Distributed Differential Dynamic Programming Architectures for Large-Scale Multi-Agent Control

Augustinos D. Saravanos, Yuichiro Aoyama, Hongchang Zhu and Evangelos A. Theodorou

***Abstract*—This paper proposes two decentralized multi-agent optimal control methods that combine the computational efficiency and scalability of Differential Dynamic Programming (DDP) and the distributed nature of the Alternating Direction Method of Multipliers (ADMM). The first one, Nested Distributed DDP (ND-DDP), is a three-level architecture which employs ADMM for consensus, an augmented Lagrangian layer for local constraints and DDP as the local optimizer. The second one, Merged Distributed DDP (MD-DDP), is a two-level architecture that addresses both consensus and local constraints with ADMM, further reducing computational complexity. Both frameworks are *fully decentralized* since all computations are parallelizable among the agents and only local communication is necessary. Simulation results that scale up to thousands of cars and hundreds of drones demonstrate the effectiveness of the algorithms. Superior scalability to large-scale systems against other DDP and sequential quadratic programming methods is also illustrated. Finally, hardware experiments on a multi-robot platform verify the applicability of the methods. A video with all results is provided in the supplementary material.**

***Index Terms*—Distributed robot systems, optimization and optimal control, multi-robot systems, swarms.**

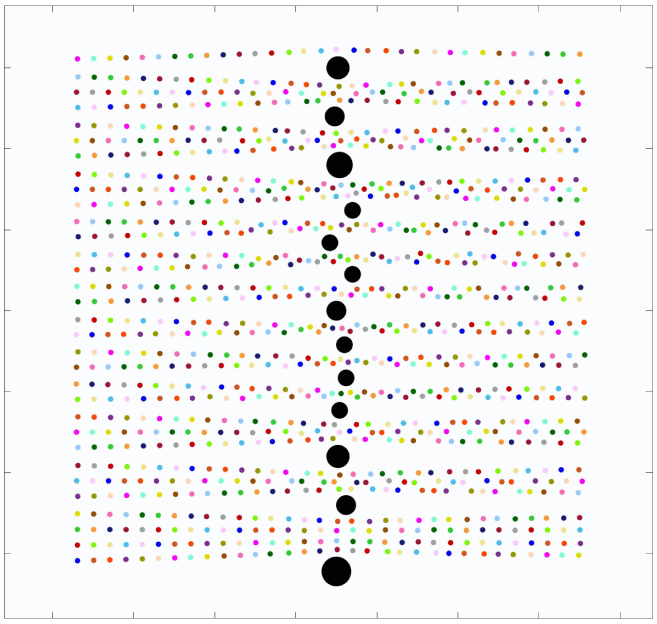

Fig. 1: A 1,024 cars fleet navigation problem in a multi-obstacle environment solved with MD-DDP.

I. Introduction

Multi-agent problems are emerging increasingly often in the robotics and control fields, with numerous significant applications such as guiding fleets of autonomous cars [1], [2], navigating formations of unmanned aerial vehicles (UAVs) [3], multi-robot motion coordination [4], air traffic management [5], swarm robotics [6] and so forth. There is a tendency that the size of such systems increases from a handful to large-scale teams of hundreds or even thousands of robots that are required to cooperatively achieve a common objective [7]–[9]. It is also desirable that these problems are solved optimally regarding the performance and control effort of the agents while ensuring their safe operation within their environment.

Such specifications can be captured intrinsically within a multi-agent optimal control setting [10], [11], where a set of agents would have to collectively minimize a global cost while satisfying constraints that might be coupled between the agents. Apart from optimal control approaches, there exists a variety of methods for addressing multi-robot control problems. For path planning and navigation, graph search-based [12], [13] and sampling-based [14], [15] algorithms have been considered the status quo during recent years. Other notable families of multi-robot control methods are found in game theory [16], potential field approaches [17], market-based methods [18], consensus-based algorithms [19], reinforcement learning [20], etc. Nevertheless, in this work, we opt to focus on optimal control approaches due to their generalizability for different types of costs, dynamics, actuation and safety constraints which is crucial for designing algorithms that are applicable to a wide variety of large-scale robotic systems.

A straightforward solution to address multi-agent optimal control is through centralized methods, where a powerful controller is assumed to perform all computations and communicate with all agents [21]. Unfortunately, this is often prohibitive for large-scale systems due to poor scalability with the increase of agents and potential communication limitations. As a consequence, there is a great necessity for designing fully decentralized optimal control approaches that enjoy computational and communication efficiency along with scalability to large-scale systems.

There exist two main classes of methods for solving nonlinear optimal control problems. The first one, referred to as direct trajectory optimization, relies on representing both the states and controls as optimization variables and solving the

This work was supported by the ARO Award #W911NF2010151 and the Georgia COVID relief fund. Augustinos D. Saravanos acknowledges financial support by the A. Onassis Foundation Scholarship. *(Corresponding author: Augustinos D. Saravanos)*

The authors are with the Daniel Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, North Avenue, Atlanta 30332, GA, USA (e-mail: asaravanos@gatech.edu; yaoyama3@gatech.edu; hozhu@gatech.edu; evangelos.theodorou@gatech.edu).

problem with an off-the-shelf nonlinear programming (NLP) algorithm such as sequential quadratic programming (SQP) [10], [22]. The second class entails the indirect methods, where only the controls are parametrized while the states are obtained through forward integration of the dynamics, alleviating the need for explicitly imposing them as constraints [11].

