

# Distributed Model Predictive Control via Separable Optimization in Multiagent Networks

Ola Shorinwa , Graduate Student Member, IEEE, and Mac Schwager , Member, IEEE

**Abstract**—We present a distributed model predictive control method, which enables a group of agents to compute their control inputs locally while communicating with their neighbors over a communication network. While many distributed model predictive control methods require a central station for some coordination or computation of the optimization variables, our method does not require a central station, making our approach applicable to a variety of communication network topologies. With our method, each agent solves for its control inputs without solving for the control inputs of other agents, allowing for efficient optimization by each agent, unlike some other distributed methods. Furthermore, our method attains the linear convergence to the optimal control inputs in convex model predictive control problems, improving upon the sublinear convergence rates provided by some other distributed methods such as dual decomposition methods. Moreover, our algorithm provides a closed-loop controller for convex model predictive control problems with affine constraints. We demonstrate our method in both convex and nonconvex model predictive control problems in wireless transceiver alignment and satellite deployment, where we show the robustness of our method to time delays.

**Index Terms**—Distributed optimization, multiagent networks, optimal control, separable optimization, distributed model predictive control (MPC).

## I. INTRODUCTION

THE practical effectiveness of model predictive control (MPC) as a means for optimizing online control actions has spurred its application in a variety of problems, such as power management and chemical process control [1], [2], [3], [4], [5], climate regulation in buildings [6], [7], [8], [9], [10], [11], [12], [13], stabilization of vehicle platoons [14], [15], [16],

[17], [18], network scheduling and bandwidth allocation [19], [20], [21], and trajectory optimization in robotics [22], [23], [24], [25], [26]. In all these problems, the control inputs applied by each agent influence other agents, coupling the performance of all agents through the objective and constraint functions or dynamics models in the problem. As such, achieving the desired overall performance requires coordination among all agents. Centralized methods allow for the computation of the control inputs of each agent while considering the existing coupling between the agents; however, these methods require all agents to communicate with a central station, which might not be possible in many problems. Moreover, the computation of all the control inputs centrally can be particularly difficult in problems with a large number of agents. Distributed MPC methods attempt to resolve these issues by enabling each agent to compute its control inputs locally while communicating with its neighbors over a communication network.

Nonetheless, many distributed MPC methods still require a central station for the coordination of the agents or computation of some auxiliary variables [27], [28], [29], [30], [31], imposing restrictive constraints on the topology of the communication network between agents. In other distributed MPC methods, each agent solves the global optimization problem for its control inputs [32], [33], requiring each agent to know the objective, dynamics, and constraints functions of all agents, information unavailable to each agent in many situations. Eliminating these requirements, we present a distributed MPC method where each agent computes its control inputs locally, in parallel with other agents, allowing for efficient optimization, especially in problems that require high-frequency control updates.

We decompose the MPC problem to allow for distributed computation of the control inputs locally by each agent, following a variant of the alternating direction method of multipliers (ADMM) in [34]. In our algorithm, each agent collaborates with those agents that influence its performance, as defined by its objective and constraint functions, communicating with these agents over a communication network to compute its local control inputs. Our algorithm exploits the low-dimensionality of the coupling constraints between agents to enable each agent to compute its control inputs locally from a smaller MPC problem, involving only its objective functions, dynamics, and constraint functions, improving the efficiency of the computation and communication procedures associated with distributed MPC. We prove the linear convergence of the control inputs computed

Manuscript received 26 May 2022; revised 15 December 2022; accepted 8 April 2023. Date of publication 21 April 2023; date of current version 29 December 2023. This work was supported in part by ONR under Grant N00014-18-1-2830 and in part by NSF NRI under Award 1830402 and Award 1925030. Recommended by Associate Editor M. Lazar. (Corresponding author: Ola Shorinwa.)

Ola Shorinwa is with the Department of Mechanical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: shorinwa@stanford.edu).

Mac Schwager is with the Department of Aeronautics and Astronautics Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: schwager@stanford.edu).

Color versions of one or more figures in this article are available at <https://doi.org/10.1109/TAC.2023.3269338>.

Digital Object Identifier 10.1109/TAC.2023.3269338

by each agent to the optimal control inputs for convex MPC problems, with strongly convex objective functions. As a result, our method provides faster convergence rates, which improve upon the sublinear convergence rates provided by other distributed MPC methods.

Many distributed MPC methods only consider affine dynamics models for each agent along with convex state and input constraints, with the exception of a few [35], [36]. In contrast, our method applies to nonconvex MPC problems, with nonconvex objective and constraint functions. As such, we demonstrate our algorithm in MPC of satellites, considering the nonconvex dynamics models and problem constraints, representative of other nonconvex MPC problems. In addition, we apply our algorithm to a wireless transceiver alignment problem using convex MPC, where we demonstrate the robustness of our method to time delays. Empirically, our method converges about two times faster in the convex problem and about nine times faster in the nonconvex problem compared to other distributed MPC methods.

## I. Contributions

Our contributions are as follows.

- 1) We present an algorithm for separable optimization of distributed MPC problems (SOD-MPC), where each agent solves a smaller optimization problem locally to compute its control inputs without relying on a central station for any coordination or computation, in contrast with some other distributed methods.
- 2) With our algorithm, each agent does not compute the control inputs of other agents irrelevant to its performance and does not share its objective functions, dynamics models, and constraint functions with other agents, preserving the privacy of each agent.
- 3) In addition, our algorithm produces a closed-loop controller for convex MPC problems with affine constraints, providing the robustness to modeling errors and time delays arising from computation and communication between agents.

## II. RELATED WORKS

Lyapunov-based distributed control methods enable agents to solve MPC problems locally in a sequential order, with each agent receiving the control inputs of all preceding agents [27], [36], [37], [38], [39]. Upon the computation of its control inputs, each agent communicates its control inputs as well as the control inputs of all preceding agents to the following agent in the sequence. By not allowing for parallel computation of the control inputs by each agent, these methods can be ineffective in problems with a large number of agents, as each agent waits for all preceding agents to compute their control inputs before computing its control inputs. Moreover, the communication complexity of these methods grows linearly with the number of agents, further limiting the scalability of these approaches. These methods assume the existence of a Lyapunov-based controller which might not be readily available in many situations.

In other distributed approaches [32], [33], [35], [40], each agent optimizes over the global optimization problem to compute its control inputs in parallel with other agents while constraining the control inputs of other agents to the values received from these agents at the beginning of each iteration. Consequently, these methods require each agent to communicate with all other agents in the problem, imposing a stringent requirement on the topology of the communication network. Furthermore, these methods require each agent to have knowledge of the dynamics models, constraint functions, and objective functions of all agents, information not necessarily available to all agents in many problems.

Other approaches employ a distributed hierarchical control scheme where each agent shares its reference trajectory with all its neighbors and solves a local MPC problem to compute its control inputs [28], [41]. These approaches combine a stabilizing state-feedback controller, computed centrally, with the nominal control inputs, which can be computed using tube-based methods [42], during the online execution of the distributed MPC scheme. Centralized computation of a state-feedback controller along with feasible state and control sets can prove challenging, making these methods unsuitable for many problems. In addition, these approaches assume that the actual trajectories of all agents remain within a bounded region of the reference trajectory, a limiting assumption that is violated when agents deviate beyond the bounds of the reference trajectory to satisfy feasibility constraints in their local problems.

Dual decomposition methods enable each agent to compute its control inputs without knowledge of the local objective functions and constraints of other agents by solving the dual problem [43], [44], [45], [46], obtained from the Lagrangian of the original optimization problem, (refer to the work in [47] for a survey of dual decomposition methods). At each iteration of these methods, each agent computes a dual solution from which the agent can compute its control inputs. However, the feasibility of the resulting control inputs only occurs after the convergence of the dual variables to the optimal dual solution. Consequently, the iterations within dual decomposition methods cannot be terminated at any time before convergence to maintain feasibility, even in cases where the time utilized for the computation exceeds the agent's control intervals. Moreover, dual decomposition methods exhibit sublinear convergence at  $O(1/k)$  and, thus, require a significant number of iterations to converge in many problems. To improve its convergence rate, some dual approaches employ accelerated gradient methods in updating the dual variables [48] while others utilize a primal-dual active-set method [49]. In some other methods, each agent computes its control inputs from the primal problem using a distributed barrier method [50], [51], which ensures the feasibility of the computed control inputs.

The ADMM improves upon dual decomposition methods, extending its convergence properties to a broader class of problems while retaining its distributed properties. Some distributed MPC methods apply ADMM to the dual problem, where each agent keeps a local copy of the dual variable with a consensus constraint on these variables [52], [53]. These methods suffer the same challenges with dual decomposition methods with respect

to the feasibility of the resulting control inputs and a sublinear convergence rate of  $O(1/k)$ . Other methods apply ADMM to the primal problem, overcoming these challenges, but require a central station for updating the dual variables or for termination of the iterations [29], [30], [31], [54], [55], [56], making them unsuitable for many problems. To eschew central computation, some primal methods require each agent to compute the control inputs and states of all agents, resulting in local optimization problems with a significantly greater number of optimization variables, which require notable computational effort to solve. In addition, all agents share their solutions with their neighbors, further degrading the efficiency of these methods.

Our distributed MPC method overcomes these challenges, enabling each agent to compute its control inputs efficiently, without having to compute the control inputs of other agents unnecessarily. Furthermore, our algorithm applies to problems with a variety of communication network topologies and scales to problems with a large number of agents.

The article is organized as follows: In Section IV, we present the MPC problem along with the dynamics models, objective functions, and problem constraints between agents. In Section V, we derive SOD-MPC, a distributed method for MPC problems, and prove the linear convergence of our method to the optimal control inputs for strongly convex MPC problems. We derive a distributed closed-loop controller in Section VI and provide an efficient procedure for computing the optimal control inputs using the Riccati equations. We demonstrate superior convergence rates of our method compared to dual decomposition and ADMM-MPC methods in Section VII and examine the robustness of the closed-loop controller provided by our approach compared to other distributed approaches and an open-loop centralized controller. We conclude in Section VIII.

### III. PRELIMINARIES AND NOTATION

We provide the definition of the following mathematical concepts that will prove useful in the analysis of our algorithm.

*Definition 1 (Convex Set):* A set  $\mathcal{C}$  is convex if for all  $x, y \in \mathcal{C}$  and all  $\zeta \in [0, 1]$

$$\zeta x + (1 - \zeta)y \in \mathcal{C}. \quad (1)$$

*Definition 2 (Convex Function):* A function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is convex if its domain  $\text{dom}(f) \subseteq \mathbb{R}^n$  is convex, and for all  $x, y \in \text{dom}(f)$  and all  $\zeta \in [0, 1]$

$$f(\zeta x + (1 - \zeta)y) \leq \zeta f(x) + (1 - \zeta)f(y). \quad (2)$$

*Definition 3 (Strong Convexity):* A function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is  $m_f$ -strongly convex if and only if there exists  $m_f > 0$  such that

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x) + m_f \|y - x\|^2 \quad (3)$$

for all  $x, y \in \text{dom}(f)$  where  $\nabla f(x)$  denotes the gradient of  $f$  evaluated at  $x$ .

*Definition 4 (Lipschitz Continuity):* A function  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$  is Lipschitz continuous at  $x \in \text{dom}(f)$  if there exists a constant  $L_f > 0$  such that for all  $y \in \text{dom}(f)$  sufficiently close to  $x$

$$\|f(y) - f(x)\|_p \leq L_f \|y - x\|_p \quad (4)$$

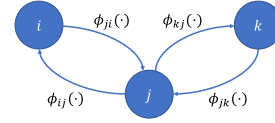


Fig. 1. Graphical representation of the coupling constraints  $\phi$  between agents in the MPC problem. The vertices represent the agents while edges between agents signify a coupling constraint between these agents.

where  $\|\cdot\|_p$  denotes the  $p$ -norm. When the inequality in (4) holds for all  $x, y \in \text{dom}(f)$ ,  $f$  is a Lipschitz function. In this work, we consider the  $\ell_2$ -norm.

*Definition 5 (Q-Linear Convergence):* A sequence  $\{z^k\}$  converges to a stationary point  $z^*$  Q-linearly if there exists  $\delta \in (0, 1)$  such that

$$\frac{\|z^{k+1} - z^*\|}{\|z^k - z^*\|} \leq \delta \quad (5)$$

for all  $k$  sufficiently large.

*Definition 6 (R-Linear Convergence):* A sequence  $\{z^k\}$  converges to a stationary point  $z^*$  R-linearly if there exists another sequence  $\{b^k\}$  such that

$$\|z^{k+1} - z^*\| \leq b^k \quad (6)$$

$\{b^k\}$  converges Q-linearly to zero, and  $b^k \geq 0$  for all  $k$ .

We represent the maximum singular value of a matrix  $A$  as  $\sigma_{\max}(A)$  and its minimum nonzero singular value as  $\sigma_{\min}(A)$ .

## IV. PROBLEM FORMULATION

We consider a general MPC problem among  $N$  agents where the performance of each agent influences the performance of other agents through coupling constraint functions, denoted by  $\phi_{ij}(\cdot)$  for a coupling constraint between agents  $i$  and  $j$ . We represent the coupling between agents graphically, as discussed in the following section.

