed model predictive control for large-scale system. IEEE Trans Control Syst Technol 2012;21(3):991–8.
[23] Giselsson P, Rantzer A. Distributed model predictive control with suboptimality and stability guarantees. In: 49th IEEE conference on decision and control. CDC, IEEE; 2010, p. 7272–7.
[24] Giselsson P, Doan M, Keviczky T, De Schutter B, Rantzer A. Accelerated gradient methods and dual decomposition in distributed model predictive control. Automatica 2013;49(3):829–33.

[25] Giselsson P, Rantzer A. On feasibility, stability and performance in distributed model predictive control. IEEE Trans Autom Control 2013;59(4):1031–6.
[26] Köhler J, Müller MA, Allgöwer F. Distributed model predictive control—Recursive feasibility under inexact dual optimization. Automatica 2019;102:1–9.
[27] Doan M, Keviczky T, De Schutter B. A distributed optimization-based approach for hierarchical MPC of large-scale systems with coupled dynamics and constraints. In: 50th IEEE conference on decision and control (CDC) and European control conference. ECC, IEEE; 2011, p. 5236–41.
[28] Biegel B, Stoustrup J, Bendtsen J, Andersen P. Model predictive control for power flows in networks with limited capacity. In: 2012 American control conference. ACC, IEEE; 2012, p. 2959–64.
[29] Biegel B, Andersen P, Stoustrup J, Bendtsen J. Congestion management in a smart grid via shadow prices. IFAC Proc Vol 2012;45(21):518–23.
[30] Biegel B, Stoustrup J, Andersen P. Distributed MPC via dual decomposition. In: Distributed model predictive control made easy. Springer; 2014, p. 179–92.
[31] Pflaum P, Alamir M, Lamoudi M. Comparison of a primal and a dual decomposition for distributed MPC in smart districts. In: IEEE international conference on smart grid communications (smartGridComm). IEEE; 2014, p. 55–60.
[32] Pflaum P, Alamir M, Lamoudi M. Scalability study for a hierarchical NMPC scheme for resource sharing problems. In: IEEE European control conference. ECC, IEEE; 2015, p. 1468–73. http://dx.doi.org/10.1109/ECC.2015.7330746.
[33] Razzanelli M, Crisostomi E, Pallottino L, Pannocchia G. Distributed model predictive control for energy management in a network of microgrids using the dual decomposition method. Optim Control Appl Methods 2020;41(1):25–41.
[34] Eser S, Stoffel P, Kümpel A, Müller D. Distributed model predictive control of a nonlinear building energy system using consensus ADMM. In: 30th mediterranean conference on control and automation. MED, IEEE; 2022, p. 902–7.
[35] Maxeiner L, Engell S. Hierarchical MPC of batch reactors with shared resources. IFAC-PapersOnLine 2017;50(1):12041–6.
[36] Houska B, Shi J. Distributed MPC with ALADIN–a tutorial. 2022, arXiv preprint arXiv:2204.01654.
[37] Yfantis V, Gafur N, Wagner A, Ruskowski M. Hierarchical distributed model predictive control based on dual decomposition and quadratic approximation. In: 30th mediterranean conference on control and automation. MED, IEEE; 2022, p. 914–9.
[38] Halvgaard R, Vandenberghe L, Poulsen N, Madsen H, Jørgensen J. Distributed model predictive control for smart energy systems. IEEE Trans Smart Grid 2016;7(3):1675–82.
[39] Maestre J, Negenborn R. Distributed model predictive control made easy. Springer; 2014.
[40] Nocedal J, Wright S. Numerical optimization. Springer Science & Business Media; 2006.
[41] Shor N, Kiwiel K, Ruszcayǹski A. Minimization methods for non-differentiable functions. Springer-Verlag; 1985.
[42] Mäkelä M. Survey of bundle methods for nonsmooth optimization. Optim Methods Softw 2002;17(1):1–29. http://dx.doi.org/10.1080/10556780290027828.
[43] Maxeiner L, Engell S. An accelerated dual method based on analytical extrapolation for distributed quadratic optimization of large-scale production complexes. Comput Chem Eng 2020;135(1):106728. http://dx.doi.org/10.1016/j.compchemeng.2020.106728.
[44] Boyd S, Parikh N, Chu E, Peleato B, Eckstein J. Distributed optimization and statistical learning via the alternating direction method of multipliers. Found Trends Mach Learn 2011;3(1):1–122. http://dx.doi.org/10.1561/2200000016, URL https://www.nowpublishers.com/article/Details/MAL-016.
[45] He B, Yang H, Wang S. Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities. J Optim Theory Appl 2000;106(2):337–56. http://dx.doi.org/10.1023/A:1004603514434.
[46] Wang S, Liao L. Decomposition method with a variable parameter for a class of monotone variational inequality problems. J Optim Theory Appl 2001;109(2):415–29. http://dx.doi.org/10.1023/A:1017522623963.
[47] Maxeiner L. Dual-based methods for distributed optimization of interconnected systems. Shaker Verlag; 2021.
[48] Lunze J. Regelungstechnik 2: Mehrgrößensysteme, digitale regelung. Springer-Verlag; 2014.
[49] Bezanson J, Edelman A, Karpinski S, Shah V. Julia: A fresh approach to numerical computing. SIAM Rev 2017;59(1):65–98. http://dx.doi.org/10.1137/141000671.
[50] Lubin M, Dowson O, Garcia JD, Huchette J, Legat B, Vielma JP. JuMP 1.0: recent improvements to a modeling language for mathematical optimization. Math Program Comput 2023;1–9.
[51] Gurobi Optimization L. Gurobi Optimizer Reference Manual. 2023, URL https://www.gurobi.com.
[52] Wächter A, Biegler L. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math Program 2006;106(1):25–57. http://dx.doi.org/10.1007/s10107-004-0559-y.
[53] Vujanic R, Esfahani P, Goulart P, Mariéthoz S, Morari M. A decomposition method for large scale MILPs, with performance guarantees and a power system application. Automatica 2016;67:144–56.
[54] Wenzel S, Riedl F, Engell S. An efficient hierarchical market-like coordination algorithm for coupled production systems based on quadratic approximation. Comput Chem Eng 2020;134(8):106704. http://dx.doi.org/10.1016/j.compchemeng.2019.106704.