### Coupling Graph

We represent the coupling between agents using a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  with a set of vertices  $\mathcal{V}$  denoting the agents and a set of edges  $\mathcal{E}$  denoting the coupling between agents; hence, if agent  $i$  influences the control inputs of agent  $j$ , then edge  $(i, j)$  exists in  $\mathcal{E}$ . We assume that an agent can communicate with another agent over a communication network in a timely fashion if its states and control inputs influence the control inputs of the other agent, to ensure the feasibility of the control inputs computed by each agent. We describe the neighbor set of agent  $i$  as  $\mathcal{N}_i$ , consisting of all agents that can communicate with agent  $i$ .

Fig. 1 depicts the graphical representation of the coupling constraints between agents.

*Remark 1 (Dynamic Graphs):* Our method allows for dynamic coupling graphs, which can change from time step to time step of the MPC problem; however, we assume the coupling graph remains static within a single time step during which the MPC optimization problem is solved. Empirically, we see solve

times on the order of tens of milliseconds, which is typically faster than the timescale of the dynamics, giving a justification for this assumption.

Coupling between agents in the MPC problem could arise through the dynamics model of each agent. We define  $\mathcal{N}_i^d \subseteq \mathcal{N}_i$  as the set of agents that influence the dynamics of agent  $i$ . We denote the state of agent  $i$  at time  $t$  as  $\hat{x}_{i,t} \in \mathbb{R}^{\hat{n}_i}$ , with its control inputs denoted by  $\hat{u}_{i,t} \in \mathbb{R}^{\hat{m}_i}$ . We represent the dynamics model of agent  $i$  by

$$\hat{x}_{i,t+1} = \mathcal{M}_{i,t}(\hat{x}_{i,t}, X_{i,t}, \hat{u}_{i,t}, U_{i,t}) \quad (7)$$

where  $X_{i,t} = [\hat{x}_{r,t}^\top, \forall r \in \mathcal{N}_i^d]^\top$  denotes the concatenation of the states of all agents, which influence the dynamics of agent  $i$ , with  $U_{i,t}$  denoting the concatenation of the control inputs of these agents.

For distributed optimization, agent  $i$  keeps a local copy of the variables  $X_{i,t}$  and  $U_{i,t}$ ,  $\forall t$ , with an associated set of equality constraints between corresponding variables computed by agent  $i$  and each agent in  $\mathcal{N}_i^d$ . We denote agent  $i$ 's local copies of agent  $j$ 's state and control inputs as  $\hat{x}_{j,t}^i$  and  $\hat{u}_{j,t}^i$ , respectively, and its local copies of the concatenation of the states and control inputs of all agents in  $\mathcal{N}_i^d$  as  $X_{i,t}^c$  and  $U_{i,t}^c$ , respectively. With this procedure, the dynamics model of agent  $i$  depends on its states and control inputs and its local variables  $X_{i,t}^c$  and  $U_{i,t}^c$ . Furthermore, the dynamics model of agent  $i$  induces the equality constraints given by

$$\hat{x}_{j,t}^i = \hat{x}_{j,t}, \quad \hat{u}_{j,t}^i = \hat{u}_{j,t} \quad \forall t \quad \forall j \in \mathcal{N}_i^d \quad (8)$$

which are incorporated into the corresponding coupling constraint functions between agents  $i$  and  $j$ , given by  $\phi_{ij}(\cdot)$ ,  $\forall j \in \mathcal{N}_i^d$ .

We represent the concatenation of the state of agent  $i$  and  $X_{i,t}^c$  at time  $t$  as  $x_{i,t} = [\hat{x}_{i,t}^\top, X_{i,t}^{c\top}]^\top$  and the concatenation of its control inputs and  $U_{i,t}^c$  as  $u_{i,t} = [\hat{u}_{i,t}^\top, U_{i,t}^{c\top}]^\top$ . In addition, we represent the optimization variables of agent  $i$  as  $x_i = [x_{i,0}^\top, \dots, x_{i,\mathcal{P}}^\top]^\top$  and  $u_i = [u_{i,0}^\top, \dots, u_{i,\mathcal{P}}^\top]^\top$ , concatenating its states and control inputs over a problem horizon  $\mathcal{P}$  on each instance of the MPC problem. We collect the dynamics constraints across all time indices into a single constraint function  $d_i(x_i, u_i) = 0$ .

Over the graph  $\mathcal{G}$ , we describe the MPC problem by

$$\begin{aligned} & \underset{x, u}{\text{minimize}} \quad \sum_{i=1}^N f_i(x_i, u_i) \\ & \text{subject to} \quad d_i(x_i, u_i) = 0, \quad i = 1, \dots, N \\ & \quad g_i(x_i, u_i) = 0, \quad i = 1, \dots, N \\ & \quad h_i(x_i, u_i) \leq 0, \quad i = 1, \dots, N \\ & \quad \phi_{ij}(x_i, u_i, x_j, u_j) = 0 \quad \forall (i, j) \in \mathcal{E} \end{aligned} \quad (9)$$

where  $x_i \in \mathbb{R}^{n_i}$ ,  $u_i \in \mathbb{R}^{m_i}$ , and  $f_i : \mathbb{R}^{n_i} \times \mathbb{R}^{m_i} \rightarrow \mathbb{R}$  represents the local objective function of agent  $i$ . We assume the objective function  $f_i$  is lower-bounded,  $\forall i$ . We represent other equality constraints in the MPC problem by  $g_i$ , including constraints on its initial states. Likewise, we include inequality constraints on the optimization variables of agent  $i$  in  $h_i$ , such

as constraints on its feasible states and its control limits. The optimization variables of agent  $i$  depend on the optimization variables of agent  $j$  through the coupling constraint function  $\phi_{ij}(\cdot)$ . The coupling constraints between agents can vary over time; however, we do not indicate its dependence on time explicitly. We represent the concatenation of the states of all agents as  $x \in \mathbb{R}^{n_x}$  and their control inputs as  $u \in \mathbb{R}^{n_u}$ . For simplicity of exposition, we consider equality constraints for  $\phi_{ij}(\cdot)$ ; however, our method readily applies to inequality constraints between agents. We note that inequality-constrained problems can be reformulated using auxiliary variables into the same form as (9).

For an MPC problem with a problem horizon  $\mathcal{P}$ , we define the objective function as

$$f_i(x_i, u_i) = \mathcal{W}_{i,\mathcal{P}}(x_i, u_i) + \sum_{\tau=0}^{\mathcal{P}-1} \mathcal{W}_{i,\tau}(x_{i,\tau}, u_{i,\tau}) \quad (10)$$

where  $\mathcal{W}_{i,\tau} : \mathbb{R}^{n_i} \times \mathbb{R}^{m_i} \rightarrow \mathbb{R}$  represents the per-stage objective function at time  $\tau$ . In many MPC problems,  $\mathcal{W}_{i,\tau}(\cdot)$  takes a quadratic form given by

$$\begin{aligned} \mathcal{W}_{i,\tau}(x_{i,\tau}, u_{i,\tau}) &= (x_{i,\tau} - \tilde{x}_{i,\tau})^\top \mathcal{Q}_{i,\tau} (x_{i,\tau} - \tilde{x}_{i,\tau}) \\ &\quad + u_{i,\tau}^\top \mathcal{R}_{i,\tau} u_{i,\tau} \end{aligned} \quad (11)$$

where  $\tilde{x}_{i,\tau}$  denotes a desired reference state trajectory,  $\mathcal{Q}_{i,\tau} \in \mathbb{R}^{n_i} \times \mathbb{R}^{n_i}$  represents a positive semidefinite weight matrix, and  $\mathcal{R}_{i,\tau} \in \mathbb{R}^{m_i} \times \mathbb{R}^{m_i}$  represents a positive-definite weight matrix. The quadratic objective function in (11) penalizes deviations from the desired state trajectory  $\tilde{x}_{i,\tau}$  while minimizing energy consumption. We assume that the terminal cost and terminal set are selected such that stability and recursive feasibility of the MPC problem are guaranteed [40], [57], [58]. This procedure may involve constraint tightening of the terminal set or relaxation of the dynamics constraints in problems with terminal set constraints [59], [60]. In problems without terminal set constraints, recursive feasibility and stability can be guaranteed if the problem horizon and terminal cost are sufficiently large [61]. We note a terminal set can be selected to guarantee the stability of the MPC problem even in situations where a suboptimal solution is computed, which is particularly relevant in nonconvex problems [62], [63], [64]. Moreover, to preserve the separability of the objective and constraint functions in (9), we can synthesize a separable terminal cost function, along with local terminal set constraints, to guarantee the stability of the MPC problem, as described in [46].

## V. DISTRIBUTED CONTROL

Noting that centralized approaches for solving the MPC problem (9) require the computation of the control inputs of all agents at a central station, which can be impractical in many situations; we derive an algorithm for distributed MPC, SOD-MPC, which enables each agent to compute its local states and control inputs without computing the states and control inputs of other agents irrelevant to its performance.

We note that some other consensus ADMM methods [44], [65], [66] require the introduction of local copies of all the optimization variables arising in the coupling constraint functions in (9), followed by a subsequent transformation of the coupling constraints to local constraints enforced by each agent. Moreover, the coupling constraints in (9) are replaced by consensus constraints between the original optimization variables and the new local copies. This approach results in greater computation and communication overhead, considering the increase in the number of the optimization variables in the problem. As a result, we derive a distributed algorithm that avoids this approach.

We apply the SOVA method [34] based on ADMM to obtain distributed update procedures for the local variables of each agent. We express the coupling constraint function between agents  $i$  and  $j$  as

$$\phi_{ij}(x_i, u_i, x_j, u_j) = \phi_{ij}^i(x_i, u_i) - \phi_{ij}^j(x_j, u_j) \quad (12)$$

where  $\phi_{ij}^i : \mathbb{R}^{n_i} \times \mathbb{R}^{m_i} \rightarrow \mathbb{R}^{n_{ij}}$  depends only on the optimization variables of agent  $i$  and  $\phi_{ij}^j : \mathbb{R}^{n_j} \times \mathbb{R}^{m_j} \rightarrow \mathbb{R}^{n_{ij}}$  depends only on the optimization variables of agent  $j$ . We emphasize that the decomposition in (12) exists for any arbitrary coupling constraint function  $\phi_{ij}$ , which we demonstrate in the subsequent discussion, where we consider two broad classes of MPC problems.

### V. Convex MPC Problems

In convex MPC problems, the coupling constraint functions must be affine, which shows that the optimization variables of agents  $i$  and  $j$  do not appear together in the same term within the coupling constraint function. Consequently, we can group all terms depending on the optimization variables of agent  $i$  into a new function  $\phi_{ij}^i(\cdot)$  and, likewise, group all terms depending on the optimization variables of agent  $j$  into  $\phi_{ij}^j(\cdot)$ , such that we obtain the expression in (12). We can assign the constant term in the coupling constraint function to either  $\phi_{ij}^i(\cdot)$  or  $\phi_{ij}^j(\cdot)$  or split it between both functions.

### V. Nonconvex MPC Problems

The coupling constraints in some nonconvex MPC problems possess a separable structure naturally. For example, coupling constraints of the form

$$\phi_{ij}(u_i, u_j) = \|u_i\|_2^2 - \|u_j\|_2^2 \quad (13)$$

can be simply expressed in the form given in (12) with

$$\phi_{ij}^i(u_i) = \|u_i\|_2^2 \quad \text{and} \quad \phi_{ij}^j(u_j) = \|u_j\|_2^2. \quad (14)$$

We consider more general coupling constraint functions that may not possess a separable structure, e.g., constraint functions with bilinear terms. Many decomposition strategies exist for expressing these constraint functions in the form given in (12). Here, we discuss the simplest decomposition strategy, which involves introducing new optimization variables representing local copies of the original optimization variables in the original coupling constraint. Subsequently, we transform the coupling constraint to a local constraint, enforced by each agent locally.

Finally, we create a new coupling constraint enforcing equality between the original optimization variables and the local copies of these variables, which were introduced to decouple the original coupling constraint. The new coupling constraint exists in the form given in (12).

To provide a concrete illustration, we consider the bilinear constraint  $x_i^\top x_j = 0$ . We begin by introducing a local copy of  $x_i$ , denoted by  $\check{x}_i$ , and a local copy of  $x_j$ , denoted by  $\check{x}_j$ . Next, we assign  $\check{x}_j$  to agent  $i$  and  $\check{x}_i$  to agent  $j$ . Agent  $i$  enforces the local constraint

$$x_i^\top \check{x}_j = 0 \quad (15)$$

while agent  $j$  enforces the local constraint

$$\check{x}_i^\top x_j = 0. \quad (16)$$

Finally, we create a pair of new coupling constraints

$$x_i - \check{x}_i = 0 \quad \text{and} \quad x_j - \check{x}_j = 0 \quad (17)$$

which are in the form given in (12). We note that more efficient decomposition strategies exist; however, we do not discuss these strategies here, due to space constraints.

Expressing the coupling constraints in the form in (12) enables us to decouple the contribution of each agent to the value of the coupling constraint, allowing each agent to optimize over this constraint locally.

With the constraint in (12), the MPC problem in (9) is given by

$$\begin{aligned} & \underset{\mathbf{x}, \mathbf{u}}{\text{minimize}} \quad \sum_{i=1}^N f_i(x_i, u_i) \\ & \text{subject to} \quad d_i(x_i, u_i) = 0, \quad i = 1, \dots, N \\ & \quad g_i(x_i, u_i) = 0, \quad i = 1, \dots, N \\ & \quad h_i(x_i, u_i) \leq 0, \quad i = 1, \dots, N \\ & \quad \phi_{ij}^i(x_i, u_i) = \phi_{ij}^j(x_j, u_j) \quad \forall (i, j) \in \mathcal{E} \end{aligned} \quad (18)$$

where agent  $i$  computes its local optimization variables  $x_i$  and  $u_i$ . We introduce the auxiliary variables  $c_{ij} \in \mathbb{R}^{n_{ij}}$  in the coupling constraint  $\phi_{ij}(\cdot)$ , expressing the MPC problem in (18) as

$$\begin{aligned} & \underset{\mathbf{x}, \mathbf{u}, \mathbf{c}}{\text{minimize}} \quad \sum_{i=1}^N f_i(x_i, u_i) \\ & \text{subject to} \quad d_i(x_i, u_i) = 0, \quad i = 1, \dots, N \\ & \quad g_i(x_i, u_i) = 0, \quad i = 1, \dots, N \\ & \quad h_i(x_i, u_i) \leq 0, \quad i = 1, \dots, N \\ & \quad \phi_{ij}^i(x_i, u_i) = c_{ij} \quad \forall (i, j) \in \mathcal{E} \\ & \quad \phi_{ij}^j(x_j, u_j) = c_{ij} \quad \forall (i, j) \in \mathcal{E} \end{aligned} \quad (19)$$

where  $\mathbf{c} = [c_{ij}^\top, \forall (i, j) \in \mathcal{E}]^\top$ .

Next, we derive update procedures for the states and control inputs of each agent from the augmented Lagrangian for the

problem in (19)

$$\begin{aligned} \mathcal{L}_a(\mathbf{x}, \mathbf{u}, \mathbf{c}, \mathbf{v}, \mathbf{w}) = & \sum_{i=1}^N f_i(x_i, u_i) \\ & + \sum_{(i,j) \in \mathcal{E}} \left( v_{ij}^\top (\phi_{ij}^i(x_i, u_i) - c_{ij}) \right. \\ & \quad \left. + w_{ij}^\top (\phi_{ij}^j(x_j, u_j) - c_{ij}) \right) \\ & + \frac{\rho}{2} \sum_{(i,j) \in \mathcal{E}} \left( \|\phi_{ij}^i(x_i, u_i) - c_{ij}\|_2^2 \right. \\ & \quad \left. + \|\phi_{ij}^j(x_j, u_j) - c_{ij}\|_2^2 \right) \end{aligned} \quad (20)$$

with dual variables  $v_{ij} \in \mathbb{R}^{n_{ij}}$  and  $w_{ij} \in \mathbb{R}^{n_{ij}}$  for the coupling constraints between agents  $i$  and  $j$ , where  $\mathbf{v} = [v_{ij}^\top, \forall (i,j) \in \mathcal{E}]^\top$  and  $\mathbf{w} = [w_{ij}^\top, \forall (i,j) \in \mathcal{E}]^\top$ . We have not dualized the dynamics constraints and the local equality and inequality constraints in (20). Rather, we restrict the domain of  $\mathcal{L}_a(\cdot)$  such that

$$\begin{aligned} d_i(x_i, u_i) &= 0, \quad i = 1, \dots, N \\ g_i(x_i, u_i) &= 0, \quad i = 1, \dots, N \\ h_i(x_i, u_i) &\leq 0, \quad i = 1, \dots, N \end{aligned} \quad (21)$$

which ensures that the control inputs generated by our algorithm remain feasible for each agent.

The positive parameter  $\rho$  in the augmented Lagrangian determines the penalty on the violation of the coupling constraints between the agents. Each agent updates its primal variables iteratively as the minimizers of the augmented Lagrangian using its dual variables at the previous iteration and subsequently updates its dual variables through dual ascent on the augmented Lagrangian using its computed primal variables. We update the primal variables  $\mathbf{x}$  and  $\mathbf{u}$  before updating the auxiliary variable  $\mathbf{c}$ .

For any value of  $\rho > 0$ , the augmented Lagrangian is quadratic in  $c_{ij}$ . As a result, the update procedure for  $c_{ij}$  yields a closed-form solution, given by

$$c_{ij}^{k+1} = \frac{v_{ij}^k + w_{ij}^k}{2\rho} + \frac{\phi_{ij}^i(x_i^{k+1}, u_i^{k+1}) + \phi_{ij}^j(x_j^{k+1}, u_j^{k+1})}{2} \quad (22)$$

at iteration  $k$ . The dual-ascent procedure for updating the dual variables results in

$$v_{ij}^{k+1} = v_{ij}^k + \rho (\phi_{ij}^i(x_i^{k+1}, u_i^{k+1}) - c_{ij}^{k+1}) \quad (23)$$

and

$$w_{ij}^{k+1} = w_{ij}^k + \rho (\phi_{ij}^j(x_j^{k+1}, u_j^{k+1}) - c_{ij}^{k+1}). \quad (24)$$

By simplifying (23) and (24) using (22), we obtain that  $v_{ij}^{k+1} = -w_{ij}^{k+1}$ , at each iteration  $k$ .

Agent  $i$  updates  $x_i^k$  and  $u_i^k$  at iteration  $k$  as the minimizer of the problem

$$\begin{aligned} & \underset{x_i, u_i}{\text{minimize}} \quad f_i(x_i, u_i) + q_i^{k\top} \phi_i^i(x_i, u_i) \\ & \quad + \rho \sum_{j \in \mathcal{N}_i} \left\| \phi_{ij}^i(x_i, u_i) - \psi_{ij}(x_i^k, u_i^k, x_j^k, u_j^k) \right\|_2^2 \\ & \text{subject to} \quad d_i(x_i, u_i) = 0 \\ & \quad g_i(x_i, u_i) = 0 \\ & \quad h_i(x_i, u_i) \leq 0 \end{aligned} \quad (25)$$

where  $\phi_i(\cdot)$  represents the vertical concatenation of the coupling constraint functions between agent  $i$  and all its neighbors, given by

$$\begin{aligned} \phi_i(\cdot) &= [(\phi_{ij}^i(\cdot) - \phi_{ij}^j(\cdot))^\top \quad \forall j \in \mathcal{N}_i]^\top \\ \phi_i^i(\cdot) &= [\phi_{ij}^i(\cdot)^\top \quad \forall j \in \mathcal{N}_i]^\top \end{aligned} \quad (26)$$

and

$$\psi_{ij}(x_i^k, u_i^k, x_j^k, u_j^k) = \frac{\phi_{ij}^i(x_i^k, u_i^k) + \phi_{ij}^j(x_j^k, u_j^k)}{2}. \quad (27)$$

We have introduced a composite dual variable  $q_i \in \mathbb{R}^{n_c}$ , with the component of  $q_i^k$  corresponding to edge  $(i, j)$  given by

$$[q_i^k]_{ij} = v_{ij}^k + w_{ji}^k \quad (28)$$

where  $\mathbf{v}^0 = \mathbf{w}^0 = 0$ . From (23) and (24), we obtain the corresponding update procedure for  $q_i$  at iteration  $k$ , given by

$$q_i^{k+1} = q_i^k + \rho \phi_i(x_i^{k+1}, u_i^{k+1}, x_{\mathcal{N}_i}^{k+1}, u_{\mathcal{N}_i}^{k+1}) \quad (29)$$

where  $x_{\mathcal{N}_i}$  and  $u_{\mathcal{N}_i}$  denote the vertical concatenation of the optimization variables of all the neighbors of agent  $i$ . In deriving the update procedure in (25), we have assumed that agents  $i$  and  $j$  have access to  $c_{ij}^{k+1}$ , i.e., agent  $i$  has access to the value of  $\phi_{ij}^j(x_j^{k+1}, u_j^{k+1})$  and agent  $j$  has access to the value of  $\phi_{ij}^i(x_i^{k+1}, u_i^{k+1})$ , which requires that the communication graph within a single instance of the MPC problem remains static. This assumption enables us to reduce the number of optimization variables maintained by each robot, as each agent maintains a smaller composite dual variable, as opposed to maintaining a dual variable for each edge in  $\mathcal{G}$ . Note that this assumption does not restrict the communication graph from changing across different instances of the MPC problem. In addition, we note that the update procedures can be modified to allow for dynamic communication graphs within a single instance of the MPC problem; however, the resulting algorithm would require each agent to maintain a dual variable for each edge in  $\mathcal{G}$ .

Each agent does not need to compute the auxiliary variables as these variables have been included in the update procedures in (25). From the update equations (25) and (29), agent  $i$  only communicates with its neighbors to evaluate its coupling constraints  $\phi_i(\cdot)$  and  $\psi_{ij}(\cdot), \forall j \in \mathcal{N}_i$ . Specifically, agent  $i$  shares the value of  $\phi_{ij}^i(x_i^{k+1}, u_i^{k+1})$  with agent  $j$ , its neighbor,  $\forall j \in \mathcal{N}_i$ , at iteration  $k$ . As a result, SOD-MPC does not necessarily require each agent to share its iterates for the local optimization variables

---

**Algorithm 1:** Separable Optimization for Distributed MPC (SOD-MPC).

---

**do in parallel**  $i = 1, \dots, N$   
 |  $(x_i, u_i) \leftarrow \text{DistributedControl}(t, \bar{x}_{i,t})$   
 | Apply control input  $\tilde{u}_{i,t}$ .  
**while** *task is in progress*;

---

**Function**  $\text{DistributedControl}(t, \bar{x}_{i,t})$

---

Initialization:

$k \leftarrow 0$

$q_i^0 \leftarrow 0$

$(x_i^0, u_i^0) \leftarrow \underset{x_i, u_i}{\text{argmin}} f_i(x_i, u_i)$

**do in parallel**  $i = 1, \dots, N$

|  $(x_i^{k+1}, u_i^{k+1}) \leftarrow \text{Procedure (25)}$

Communication Step:

Agent  $i$  shares the value of  $\phi_{ij}^i(x_i^{k+1}, u_i^{k+1})$  with agent  $j$ , its neighbor, and receives the value of  $\phi_{ij}^j(x_j^{k+1}, u_j^{k+1})$ ,  $\forall j \in \mathcal{N}_i$ .

|  $q_i^{k+1} \leftarrow \text{Procedure (29)}$

|  $k \leftarrow k + 1$

**while** *not converged or stopping criterion is not met*;

**return**  $(x_i, u_i)$

---

$x$ ,  $u$ , and  $q$  with its neighbors. Rather, each agent shares the value of its component of the coupling constraints with its neighbors, ensuring the privacy of its local variables. In addition, each agent only requires its local objective and constraint functions in its update procedures, maintaining the privacy of these local functions and the problem data.

*Theorem 1:* For a nonconvex MPC problem, the iterates  $(x_i^k, u_i^k)$  of agent  $i$  converge to a locally optimal solution,  $\forall i \in \mathcal{V}$ , with the assumption that the objective function of (9) has Lipschitz continuous gradients.

*Proof:* We omit the proof here. Refer to the work in [67] for the proof.  $\square$

Algorithm 1 outlines SOD-MPC for distributed MPC.

At each iteration, the states and control inputs of agent  $i$ , contained in its local optimization variables  $x_i$  and  $u_i$  satisfy all the constraints in (9), except its coupling constraint, which is satisfied at convergence of our algorithm. Hence, the *DistributedControl* procedure can be terminated early to allow each agent to apply its control inputs, if absolute satisfaction of the coupling constraints is not required. Since the iterates of each agent satisfy its dynamics constraints, control inputs constraints, and terminal set constraints, we note that early termination of our algorithm does not negatively impact the stability guarantees of suboptimal MPC [62]. Moreover, our algorithm preserves the same stability guarantees in nonconvex MPC problems, where, generally, suboptimal solutions are computed. With SOD-MPC, agent  $i$  computes its control inputs  $\tilde{u}_{i,t}$  at its current state  $\bar{x}_{i,t}$  and repeats this procedure for its control inputs at the next time instant.

## A. Convergence Analysis for Convex MPC Problems

We examine the convergence rate of our algorithm on convex MPC problems. We note that the constraint functions  $d_i$ ,  $g_i$ , and  $\phi_{ij}$  are restricted to affine functions in convex model predictive problems. In addition, the function  $h_i$  must be convex. We express the optimization problem in (18) in terms of the control inputs by eliminating the state variables using the dynamics model in  $d_i(\cdot)$  for agent  $i$ . In addition, we eliminate constraints on the initial state of each agent, specified in  $g_i$ . With this formulation, the convex MPC problem is given by

$$\begin{aligned} & \underset{\mathbf{u}}{\text{minimize}} \sum_{i=1}^N \mathbb{F}_i(u_i) \\ & \text{subject to } \mathbb{G}_i(u_i) = 0, \quad i = 1, \dots, N \\ & \quad \mathbb{H}_i(u_i) \leq 0, \quad i = 1, \dots, N \\ & \quad \Phi_{ij}^i(u_i) = \Phi_{ij}^j(u_j) \quad \forall (i, j) \in \mathcal{E} \end{aligned} \quad (30)$$

where the constraint functions  $\mathbb{G}_i$  and  $\Phi_{ij}$  are affine and the objective function  $\mathbb{F}_i$  and constraint function  $\mathbb{H}_i$  are convex functions. We assume that the problem in (30) and the associated update procedures are feasible and, in addition, an optimal primal–dual solution of (30) exists.

For convex model predictive problems, described by (30), we provide a general convergence result in the following theorem.

*Theorem 2 (Convergence of  $\{\mathbf{u}^k\}$ ):* In convex MPC problems, the sequence of control inputs computed by all agents  $\{\mathbf{u}^k\}$  converges sublinearly to the optimal control inputs  $\mathbf{u}^*$ .

*Proof:* We refer readers to the work in [68] for the proof.  $\square$

SOD-MPC converges for any value of  $\rho > 0$ . Specific strategies for selecting the value of  $\rho$  can be found in [69] and [70]. We can make stronger statements on the convergence rate of our algorithm in convex MPC problems without the local constraint functions  $\mathbb{G}_i$  and  $\mathbb{H}_i$  in (30). Particularly, the iterates generated by our algorithm converge linearly to the optimal solution in these problems. To show the linear convergence of our algorithm, we assume that the local objective function  $\mathbb{F}_i$  of agent  $i$  is continuously differentiable with Lipschitz continuous gradients and strongly convex, for all agents. This assumption is not restrictive, considering that, in many MPC problems, the objective function takes a quadratic form with a positive-definite Hessian. In addition, we define the sequence  $\{z^k\}$  as

$$z^k = \begin{bmatrix} q^k \\ c^k \end{bmatrix}, \quad (31)$$

composed of the primal variable  $c$  and the dual variable  $q$ , where  $c = [c_{ij}^T, \forall (i, j) \in \mathcal{E}]^T$  and  $q = [q_i^T, \forall i \in \mathcal{V}]^T$ .

*Lemma 1 (Sufficient Descent):* The error  $\|z^k - z^*\|$  decreases monotonically at each iteration, until convergence of the sequence  $\{z^k\}$  to a limit point, with

$$\begin{aligned} & \|z^k - z^*\|_W^2 - \|z^{k+1} - z^*\|_W^2 \\ & \geq m_f \|\mathbf{u}^{k+1} - \mathbf{u}^*\|_2^2 + \|z^{k+1} - z^k\|_W^2 \end{aligned} \quad (32)$$

where  $z^*$  and  $u^*$  denote the optimal solution of  $z$  and the control inputs, respectively, and  $W > 0$ .

*Proof:* We provide the proof in the Appendix.  $\square$

**Lemma 2 (Q-Linear Convergence of  $\{z^k\}$ ):** The sequence  $\{z^k\}$  converges Q-linearly to  $z^*$  with

$$\frac{\|z^{k+1} - z^*\|_W^2}{\|z^k - z^*\|_W^2} \leq \frac{1}{\ell + 1} \quad (33)$$

where

$$\ell = \min \left\{ \frac{4m_f \rho (\mu - 1) \sigma_{\min}^2(\mathcal{J})}{\rho^2 (\mu - 1) \sigma_{\max}^2(\bar{\mathcal{J}}) \sigma_{\min}^2(\mathcal{J}) + 4\mu L_f^2}, \frac{\sigma_{\min}^2(\mathcal{J})}{\mu \sigma_{\min}^2(\bar{\mathcal{J}})} \right\} \quad (34)$$

with  $\ell > 0$  and  $\mu > 1$ .

*Proof:* We provide the proof in the Appendix.  $\square$

**Theorem 3 (R-Linear Convergence of  $\{u^k\}$ ):** The sequence of control inputs computed by all agents  $\{u^k\}$  converges R-linearly to the optimal control inputs  $u^*$  with

$$\|u^{k+1} - u^*\|_W^2 \leq \frac{1}{m_f} \|z^k - z^*\|_W^2 \quad (35)$$

where  $m_f > 0$ , for convex MPC problems with strongly convex objective functions and Lipschitz continuous gradients.

*Proof:* From Lemma 1 and Q-linear convergence of  $\{z^k\}$  (see Lemma 2)

$$\|u^{k+1} - u^*\|_2^2 \leq \frac{1}{m_f} \|z^k - z^*\|_W^2 \quad (36)$$

showing R-linear convergence of  $\{u^k\}$  to the optimal control inputs  $u^*$ .  $\square$

## VI. CLOSED-LOOP CONTROL LAW

SOD-MPC provides a time-varying closed-loop control law for convex MPC problems with affine constraints, from the update procedure in (25). We consider model predictive problems where the dynamics model of agent  $i$  is given by

$$x_{i,\tau+1} = A_{i,\tau} x_{i,\tau} + B_{i,\tau} u_{i,\tau} \quad (37)$$

with  $x_{i,\tau} \in \mathbb{R}^{n_i}$  denoting its state,  $u_{i,\tau} \in \mathbb{R}^{m_i}$  denoting its control inputs, and  $A_{i,\tau} \in \mathbb{R}^{n_i \times n_i}$  and  $B_{i,\tau} \in \mathbb{R}^{n_i \times m_i}$  describing its dynamics, at time  $\tau$ . In these problems, the agents compute their states and control inputs by solving the optimization problem

$$\begin{aligned} & \text{minimize}_{x,u} \sum_{i=1}^N f_i(x_i, u_i) \\ & \text{subject to } x_{i,0} = \bar{x}_{i,t}, \quad i = 1, \dots, N \\ & x_{i,\tau+1} = A_{i,\tau} x_{i,\tau} + B_{i,\tau} u_{i,\tau} \quad \forall \tau, i = 1, \dots, N \\ & \phi_{ij}(x_i, u_i, x_j, u_j) = 0 \quad \forall (i, j) \in \mathcal{E} \end{aligned} \quad (38)$$

where  $\bar{x}_{i,t}$  indicates the state of agent  $i$  at time  $t$  and  $f_i$  represents the objective function of agent  $i$  given by

$$f_i(x_i, u_i) = x_{i,p}^\top H_i x_{i,p} + \sum_{\tau=0}^{p-1} (x_{i,\tau}^\top Q_{i,\tau} x_{i,\tau} + u_{i,\tau}^\top R_{i,\tau} u_{i,\tau}) \quad (39)$$

with a terminal weight  $H_i \in \mathbb{R}^{n_i \times n_i}$ , weights on agent  $i$ 's states  $Q_i \in \mathbb{R}^{n_i \times n_i}$ , and weights on its control inputs  $R_i \in \mathbb{R}^{m_i \times m_i}$ . For the existence of a stabilizing solution to the MPC problem, we require positive-definite weights  $Q_i$  and positive semidefinite weights for  $R_i$ .

Using Algorithm 1, agent  $i$  computes its states and control inputs at time  $t$  from the optimization problem

$$\begin{aligned} & \text{minimize}_{x_i, u_i} x_{i,p}^\top H_i x_{i,p} + \sum_{\tau=0}^{p-1} (x_{i,\tau}^\top Q_{i,\tau} x_{i,\tau} + u_{i,\tau}^\top R_{i,\tau} u_{i,\tau}) \\ & + q_i^{k\top} \phi_i^i(x_i, u_i) \\ & + \rho \sum_{j \in \mathcal{N}_i} \left\| \phi_{ij}^i(x_i, u_i) - \psi_{ij}(x_i^k, u_i^k, x_j^k, u_j^k) \right\|_2^2 \\ & \text{subject to } x_{i,0} = \bar{x}_{i,t} \\ & x_{i,\tau+1} = A_{i,\tau} x_{i,\tau} + B_{i,\tau} u_{i,\tau} \quad \forall \tau \end{aligned} \quad (40)$$

with  $\psi_{ij}(\cdot)$  given by (27). A direct approach to computing agent  $i$ 's control inputs involves solving the quadratic program (40) numerically. Numerical quadratic optimization techniques involve factorizing the Hessian of the Lagrangian of (40), which can require notable computational effort, depending on the length of the problem horizon and the dimensions of the state and control inputs of each agent. Consequently, this approach quickly creates computation overhead, especially in problems with high control bandwidths.

In contrast, we derive a time-varying closed-loop control law for computing the optimal control inputs of (40), which involves factorizing a matrix of a much smaller dimension, significantly reducing the computation overhead associated with solving (40). For convex MPC problems, we eliminate the state variables of agent  $i$  in the coupling constraints  $\phi_i^i(\cdot)$  and express the resulting coupling constraints at time  $t$  as

$$\phi_i^i(u_i, t) = \nu_{i,t}^i u_i + \eta_{i,t}^i \quad (41)$$

where  $\nu_{i,t}^i$  represents the Jacobian of  $\phi_i^i(\cdot)$  with respect to  $u_i$  given by

$$\nu_{i,t}^i = \nabla_{u_i} \phi_i^i(u_i, t) \quad (42)$$

and  $\eta_{i,t}^i$  represents constant terms in  $\phi_i^i(\cdot)$ , independent of agent  $i$ 's control inputs.

From dynamic programming, agent  $i$  solves for its control input  $u_{i,p-1}$  from the optimization problem

$$\begin{aligned} & \text{minimize}_{x_{i,p}, u_{i,p-1}} x_{i,p}^\top H_i x_{i,p} + q_{i,p}^{k\top} \nu_{i,p}^i u_{i,p-1} \\ & + \rho \sum_{j \in \mathcal{N}_i} \left\| \nu_{ij,p}^i u_{i,p-1} - \gamma_{ij,p}(u_{i,p-1}^k, u_{j,p-1}^k) \right\|_2^2 \\ & + x_{i,p-1}^\top Q_{i,p-1} x_{i,p-1} \\ & + u_{i,p-1}^\top R_{i,p-1} u_{i,p-1} + q_{i,p-1}^{k\top} \nu_{i,p-1}^i u_{i,p-1} \\ & + \rho \sum_{j \in \mathcal{N}_i} \left\| \nu_{ij,p-1}^i u_{i,p-1} - \gamma_{ij,p-1}(u_{i,p-1}^k, u_{j,p-1}^k) \right\|_2^2 \\ & \text{subject to } x_{i,p} = A_{i,p-1} x_{i,p-1} + B_{i,p-1} u_{i,p-1} \end{aligned} \quad (43)$$



given  $x_{i,p-1}$ , the state of agent  $i$  at time  $\mathcal{P} - 1$ , with

$$\gamma_{ij,\tau}(u_i, u_j) = \frac{\phi_{ij}^i(u_i, \tau) + \phi_{ij}^j(u_j, \tau)}{2} - n_{ij,\tau}^i \quad (44)$$

which includes the coupling constraints between agents  $i$  and  $j$ . The optimization problem simplifies to

$$\begin{aligned} & \text{minimize}_{x_{i,p}, u_{i,p-1}} x_{i,p}^\top H_i x_{i,p} + x_{i,p-1}^\top Q_{i,p-1} x_{i,p-1} \\ & \quad + u_{i,p-1}^\top \mathcal{R}_{i,p-1} u_{i,p-1} + z_{i,p-1}^{k\top} u_{i,p-1} \\ & \text{subject to } x_{i,p} = A_{i,p-1} x_{i,p-1} + B_{i,p-1} u_{i,p-1} \end{aligned} \quad (45)$$

where

$$\mathcal{R}_{i,p-1} = R_{i,p-1} + \rho \sum_{j \in \mathcal{N}_i} (\nu_{ij,p}^{i\top} \nu_{ij,p}^i + \nu_{ij,p-1}^{i\top} \nu_{ij,p-1}^i) \quad (46)$$

and

$$z_{i,p-1}^k = \sum_{\tau=p-1}^{\mathcal{P}} \left( \nu_{i,\tau}^{i\top} q_{i,\tau}^k - 2\rho \sum_{j \in \mathcal{N}_i} \nu_{ij,\tau}^{i\top} \gamma_{ij,\tau}(u_{i,p-1}^k, u_{j,p-1}^k) \right) \quad (47)$$

for the control inputs  $u_{i,p-1}$  at iteration  $k$ . The optimization problem for the control inputs of agent  $i$  at time  $\tau$  takes the same form as the problem in (45), given by

$$\begin{aligned} & \text{minimize}_{x_{i,\tau}, u_{i,\tau-1}} x_{i,\tau}^\top \mathcal{H}_{i,\tau} x_{i,\tau} + x_{i,\tau-1}^\top Q_{i,\tau-1} x_{i,\tau-1} \\ & \quad + u_{i,\tau-1}^\top \mathcal{R}_{i,\tau-1} u_{i,\tau-1} \\ & \quad + (z_{i,\tau-1}^k - B_{i,\tau-1}^\top \mathcal{C}_{i,\tau})^\top u_{i,\tau-1} \\ & \text{subject to } x_{i,\tau} = A_{i,\tau-1} x_{i,\tau-1} + B_{i,\tau-1} u_{i,\tau-1} \end{aligned} \quad (48)$$

where

$$\mathcal{R}_{i,\tau} = R_{i,\tau} + \rho \sum_{j \in \mathcal{N}_i} \nu_{ij,\tau}^{i\top} \nu_{ij,\tau}^i \quad (49)$$

and

$$z_{i,\tau}^k = \nu_{i,\tau}^{i\top} q_{i,\tau}^k - 2\rho \sum_{j \in \mathcal{N}_i} \nu_{ij,\tau}^{i\top} \gamma_{ij,\tau}(u_{i,\tau}^k, u_{j,\tau}^k) \quad (50)$$

for  $\tau \in \{1, \dots, \mathcal{P} - 1\}$ . With

$$G_{i,\tau} = (B_{i,\tau}^\top \mathcal{H}_{i,\tau+1} B_{i,\tau} + \mathcal{R}_{i,\tau})^{-1} \quad (51)$$

agent  $i$  computes  $\mathcal{H}_{i,\tau} \in \mathbb{R}^{n_i \times n_i}$  from the Riccati equation

$$\begin{aligned} \mathcal{H}_{i,\tau} &= A_{i,\tau}^\top \mathcal{H}_{i,\tau+1} A_{i,\tau} + Q_{i,\tau} \\ & \quad - A_{i,\tau}^\top \mathcal{H}_{i,\tau+1} B_{i,\tau} G_{i,\tau} B_{i,\tau}^\top \mathcal{H}_{i,\tau+1} A_{i,\tau} \end{aligned} \quad (52)$$

beginning with  $\mathcal{H}_{i,\mathcal{P}} = H_i$ , for  $\tau = 0, \dots, \mathcal{P} - 1$ , and likewise, computes  $\mathcal{C}_{i,\tau} \in \mathbb{R}^{n_i}$  from the Riccati equation

$$\mathcal{C}_{i,\tau} = A_{i,\tau}^\top \mathcal{C}_{i,\tau+1} + A_{i,\tau}^\top \mathcal{H}_{i,\tau+1} B_{i,\tau} G_{i,\tau} (z_{i,\tau}^k - B_{i,\tau}^\top \mathcal{C}_{i,\tau+1}) \quad (53)$$

beginning with  $\mathcal{C}_{i,\mathcal{P}} = 0$ .

By solving (48), agent  $i$  obtains its control inputs

$$u_{i,\tau}^{k+1} = -G_{i,\tau} \left( B_{i,\tau}^\top \mathcal{H}_{i,\tau+1} A_{i,\tau} x_{i,\tau}^{k+1} + \frac{1}{2} (z_{i,\tau}^k - B_{i,\tau}^\top \mathcal{C}_{i,\tau+1}) \right) \quad (54)$$

with the resulting value of the objective function given by

$$\begin{aligned} f_{i,\tau} &= x_{i,\tau}^{(k+1)\top} \mathcal{H}_{i,\tau} x_{i,\tau}^{(k+1)} - x_{i,\tau}^{(k+1)\top} \mathcal{C}_{i,\tau} \\ & \quad - \frac{1}{4} \sum_{w=\tau}^{\mathcal{P}-1} (z_{i,w}^k - B_{i,w}^\top \mathcal{C}_{i,w+1})^\top \\ & \quad \times G_{i,w} (z_{i,w}^k - B_{i,w}^\top \mathcal{C}_{i,w+1}) \end{aligned} \quad (55)$$

for  $\tau = 0, \dots, \mathcal{P} - 1$ . Note that agent  $i$  needs to factorize the matrix in (51) only once when solving for its control inputs.

For each agent, the closed-loop control law in (54) consists of a feedback controller that depends on the current state of the agent and a feedforward controller that incorporates information from its neighbors into its inputs, enabling each agent to satisfy the coupling constraints in (38).

With (54), SOD-MPC provides a computationally efficient procedure for computing the optimal states and control inputs at every iteration. Agent  $i$ 's state  $x_{i,\tau}^{k+1}$  result from its dynamics model in (37) by applying  $u_{i,\tau-1}^{k+1}$  at state  $x_{i,\tau-1}^{k+1}$ .

The closed-loop control law involves factorizing the matrix in (51). The factorization procedure can be performed using the Cholesky or LDL<sup>T</sup> decomposition, which requires  $O(m_i^3)$  floating-point operations (FLOPS) in the dense case and  $O(\text{nnz}(L))$  FLOPS if sparsity is exploited, where  $L \in \mathbb{R}^{m_i \times m_i}$  denotes a lower triangular matrix. Similarly, when solving (40) using a direct numerical approach, the Hessian of the Lagrangian of (40) can be factorized using the Cholesky or LDL<sup>T</sup> decomposition, which would require  $O(N_\tau^3(n_i^3 + m_i^3))$  FLOPS in the dense case and  $O(\text{nnz}(L))$  FLOPS if sparsity is exploited, where  $L \in \mathbb{R}^{N_\tau(n_i+m_i) \times N_\tau(n_i+m_i)}$  denotes a lower triangular matrix. Hence, our approach reduces the computational complexity of solving (40) significantly, even after exploiting the sparsity of the problem or reducing the problem size by eliminating the state variables and subsequently solving for the control inputs. Particularly, the computational complexity of our method does not depend on the length of the problem horizon, unlike the computational complexity of a direct numerical approach.

While other distributed MPC methods can be applied to solve (38), these methods do not consider the effects of time delays on the control inputs applied by each agent, which can be particularly significant in problems involving agents with fast dynamics. In contrast, SOD-MPC produces a closed-loop control law, providing the robustness to time delays arising from the computation and communication procedures performed by each agent, in addition to modeling errors in the dynamics of the agents.

Algorithm 2 describes SOD-MPC for the convex MPC problem.

The agents execute the *DistributedClosedControl* procedure to compute a closed-loop controller at each time  $t$ . On completion of this procedure, each agent measures its current state at time  $t'$  and computes its feedback control inputs at its current

**Algorithm 2:** SOD-MPC for Closed-Loop Distributed MPC.

---

**do in parallel**  $i = 1, \dots, N$   
 $(x_i, u_i) \leftarrow \text{DistributedClosedControl}(t, \bar{x}_{i,t})$   
 Get state  $\bar{x}_{i,t'}$  at current time  $t'$ .  
 $\check{u}_{i,t'} \leftarrow u_{i,0}(\bar{x}_{i,t'})$   
 Apply control input  $\check{u}_{i,t'}$ .

---

**while** task is in progress;

---

**Function** DistributedClosedControl( $t, \bar{x}_{i,t}$ ).

---

Initialization:  
 $k \leftarrow 0$   
 $q_i^0 \leftarrow 0$   
 $x_{i,0} \leftarrow \bar{x}_{i,t}$

**do in parallel**  $i = 1, \dots, N$   
 Compute  $\mathcal{H}_{i,\tau}, \mathcal{C}_{i,\tau}$  with (52), (53)  $\forall \tau$ .  
**for**  $\tau = 0, \dots, \mathcal{P} - 1$  **do**  
 $u_{i,\tau}^{k+1} \leftarrow \text{Equation (54)}$   
 $x_{i,\tau+1}^{k+1} \leftarrow \text{Equation (37)}$   
**end**  
 Communication Step:  
 Agent  $i$  shares the value of  $\phi_{ij}^i(x_i^{k+1}, u_i^{k+1})$  with agent  $j$ , its neighbor, and receives the value of  $\phi_{ij}^j(x_j^{k+1}, u_j^{k+1}), \forall j \in \mathcal{N}_i$ .  
 $q_i^{k+1} \leftarrow \text{Equation (29)}$   
 $k \leftarrow k + 1$

---

**while** not converged or stopping criterion is not met;  
**return**  $(x_i, u_i)$

---

state, with its feedforward controller utilizing the most recent information received from its neighbors. Each agent repeats this procedure to obtain its control inputs at the next time instant. Noting that the states and control inputs obtained using SOD-MPC always satisfy the dynamics constraints of each agent, the *DistributedClosedControl* procedure can be terminated early if needed. In these cases, each agent uses the most recent information received from its neighbors in computing its control inputs.

## VII. SIMULATIONS

We examine the performance of SOD-MPC in convex and nonconvex MPC problems, comparing our method to other distributed MPC methods including ADMM-MPC [54] and dual decomposition [43], [44], [45], [48]. In ADMM-MPC, each agent optimizes over a copy of the states and control inputs of all its neighbors, with a consensus constraint between each pair of neighboring agents, ensuring the agents compute the same solution for equivalent variables. Consequently, ADMM-MPC differs from SOD-MPC, which does not require an agent to compute the states and control inputs of other agents. Rather, each agent in SOD-MPC optimizes over its contributions to the coupling constraints between the agent and its neighbors. We select the gradient ascent step-size in the dual decomposition algorithm and penalty parameters in ADMM-MPC and SOD-MPC using a modified binary search procedure, in which we examine the convergence rate of each algorithm at each candidate value of each parameter. Ultimately, we select the

parameters that provide the fastest convergence rates in each method. We assume each agent represents the numerical values of its local optimization variables using the double-precision floating-point format.

*Remark 2:* ADMM-MPC does not consider the relevance of neighboring agents' variables to an agent's objective or constraint functions when assigning local variables to each agent. We modify ADMM-MPC to remove extraneous variables within each agent's set of local variables, obviating unnecessary optimization over these variables, which would otherwise degrade the convergence rate of the method. In modifying ADMM-MPC, we further eliminate the need for the computation of the dual variables at a central station, extending the method beyond problems with fully-connected communication networks.

### A. Convex MPC

We consider an MPC problem involving agents with directional transceivers where optimum communication between agents requires alignment of the transceivers along an optimal surface (direction). For optimum communication while agents perform their desired tasks, we introduce a coupling constraint between neighbors, specifying the optimal surface for the transceivers for each pair of neighboring agents. The agents compute their optimal control inputs from the MPC problem

$$\begin{aligned} & \underset{x, u}{\text{minimize}} \sum_{i=1}^N f_i(x_i, u_i) \\ & \text{subject to } x_{i,0} = \bar{x}_{i,t}, \quad i = 1, \dots, N \\ & \quad x_{i,\tau+1} = A_{i,\tau}x_{i,\tau} + B_{i,\tau}u_{i,\tau} \quad \forall \tau, i = 1, \dots, N \\ & \quad \phi_{ij}(x_i, u_i, x_j, u_j) = 0 \quad \forall (i, j) \in \mathcal{E} \end{aligned} \quad (56)$$

where  $\bar{x}_{i,t}$  represents the state of agent  $i$  at time  $t$ ,  $f_i(\cdot)$  represents the objective function in (39), and  $\phi_{ij}(\cdot)$  describes the optimal surface for the transceivers on agents  $i$  and  $j$ . We define the coupling constraint between agents  $i$  and  $j$  as

$$\phi_{ij}(x_i, u_i, x_j, u_j) = e_{ij}^\top (x_i - x_j) \quad (57)$$

where  $e_{ij} \in \mathbb{R}^{n_{ij}}$  specifies the normal to the desired hyperplane for optimal communication between agents  $i$  and  $j$ . By expressing the coupling constraint in (57) in the form given in (12), we obtain that

$$\phi_{ij}^i(x_i, u_i) = e_{ij}^\top x_i \quad \text{and} \quad \phi_{ij}^j(x_j, u_j) = e_{ij}^\top x_j. \quad (58)$$

With SOD-MPC, each agent only computes its states and control inputs without computing the states and control inputs of other agents, including its neighbors. Consequently, each agent does not communicate its control inputs to other agents. As such, in SOD-MPC, agent  $i$  communicates the value of  $\phi_{ij}^i(\cdot)$ , represented by 64 b of information, with agent  $j$ . In contrast, each agent in ADMM-MPC shares 384 b of information with its neighbors. Each agent in dual decomposition shares the same amount of information as agents in SOD-MPC.

We consider a randomly-generated connected communication network between the agents. In Fig. 2, we show the transceivers on each agent along with the surfaces for optimal communication between a pair of transceivers. Each agent computes its optimal

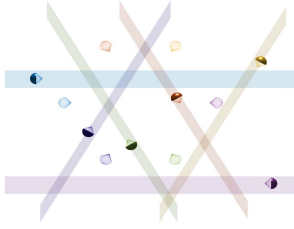


Fig. 2. Each agent has a directional transceiver displayed in the figure, with optimum communication between a pair of transceivers achieved when the transceivers lie on a collinear surface. By solving the MPC problem, each agent computes its control inputs to arrive at its desired location, depicted by the transceiver in the wireframe, while optimizing communication with its neighbors.

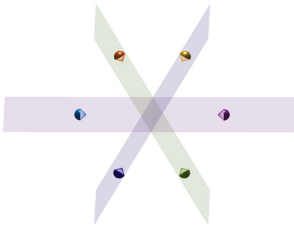


Fig. 3. At the end of the MPC problem, each agent arrives at its desired location with the transceivers oriented along collinear surfaces for optimum communication.

control inputs from (56) to arrive at its desired location, depicted in the wireframe, while optimizing communication with its neighbors. By applying the resulting control inputs, the agents arrive at their desired locations, depicted in Fig. 3.

**1) Convergence Analysis:** We begin by examining the convergence rate of SOD-MPC to the optimal solution of the centralized problem compared to the convergence rate of ADMM-MPC and dual decomposition for the MPC problem in (56) with  $N = 60$  agents and a sampling interval of 0.01 s. To visualize the solution obtained by each method in Fig. 3, we set  $n_i = 6$  and  $m_i = 3$  for agent  $i$ . In certain problems, the satisfaction of the coupling constraints between agents attains greater importance when these constraints influence the overall performance of all agents more significantly than the expended effort by each agent. As such, we compute the violations of the coupling constraints at each iteration along with the convergence of the value of the objective function to its optimal value.

Fig. 4 shows the convergence rates of SOD-MPC, dual decomposition, and ADMM-MPC to the optimal solution. The objective value of the solution obtained using SOD-MPC converges faster to the optimal objective value compared to the other methods. SOD-MPC requires about 100 iterations for convergence to the optimal objective value compared to dual decomposition, which converges in about 1000 iterations. ADMM-MPC exhibits the slowest convergence rate among all the methods. ADMM-MPC requires each agent to optimize over local copies of the optimization variables of other agents with which it shares a coupling constraint. As a result, agents in ADMM-MPC solve local optimization problems with a greater number of optimization variables compared to agents in SOD-MPC, leading to a slower convergence rate.

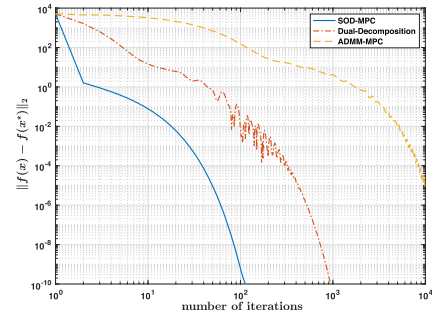


Fig. 4. Convergence of SOD-MPC, dual decomposition, and ADMM-MPC to the optimal objective value in convex MPC. SOD-MPC provides the fastest convergence rate to the optimal objective value, converging in about 100 iterations. Dual decomposition, which converges faster than ADMM-MPC, requires about 1000 iterations.

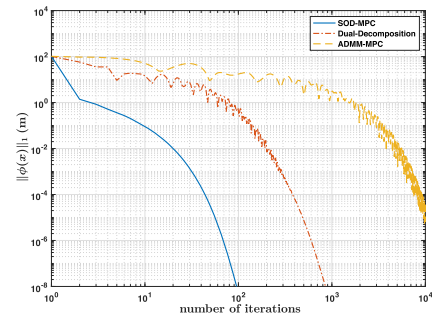


Fig. 5. Violation of the coupling constraints between agents using SOD-MPC, dual decomposition, and ADMM-MPC. The violations of these constraints in SOD-MPC converge rapidly to zero in about 100 iterations while dual decomposition requires about 1000 iterations for convergence of the coupling constraint violations.

In addition, we examine the violations of the coupling constraints between agents in Fig. 5. The violations of the coupling constraints between agents converge rapidly to zero with SOD-MPC, compared to dual decomposition and ADMM-MPC. Hence, SOD-MPC not only converges rapidly to the optimal objective value, the violations of the coupling constraints in SOD-MPC decrease rapidly likewise. As with convergence to the optimal objective value, the violations of the coupling constraints in SOD-MPC converge to zero in about 100 iterations while dual decomposition, which converges faster than ADMM-MPC, requires about 1000 iterations. With its faster convergence rates, SOD-MPC enables each agent to efficiently compute its control inputs, especially in problems with high control bandwidths. Moreover, faster convergence of the coupling constraint violations enables the application of SOD-MPC to problems where the satisfaction of these constraints significantly influences the performance of all agents.

**2) Closed-Loop Performance:** Regardless of the method applied in solving the MPC problem in (56), the computation and communication procedures performed by the agents or a central station introduce time delays between the specification of the MPC problem and the application of the control inputs by each agent. Note that the MPC problem in (56) depends on the state of each agent at time  $t$  and other time-dependent problem

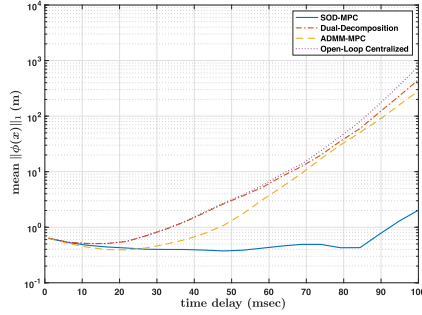


Fig. 6. Violations of the coupling constraints between agents for different time delays using SOD-MPC, dual decomposition, ADMM-MPC, and an open-loop centralized method. The mean violation of the coupling constraints increases rapidly for dual decomposition, ADMM-MPC, and the open-loop centralized method as the duration of the time delay increases. However, SOD-MPC exhibits the robustness to time delays with a mean coupling constraints violation close to 1 m even at a time delay of 100 ms where the open-loop centralized method attains a mean coupling constraints violation close to 1000 m.

parameters. Considering the time delays, the resulting control inputs will no longer be optimal for each agent at the time of application of the control inputs, which can severely impact the overall performance of all agents, especially in problems involving agents with fast dynamics.

We examine the performance of SOD-MPC, dual decomposition, ADMM-MPC, and an open-loop centralized method in the MPC problem (56) for different time delays in Fig. 6. We do not modify the MPC problems to account for the time delay between the specification of the control problem and the application of the computed control inputs. In our simulations, we propagate each agent's dynamics forward in time assuming its control inputs are set to zero while the agent is still computing its control inputs for the current instance of the model predictive optimization problem. At small time delays, all the methods attain almost the same mean violation of the coupling constraints between agents. However, the mean violation of the coupling constraints increases precipitously in dual decomposition, ADMM-MPC, and the open-loop centralized method as the duration of the time delay increases. These methods do not consider the effects of time delays in the controllers utilized by each agent; hence, the mean violation of the coupling constraints in the centralized method rises to almost 1000 m at a time delay of 100 ms. In contrast, SOD-MPC attains a mean coupling constraints violation close to 1 m even at a time delay of 100 ms. The closed-loop controller provided by SOD-MPC enables each agent to mitigate the effects of time delays, improving the performance of all agents in the MPC problem. Eventually, when the time delay exceeds 100 ms, the model predictive controllers resulting from dual decomposition, ADMM-MPC, and the open-loop centralized method become unstable.

In Fig. 7, we show the violations of the coupling constraints for the first 60 time intervals at a time delay of 100 ms. The agents begin their tasks with their transceivers not aligned along the optimal surface for communication with their neighbors. Dual decomposition, ADMM-MPC, and the open-loop centralized method fail to align the transceivers on each agent to the optimal surface specified in the coupling constraints as the

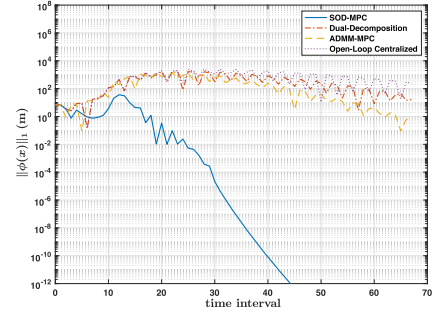


Fig. 7. Violations of the coupling constraints between agents for the first 60 time intervals at a time delay of 100 ms. While the violations of the coupling constraints do not converge with other methods, SOD-MPC attains convergence of the violations of the coupling constraints to zero.

agents proceed with their tasks. As such, the violations of the coupling constraints do not converge to zero with these methods. However, the violations of the coupling constraints converge to zero in SOD-MPC, with each agent aligning its transceiver along the specified optimal surface in its coupling constraints.

## B. Nonconvex MPC

We consider the deployment of satellites using nonconvex MPC. We assume the satellites are equipped with thrusters, enabling them to maneuver to specified desired positions and orientations. We represent the orientation of each satellite using a rotation matrix  $C(t) \in \mathbb{R}^{3 \times 3}$  at time  $t$  with the angular velocity  $\omega(t) \in \mathbb{R}^3$ . Over a sampling interval  $\delta t$  with an angular velocity  $\omega(t)$ , the satellite rotates through an angle  $\alpha \in \mathbb{R}$  around the axis  $a \in \mathbb{R}^3$  with the dynamics of the rotation matrix given by the model

$$C(t + \delta t) = D(t)C(t) \quad (59)$$

with

$$D(t) = \cos(\alpha)I + (1 - \cos(\alpha))aa^T + \sin(\alpha)\hat{\omega} \quad (60)$$

where  $I \in \mathbb{R}^{3 \times 3}$  represents the identity matrix,  $\hat{\omega} \in \mathbb{R}^{3 \times 3}$  represents a skew-symmetric matrix derived from  $\omega$ ,  $\alpha = \|\omega \cdot \delta t\|_2$ , and  $a = \frac{\omega \cdot \delta t}{\alpha}$ .

We consider the presence of other orbiting agents and space debris in the vicinity of the satellites and thus introduce collision-avoidance constraints in the MPC problem to prevent collisions. In addition, we include nonconvex coupling constraints between neighboring satellites, which indicate the dependence of the quality of service provided by the group of satellites on the states of each satellite.

The satellites compute their inputs  $\mathcal{F}$  from the MPC problem

$$\begin{aligned} & \underset{x, C, \mathcal{F}}{\text{minimize}} \sum_{i=1}^N f_i(x_i, C_i, \mathcal{F}_i) \\ & \text{subject to } d_i(x_i, C_i, \mathcal{F}_i) = 0, \quad i = 1, \dots, N \\ & \quad g_i(x_i, C_i, \mathcal{F}_i) = 0, \quad i = 1, \dots, N \\ & \quad h_i(x_i, C_i, \mathcal{F}_i) \leq 0, \quad i = 1, \dots, N \\ & \quad \phi_{ij}(x_i, C_i, \mathcal{F}_i, x_j, C_j, \mathcal{F}_j) = 0 \quad \forall (i, j) \in \mathcal{E} \end{aligned} \quad (61)$$

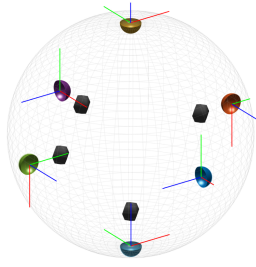


Fig. 8. Deployment of satellites to their orbits in space using nonconvex MPC. Using SOD-MPC, each satellite computes its control inputs locally while satisfying coupling constraints with other satellites and avoiding collisions with space debris, denoted by the gray objects.

where  $x_{i,\tau} \in \mathbb{R}^6$  represents the linear states and velocities of satellite  $i$  at time  $\tau$ ,  $x_i = [x_{i,\tau}^T, \tau = 0, \dots, \mathcal{P}]^T$  and  $\mathcal{F}$  and  $\mathcal{C}$  represent the concatenation of the inputs and rotation matrices of all the satellites, respectively. We represent the dynamics constraints of satellite  $i$  with  $d_i(\cdot)$  and include constraints on its initial and desired position, orientation, and velocities in  $g_i(\cdot)$ . In  $h_i(\cdot)$ , we specify the inputs constraints as well as the collision-avoidance constraints of satellite  $i$ , given by

$$\text{dist}(x_i, p) \geq \lambda_{\min} \quad \forall p \in \mathcal{A} \quad (62)$$

where  $\mathcal{A}$  comprises of the state of all space debris detected by satellite  $i$  and the state of all satellites in its vicinity. We denote the minimum distance between a satellite and other agents as  $\lambda_{\min}$ . In this problem, we define the distance function  $\text{dist}(\cdot)$  using the Euclidean norm. We specify that neighboring satellites maintain the same altitude, giving rise to the nonconvex coupling constraint  $\phi_{ij}(\cdot)$  between satellites  $i$  and  $j$ .

Using SOD-MPC, satellite  $i$  computes its inputs  $\mathcal{F}_i \in \mathbb{R}^6$  locally, without computing the states and inputs of other satellites. Each satellite applies its resulting inputs, satisfying its coupling constraints with other satellites while avoiding collisions with other orbiting agents and space debris, as depicted in Fig. 8. At each iteration, each agent in SOD-MPC and dual decomposition shares 64 b of information with its neighbors. In contrast, each agent in ADMM-MPC shares 384 b of information with its neighbors.

We assess the performance of SOD-MPC, dual decomposition, and ADMM-MPC in this problem (61), over a randomly-generated connected communication network with  $N = 12$  satellites and a 0.01-s sampling interval, solving each problem using the interior-point optimization solver IPOPT [71] with the linear solver MA-57 and the maximum number of iterations at 300. We examine the convergence of the value of the objective function obtained from each method to the objective value from a centralized method in Fig. 9. The objective value of the solution obtained using SOD-MPC converges faster than the objective value resulting from dual decomposition and ADMM-MPC, converging in about 600 iterations to the optimal objective value. ADMM-MPC converges faster than dual decomposition, requiring about 1000 iterations for convergence.

Likewise, we examine the convergence of the violations of the coupling constraints between the satellites in (61) to zero for

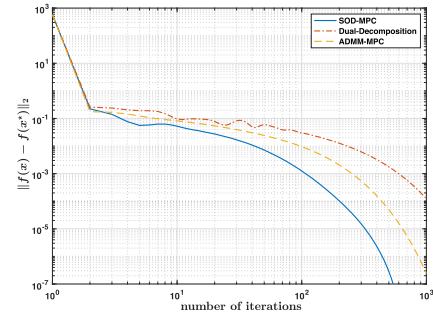


Fig. 9. Convergence of the objective value obtained using SOD-MPC, dual decomposition, and ADMM-MPC to the objective value obtained from a centralized method. SOD-MPC provides the fastest convergence rate and converges in about 600 iterations. ADMM-MPC, which converges faster than dual decomposition, requires about 1000 iterations for convergence.

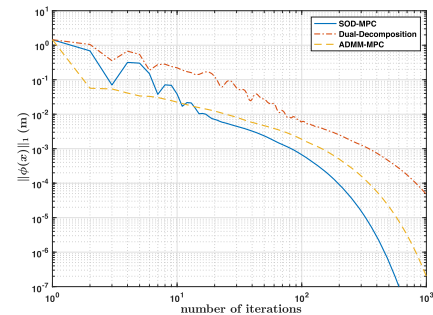


Fig. 10. Convergence of the violations of the coupling constraints between agents using SOD-MPC, dual decomposition, and ADMM-MPC. SOD-MPC converges faster than dual decomposition and ADMM-MPC in about 600 iterations, with ADMM-MPC converging faster than dual decomposition in about 1000 iterations.

SOD-MPC, dual decomposition, and ADMM-MPC, in Fig. 10. SOD-MPC attains the fastest convergence rate with the violations of the coupling constraints converging to zero in about 600 iterations. ADMM-MPC requires about 1000 iterations for convergence of the coupling constraint violations to zero, converging faster than dual decomposition. Hence, SOD-MPC provides a more efficient method for computing the control inputs of each satellite in addition to satisfying the coupling constraints between the satellites.

## VIII. CONCLUSION

We introduce an algorithm for distributed MPC, SOD-MPC, where each agent computes its control inputs locally, in parallel, without computing the control inputs of other agents irrelevant to its performance. Each agent does not rely on a central station for computing its control inputs, unlike some other distributed methods. Our algorithm achieves the linear convergence to the optimal control inputs, providing faster convergence rates compared to other distributed MPC methods including dual decomposition, in convex problems with strongly convex objective functions. In addition, SOD-MPC produces a closed-loop controller in convex MPC problems with affine constraints, providing the robustness to time delays arising from

the communication and computation procedures performed by the agents.

## APPENDIX

### A. Proof of Lemma 1

We provide the proof of Lemma 1. For the problem in (30), we express the coupling constraints between all agents as

$$\Phi(\mathbf{u}) = \mathcal{J}\mathbf{u} + \eta \quad (63)$$

where  $\mathcal{J}$  represents the Jacobian of the coupling constraints with respect to  $\mathbf{u}$  and  $\eta$  represents constant terms in the coupling constraints. For convex problems, note that the coupling constraints are affine. As such, we can always obtain the constraint in (63) in any convex MPC problem. We can further express the Jacobian as

$$\mathcal{J} = \vec{\mathcal{J}} - \overleftarrow{\mathcal{J}} \quad (64)$$

where  $\vec{\mathcal{J}}_{ij}$  denotes the Jacobian of the coupling constraint  $\phi_{ij}(\cdot)$ , represented by the edge  $(i, j) \in \mathcal{E}$ , with  $\phi_{ij}^j(\cdot) = 0$  and  $\overleftarrow{\mathcal{J}}_{ij}$  denotes the Jacobian of  $\phi_{ij}(\cdot)$  with  $\phi_{ij}^i(\cdot) = 0$ . Likewise, we express the constant terms in the coupling constraints as

$$\eta = \vec{\eta} - \overleftarrow{\eta} \quad (65)$$

with  $\vec{\eta}_{ij} = \eta_{ij}^i$  and  $\overleftarrow{\eta}_{ij} = \eta_{ij}^j$ . By defining  $\vec{\mathcal{J}}$  and  $\vec{\eta}$  as

$$\begin{aligned} \vec{\mathcal{J}} &= \vec{\mathcal{J}} + \overleftarrow{\mathcal{J}} \\ \vec{\eta} &= \vec{\eta} + \overleftarrow{\eta} \end{aligned} \quad (66)$$

we can express the update procedures for the auxiliary variables  $c$  and dual variables  $q$  of all agents (29) as

$$c^{k+1} = \frac{1}{2} (\vec{\mathcal{J}}\mathbf{u}^{k+1} + \vec{\eta}) \quad (67)$$

with

$$q^{k+1} = q^k + \rho (\mathcal{J}\mathbf{u}^{k+1} + \eta). \quad (68)$$

We define the  $W$ -norm as  $\|\cdot\|_W$  with

$$W = \begin{bmatrix} \frac{1}{4\rho}I & 0 \\ 0 & \rho I \end{bmatrix} \quad (69)$$

where  $W > 0$ .

Following the procedure in (25), the control inputs of all agents satisfy

$$\nabla\mathbb{F}(\mathbf{u}^{k+1}) + \frac{1}{2}\mathcal{J}^\top q^{k+1} + \rho\overleftarrow{\mathcal{J}}^\top (c^{k+1} - c^k) = 0 \quad (70)$$

for the update procedure at iteration  $k$ . The optimal control inputs satisfy the optimality conditions

$$\nabla\mathbb{F}(\mathbf{u}^*) + \frac{1}{2}\mathcal{J}^\top q^* = 0 \quad (71)$$

with the optimal auxiliary variables and dual variables satisfying

$$\begin{aligned} c^* &= \frac{1}{2} (\vec{\mathcal{J}}\mathbf{u}^* + \vec{\eta}) \\ 0 &= \rho (\mathcal{J}\mathbf{u}^* + \eta) \end{aligned} \quad (72)$$

from (67) and (68). With (70) and (71), we obtain

$$\begin{aligned} \nabla\mathbb{F}(\mathbf{u}^{k+1}) - \nabla\mathbb{F}(\mathbf{u}^*) + \frac{1}{2}\mathcal{J}^\top (q^{k+1} - q^*) \\ + \rho\overleftarrow{\mathcal{J}}^\top (c^{k+1} - c^k) = 0 \end{aligned} \quad (73)$$

along with

$$\begin{aligned} c^{k+1} - c^* &= \frac{1}{2}\vec{\mathcal{J}}(\mathbf{u}^{k+1} - \mathbf{u}^*) \\ q^{k+1} - q^k &= \rho\mathcal{J}(\mathbf{u}^{k+1} - \mathbf{u}^*) \end{aligned} \quad (74)$$

from (67), (68), and (72).

For a strongly convex function  $f(x)$

$$(\nabla f(x_a) - \nabla f(x_b))^\top (x_a - x_b) \geq m_f \|x_a - x_b\|_2^2 \quad (75)$$

where  $m_f > 0$ . For the convex MPC problem in (30)

$$\begin{aligned} (\nabla\mathbb{F}(\mathbf{u}^{k+1}) - \nabla\mathbb{F}(\mathbf{u}^*))^\top (\mathbf{u}^{k+1} - \mathbf{u}^*) \\ = - \left( \frac{1}{2}\mathcal{J}^\top (q^{k+1} - q^*) + \rho\overleftarrow{\mathcal{J}}^\top (c^{k+1} - c^k) \right)^\top (\mathbf{u}^{k+1} - \mathbf{u}^*) \\ = -2 (z^{k+1} - z^*)^\top W (z^{k+1} - z^k) \\ = \|z^k - z^*\|_W^2 - \|z^{k+1} - z^*\|_W^2 - \|z^{k+1} - z^k\|_W^2 \end{aligned} \quad (76)$$

from (74). Consequently

$$\begin{aligned} \|z^k - z^*\|_W^2 - \|z^{k+1} - z^*\|_W^2 \\ \geq m_f \|\mathbf{u}^{k+1} - \mathbf{u}^*\|_2^2 + \|z^{k+1} - z^k\|_W^2 \end{aligned} \quad (77)$$

from (75).

### B. Proof of Lemma 2

We prove Lemma 2. With Lipschitz continuity of  $\nabla\mathbb{F}(\mathbf{u})$

$$\|\nabla\mathbb{F}(\mathbf{u}^{k+1}) - \nabla\mathbb{F}(\mathbf{u}^*)\|_2^2 \leq L_f^2 \|\mathbf{u}^{k+1} - \mathbf{u}^*\|_2^2 \quad (78)$$

where  $L_f > 0$ . From (67)

$$\|c^{k+1} - c^*\|_2^2 \leq \frac{1}{4} \|\vec{\mathcal{J}}\|_2^2 \|\mathbf{u}^{k+1} - \mathbf{u}^*\|_2^2. \quad (79)$$

Using the relation  $\|a + b\|_2^2 \geq (\mu - 1)\|a\|_2^2 + \frac{\mu-1}{\mu}\|b\|_2^2$  for  $\mu > 1$  and (73)

$$\begin{aligned} \left\| \frac{1}{2}\mathcal{J}^\top (q^{k+1} - q^*) + \rho\overleftarrow{\mathcal{J}}^\top (c^{k+1} - c^k) \right\|_2^2 \\ \geq \frac{\mu-1}{4\mu} \sigma_{\min}^2(\mathcal{J}) \|q^{k+1} - q^*\|_2^2 \\ + (1-\mu)\rho^2 \sigma_{\min}^2(\overleftarrow{\mathcal{J}}) \|c^{k+1} - c^k\|_2^2 \end{aligned} \quad (80)$$

which gives

$$\begin{aligned} \frac{\mu-1}{4\mu} \sigma_{\min}^2(\mathcal{J}) \|q^{k+1} - q^*\|_2^2 \\ + (1-\mu)\rho^2 \sigma_{\min}^2(\overleftarrow{\mathcal{J}}) \|c^{k+1} - c^k\|_2^2 \\ \leq L_f^2 \|\mathbf{u}^{k+1} - \mathbf{u}^*\|_2^2 \end{aligned} \quad (81)$$

resulting in

$$\begin{aligned} & \frac{1}{4\rho} \|q^{k+1} - q^*\|_2^2 + \rho \|c^{k+1} - c^*\|_2^2 \\ & \leq \frac{\rho\mu\sigma_{\min}^2(\bar{\mathcal{J}})}{\sigma_{\min}^2(\mathcal{J})} \|c^{k+1} - c^k\|_2^2 \\ & \quad + \left( \frac{\mu L_f^2}{\rho(\mu-1)\sigma_{\min}^2(\mathcal{J})} + \frac{\rho}{4}\sigma_{\max}^2(\bar{\mathcal{J}}) \right) \|u^{k+1} - u^*\|_2^2 \end{aligned} \quad (82)$$

with (79).

Simplifying (82) gives

$$\begin{aligned} & \frac{\ell}{4\rho} \|q^{k+1} - q^*\|_2^2 + \ell\rho \|c^{k+1} - c^*\|_2^2 \\ & \leq \rho \|c^{k+1} - c^k\|_2^2 + m_f \|u^{k+1} - u^*\|_2^2 \end{aligned} \quad (83)$$

where

$$\ell = \min \left\{ \frac{4m_f\rho(\mu-1)\sigma_{\min}^2(\mathcal{J})}{\rho^2(\mu-1)\sigma_{\max}^2(\bar{\mathcal{J}})\sigma_{\min}^2(\mathcal{J}) + 4\mu L_f^2}, \frac{\sigma_{\min}^2(\mathcal{J})}{\mu\sigma_{\min}^2(\bar{\mathcal{J}})} \right\} \quad (84)$$

with  $\ell > 0$ . From (77) and (83)

$$\ell \|z^{k+1} - z^*\|_W^2 \leq \|z^k - z^*\|_W^2 - \|z^{k+1} - z^*\|_W^2 \quad (85)$$

resulting in

$$\frac{\|z^{k+1} - z^*\|_W^2}{\|z^k - z^*\|_W^2} \leq \frac{1}{\ell + 1} \quad (86)$$

showing Q-linear convergence of  $\{z^k\}$  to  $z^*$ .

## REFERENCES

- [1] M. Mercangöz and F. J. Doyle III, "Distributed model predictive control of an experimental four-tank system," *J. Process Control*, vol. 17, no. 3, pp. 297–308, 2007.
- [2] D. B. Pourkargar, A. Almansoori, and P. Daoutidis, "Impact of decomposition on distributed model predictive control: A process network case study," *Ind. Eng. Chem. Res.*, vol. 56, no. 34, pp. 9606–9616, 2017.
- [3] T. Faulwasser et al., "Economic nonlinear model predictive control," *Found. Trends Syst. Control*, vol. 5, no. 1, pp. 1–98, 2018.
- [4] B. Stellato, T. Geyer, and P. J. Goulart, "High-speed finite control set model predictive control for power electronics," *IEEE Trans. Power Electron.*, vol. 32, no. 5, pp. 4007–4020, May 2017.
- [5] T. Dragičević, "Model predictive control of power converters for robust and fast operation of AC microgrids," *IEEE Trans. Power Electron.*, vol. 33, no. 7, pp. 6304–6317, Jul. 2018.
- [6] G. Mantovani and L. Ferrarini, "Temperature control of a commercial building with model predictive control techniques," *IEEE Trans. Ind. Electron.*, vol. 62, no. 4, pp. 2651–2660, Apr. 2015.
- [7] G. Gholamibozanjani, J. Tarragona, A. De Gracia, C. Fernández, L. F. Cabeza, and M. M. Farid, "Model predictive control strategy applied to different types of building for space heating," *Appl. Energy*, vol. 231, pp. 959–971, 2018.
- [8] J. Álvarez, J. Redondo, E. Camponogara, J. Normey-Rico, M. Berenguel, and P. Ortigosa, "Optimizing building comfort temperature regulation via model predictive control," *Energy Buildings*, vol. 57, pp. 361–372, 2013.
- [9] D. Picard, J. Drgoña, M. Kvasnica, and L. Helsen, "Impact of the controller model complexity on model predictive control performance for buildings," *Energy Buildings*, vol. 152, pp. 739–751, 2017.
- [10] M. Maasoumy, M. Razmara, M. Shahbakhthi, and A. S. Vincentelli, "Handling model uncertainty in model predictive control for energy efficient buildings," *Energy Buildings*, vol. 77, pp. 377–392, 2014.
- [11] G. Bianchini, M. Casini, A. Vicino, and D. Zarrilli, "Demand-response in building heating systems: A model predictive control approach," *Appl. Energy*, vol. 168, pp. 159–170, 2016.
- [12] Y. Kwak, J.-H. Huh, and C. Jang, "Development of a model predictive control framework through real-time building energy management system data," *Appl. Energy*, vol. 155, pp. 1–13, 2015.
- [13] R. Tang and S. Wang, "Model predictive control for thermal energy storage and thermal comfort optimization of building demand response in smart grids," *Appl. Energy*, vol. 242, pp. 873–882, 2019.
- [14] Y. Zheng, S. E. Li, K. Li, F. Borrelli, and J. K. Hedrick, "Distributed model predictive control for heterogeneous vehicle platoons under unidirectional topologies," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 3, pp. 899–910, May 2017.
- [15] E. van Nunen, J. Reinders, E. Semsar-Kazerooni, and N. V. D. Wouw, "String stable model predictive cooperative adaptive cruise control for heterogeneous platoons," *IEEE Trans. Intell. Veh.*, vol. 4, no. 2, pp. 186–196, Jun. 2019.
- [16] J. Wang, S. Gong, S. Peeta, and L. Lu, "A real-time deployable model predictive control-based cooperative platooning approach for connected and autonomous vehicles," *Transp. Res. Part B, Methodol.*, vol. 128, pp. 271–301, 2019.
- [17] K. Yu et al., "Model predictive control for hybrid electric vehicle platooning using slope information," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 7, pp. 1894–1909, Jul. 2016.
- [18] R. Kianfar, P. Falcone, and J. Fredriksson, "A control matching model predictive control approach to string stable vehicle platooning," *Control Eng. Pract.*, vol. 45, pp. 163–173, 2015.
- [19] L. Chisci, T. Pecorella, and R. Fantacci, "Dynamic bandwidth allocation in geo satellite networks: A predictive control approach," *Control Eng. Pract.*, vol. 14, no. 9, pp. 1057–1067, 2006.
- [20] K. Xiao, S. Mao, and J. K. Tugnait, "MAQ: A multiple model predictive congestion control scheme for cognitive radio networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 4, pp. 2614–2626, Apr. 2017.
- [21] E. Henriksson, D. E. Quevedo, E. G. Peters, H. Sandberg, and K. H. Johansson, "Multiple-loop self-triggered model predictive control for network scheduling and control," *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 6, pp. 2167–2181, Nov. 2015.
- [22] M. Neunert et al., "Fast nonlinear model predictive control for unified trajectory optimization and tracking," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1398–1404.
- [23] C. Shen and Y. Shi, "Distributed implementation of nonlinear model predictive control for AUV trajectory tracking," *Automatica*, vol. 115, 2020, Art. no. 108863.
- [24] M. Bangura and R. Mahony, "Real-time model predictive control for quadrotors," *IFAC Proc. Volumes*, vol. 47, no. 3, pp. 11773–11780, 2014.
- [25] A. Alessandretti, A. P. Aguiar, and C. N. Jones, "Trajectory-tracking and path-following controllers for constrained underactuated vehicles using model predictive control," in *Proc. Eur. Control Conf.*, 2013, pp. 1371–1376.
- [26] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, "Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints," *IEEE Trans. Veh. Technol.*, vol. 66, no. 2, pp. 952–964, Feb. 2017.
- [27] P. D. Christofides, R. Scattolini, D. M. de la Pena, and J. Liu, "Distributed model predictive control: A tutorial review and future research directions," *Comput. Chem. Eng.*, vol. 51, pp. 21–41, 2013.
- [28] M. Farina and R. Scattolini, "Distributed predictive control: A non-cooperative algorithm with neighbor-to-neighbor communication for linear systems," *Automatica*, vol. 48, no. 6, pp. 1088–1096, 2012.
- [29] A. Bestler and K. Graichen, "Distributed model predictive control for continuous-time nonlinear systems based on suboptimal ADMM," *Optimal Control Appl. Methods*, vol. 40, no. 1, pp. 1–23, 2019.
- [30] H. Zheng, R. R. Negenborn, and G. Lodewijks, "Fast ADMM for distributed model predictive control of cooperative waterborne AGVs," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 4, pp. 1406–1413, Jul. 2017.
- [31] J. F. Mota, J. M. Xavier, P. M. Aguiar, and M. Püschel, "Distributed ADMM for model predictive control and congestion control," in *Proc. IEEE 51st Conf. Decis. Control*, 2012, pp. 5110–5115.
- [32] B. T. Stewart, A. N. Venkat, J. B. Rawlings, S. J. Wright, and G. Panocchia, "Cooperative distributed model predictive control," *Syst. Control Lett.*, vol. 59, no. 8, pp. 460–469, 2010.
- [33] M. A. Müller, M. Reble, and F. Allgöwer, "Cooperative control of dynamically decoupled systems via distributed model predictive control," *Int. J. Robust Nonlinear Control*, vol. 22, no. 12, pp. 1376–1397, 2012.

- [34] O. Shorinwa, T. Halsted, and M. Schwager, "Scalable distributed optimization with separable variables in multi-agent networks," in *Proc. Amer. Control Conf.*, 2020, pp. 3619–3626.
- [35] B. T. Stewart, S. J. Wright, and J. B. Rawlings, "Cooperative distributed model predictive control for nonlinear systems," *J. Process Control*, vol. 21, no. 5, pp. 698–704, 2011.
- [36] M. Heidarinejad, J. Liu, D. M. de la Peña, J. F. Davis, and P. D. Christofides, "Multirate Lyapunov-based distributed model predictive control of nonlinear uncertain systems," *J. Process Control*, vol. 21, no. 9, pp. 1231–1242, 2011.
- [37] J. Liu, D. Muñoz de la Peña, and P. D. Christofides, "Distributed model predictive control of nonlinear process systems," *AIChE J.*, vol. 55, no. 5, pp. 1171–1184, 2009.
- [38] J. Liu, D. M. de la Peña, and P. D. Christofides, "Distributed model predictive control of nonlinear systems subject to asynchronous and delayed measurements," *Automatica*, vol. 46, no. 1, pp. 52–61, 2010.
- [39] M. R. Jovanovic and B. Bamieh, "Lyapunov-based distributed control of systems on lattices," *IEEE Trans. Autom. Control*, vol. 50, no. 4, pp. 422–433, Apr. 2005.
- [40] A. N. Venkat, J. B. Rawlings, and S. J. Wright, "Stability and optimality of distributed model predictive control," in *Proc. IEEE 44th Conf. Decis. Control*, 2005, pp. 6680–6685.
- [41] B. Alrifai, F.-J. Heßeler, and D. Abel, "Coordinated non-cooperative distributed model predictive control for decoupled systems using graphs," *IFAC-PapersOnLine*, vol. 49, no. 22, pp. 216–221, 2016.
- [42] S. Rivero and G. Ferrari-Trecate, "Tube-based distributed control of linear constrained systems," *Automatica*, vol. 48, no. 11, pp. 2860–2865, 2012.
- [43] Y. Wakasa, M. Arakawa, K. Tanaka, and T. Akashi, "Decentralized model predictive control via dual decomposition," in *Proc. IEEE 47th Conf. Decis. Control*, 2008, pp. 381–386.
- [44] F. Farokhi, I. Shames, and K. H. Johansson, "Distributed MPC via dual decomposition and alternative direction method of multipliers," in *Distributed Model Predictive Control Made Easy*. Berlin, Germany: Springer, 2014, pp. 115–131.
- [45] A. Grancharova and T. A. Johansen, "Distributed quasi-nonlinear model predictive control by dual decomposition," *IFAC Proc. Volumes*, vol. 44, no. 1, pp. 1429–1434, 2011.
- [46] C. Conte, C. N. Jones, M. Morari, and M. N. Zeilinger, "Distributed synthesis and stability of cooperative distributed model predictive control for linear systems," *Automatica*, vol. 69, pp. 117–125, 2016.
- [47] J. Koko, "A survey on dual decomposition methods," *SeMA J.*, vol. 62, no. 1, pp. 27–59, 2013.
- [48] P. Giselsson, M. D. Doan, T. Keviczky, B. De Schutter, and A. Rantzer, "Accelerated gradient methods and dual decomposition in distributed model predictive control," *Automatica*, vol. 49, no. 3, pp. 829–833, 2013.
- [49] S. Koehler, C. Danielson, and F. Borrelli, "A primal-dual active-set method for distributed model predictive control," *Optimal Control Appl. Methods*, vol. 38, no. 3, pp. 399–419, 2017.
- [50] E. Camponogara and H. F. Scherer, "Distributed optimization for model predictive control of linear dynamic networks with control-input and output constraints," *IEEE Trans. Automat. Sci. Eng.*, vol. 8, no. 1, pp. 233–242, Jan. 2011.
- [51] E. Camponogara and L. B. De Oliveira, "Distributed optimization for model predictive control of linear-dynamic networks," *IEEE Trans. Syst., Man, Cybern.—Part A, Syst. Humans*, vol. 39, no. 6, pp. 1331–1338, Nov. 2009.
- [52] Z. Wang and C. J. Ong, "Distributed model predictive control of linear discrete-time systems with local and global constraints," *Automatica*, vol. 81, pp. 184–195, 2017.
- [53] R. Rostami, G. Costantini, and D. Görges, "ADMM-based distributed model predictive control: Primal and dual approaches," in *Proc. IEEE 56th Annu. Conf. Decis. Control*, 2017, pp. 6598–6603.
- [54] T. H. Summers and J. Lygeros, "Distributed model predictive consensus via the alternating direction method of multipliers," in *Proc. 50th Annu. Allerton Conf. Commun., Control, Comput.*, 2012, pp. 79–84.
- [55] P. Braun, T. Faulwasser, L. Grüne, C. M. Kellett, S. R. Weller, and K. Worthmann, "Hierarchical distributed ADMM for predictive control with applications in power networks," *IFAC J. Syst. Control*, vol. 3, pp. 10–22, 2018.
- [56] X. Hou, Y. Xiao, J. Cai, J. Hu, and J. E. Braun, "Distributed model predictive control via proximal Jacobian ADMM for building control applications," in *Proc. Amer. Control Conf.*, 2017, pp. 37–43.
- [57] P. Giselsson and A. Rantzer, "On feasibility, stability and performance in distributed model predictive control," *IEEE Trans. Autom. Control*, vol. 59, no. 4, pp. 1031–1036, Apr. 2014.
- [58] J. Köhler, M. A. Müller, and F. Allgöwer, "Distributed model predictive control—recursive feasibility under inexact dual optimization," *Automatica*, vol. 102, pp. 1–9, 2019.
- [59] M. Rubagotti, P. Patrinos, and A. Bemporad, "Stabilizing linear model predictive control under inexact numerical optimization," *IEEE Trans. Autom. Control*, vol. 59, no. 6, pp. 1660–1666, Jun. 2014.
- [60] M. Kögel and R. Findeisen, "Stabilization of inexact MPC schemes," in *Proc. IEEE 53rd Conf. Decis. Control*, 2014, pp. 5922–5928.
- [61] D. Limón, T. Alamo, F. Salas, and E. F. Camacho, "On the stability of constrained MPC without terminal constraint," *IEEE Trans. Autom. Control*, vol. 51, no. 5, pp. 832–836, May 2006.
- [62] P. O. Scokaert, D. Q. Mayne, and J. B. Rawlings, "Suboptimal model predictive control (feasibility implies stability)," *IEEE Trans. Autom. Control*, vol. 44, no. 3, pp. 648–654, Mar. 1999.
- [63] G. Pannocchia, J. B. Rawlings, and S. J. Wright, "Conditions under which suboptimal nonlinear MPC is inherently robust," *Syst. Control Lett.*, vol. 60, no. 9, pp. 747–755, 2011.
- [64] D. A. Allan, C. N. Bates, M. J. Risbeck, and J. B. Rawlings, "On the inherent robustness of optimal and suboptimal nonlinear MPC," *Syst. Control Lett.*, vol. 106, pp. 68–78, 2017.
- [65] V. Kekatos and G. B. Giannakis, "Distributed robust power system state estimation," *IEEE Trans. Power Syst.*, vol. 28, no. 2, pp. 1617–1626, May 2013.
- [66] J. F. Mota, J. M. Xavier, P. M. Aguiar, and M. Püschel, "Distributed optimization with local domains: Applications in MPC and network flows," *IEEE Trans. Autom. Control*, vol. 60, no. 7, pp. 2004–2009, Jul. 2015.
- [67] Y. Wang, W. Yin, and J. Zeng, "Global convergence of ADMM in nonconvex nonsmooth optimization," *J. Sci. Comput.*, vol. 78, no. 1, pp. 29–63, 2019.
- [68] L. Chen, D. Sun, and K.-C. Toh, "A note on the convergence of ADMM for linearly constrained convex optimization problems," *Comput. Optim. Appl.*, vol. 66, no. 2, pp. 327–343, 2017.
- [69] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson, "Optimal parameter selection for the alternating direction method of multipliers (ADMM): Quadratic problems," *IEEE Trans. Autom. Control*, vol. 60, no. 3, pp. 644–658, Mar. 2015.
- [70] A. U. Raghunathan and S. Di Cairano, "Optimal step-size selection in alternating direction method of multipliers for convex quadratic programs and model predictive control," in *Proc. Symp. Math. Theory Netw. Syst.*, 2014, pp. 807–814.
- [71] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.



**Ola Shorinwa** (Graduate Student Member, IEEE) is currently working toward the Ph.D. degree in mechanical engineering with the Department of Mechanical Engineering, Stanford University, Stanford, CA, USA.

He is currently a member of the Multi-Robot Systems Lab, Department of Mechanical Engineering, Stanford University. His research interests include distributed control, optimization, estimation, and planning for multirobot systems.



**Mac Schwager** (Member, IEEE) received the B.S. degree from Stanford University, Stanford, MA, USA, in 2000, and the M.S. and Ph.D. degrees from MIT, Cambridge, MA, USA, in 2005 and 2009, respectively, all in mechanical engineering.

He is currently an Associate Professor with the Aeronautics and Astronautics Department, Stanford University. He was a Postdoctoral Researcher working jointly with the GRASP Lab, University of Pennsylvania, and CSAIL, MIT, from 2010 to 2012, and was an Assistant Professor with Boston University from 2012 to 2015. His research interests include distributed algorithms for control, perception, and learning in groups of robots and animals.

Dr. Schwager was the recipient of the NSF CAREER award in 2014, the DARPA YFA in 2018, and a Google Faculty Research award in 2018.