Differential Dynamic Programming (DDP) is a well-known second-order indirect method that was first proposed by Jacobson and Mayne in 1970 [23] and has found numerous successful robotics applications [24]–[27]. Its solution yields an optimal state trajectory along with locally optimal feed-forward and feedback control policies. It also includes the iterative linear quadratic regulator (iLQR) algorithm [28] as a special case if only the first-order approximations of the dynamics are considered. Some appealing features of DDP include its global convergence guarantees with a quadratic rate and its low computational and memory complexities [29], [30]. These characteristics have established DDP as one of the most scalable nonlinear optimal control methods [30], [31], available in the literature. Several extensions for handling control and/or state constraints with DDP can be found in [32]–[40]. Game theoretic DDP-based methods for multi-agent settings have been presented in [41], [42].

Although DDP is considered to be one of the most scalable optimal control approaches, yet its scalability would be limited if directly applied on high-dimensional control problems in a centralized fashion. Therefore, it is highly desirable to develop new distributed algorithms that retain the advantages of DDP and are inherently scalable for large-scale multi-robot control. Along this direction, but in a non-robotics setting, a preconditioned distributed DDP-based method was presented in [43], where inter-agent interactions only appear through the dynamics but not through inequality constraints that are typical in multi-robot systems.

An attractive approach adopted in this paper in order to overcome the scalability issue is to combine DDP with the Alternating Direction Method of Multipliers (ADMM). ADMM is an optimization method that has recently received significant attention because it can lead to algorithms with a distributed structure [44]. Several ADMM-based decentralized control methods have been proposed recently [45]–[52], demonstrating a significantly improved computational efficiency against centralized methods. Nevertheless, there has been very little work in proposing schemes that attempt to merge the well-appreciated attributes of both DDP and ADMM. In [53], [54], the authors provide multi-agent ADMM-based DDP algorithms that are still semi-centralized, while scalability results are limited to a few vehicles. In [55], a semi-centralized multi-target tracking method using iLQR and ADMM was proposed, yet being distributed only among the targets and not the agents. While these methods have shown promising results, they still require a central node for computational and communication purposes and have only demonstrated low scalability in practice.

The scope of the present work is to propose distributed architectures that thoroughly exploit the capabilities of combining DDP and ADMM, leading to fully decentralized algorithms that are applicable to large-scale multi-robot systems. In particular, we propose two novel distributed schemes:

1) **Nested Distributed DDP (ND-DDP)**, a three-level decentralized architecture that utilizes ADMM for enforcing consensus between all agents, an Augmented Largangian (AL) layer for satisfying local constraints and DDP as each agent's optimizer. This framework has a general form where any constrained DDP variant can be used instead of the AL-DDP combination.
2) **Merged Distributed DDP (MD-DDP)**, a two-level decentralized architecture which exploits the fact that both consensus and local constraints can be treated in the same level through ADMM, which further reduces computational complexity.

To additionally enhance the computational efficiency of each framework, we also propose the following algorithmic improvements. First, a Nesterov acceleration technique is utilized, leading to computationally faster variants of both methods. Subsequently, a decentralized adaptation scheme for the penalty parameters induced by ADMM, that is particularly tailored for multi-agent optimal control problems, is also introduced. Both ND-DDP and MD-DDP are extensively tested in simulation on various multi-vehicle and multi-UAV problems of an increasing scale. ND-DDP is successfully applied to systems with up to 256 cars (1,024 states) and 64 drones (768 states). With MD-DDP, an even higher scalability is achieved, solving problems with up to 4,096 cars (16,384 states) and 256 drones (3,072 states). To our best knowledge, ND-DDP and MD-DDP are the first fully decentralized DDP/iLQR-based methods that have been shown to scale for multi-robot systems of such a large scale. The superior scalability of ND-DDP and MD-DDP against centralized/semi-centralized DDP and centralized/decantrized SQP is also confirmed through computational time comparisons. Finally, to verify the applicability of the methods on actual systems, we also successfully employ them on the Robotarium platform [56] for various multi-robot tasks. These experiments also underline the advantage of DDP as a dynamic optimizer to provide control policies that consist of feed-forward and feedback terms, thus enhancing the robustness of the methods against model uncertainty.

The rest of the paper is organized as follows. In Section II, a brief overview of DDP and ADMM is provided. The multi-agent optimal control problem is formulated in Section III. In Section IV, the ND-DDP architecture is proposed, while the MD-DDP framework is introduced in Section V. In Section VI, the improved versions of the architectures are presented, along with additional details on the methods. In Section VII, the performance of the algorithms on multi-car and multi-UAV systems is demonstrated through simulations. Hardware experiments on a multi-robot platform are presented in Section VIII. Finally, a conclusion and an overview of future research directions are provided in Section IX.

II. Preliminaries

A. Notation

Throughout this paper, non-bold symbols are used for scalars $a \in \mathbb{R}$ and bold lowercase and uppercase symbols for vectors $\boldsymbol{a} \in \mathbb{R}^n$ and matrices $\boldsymbol{A} \in \mathbb{R}^{n \times m}$, respectively. With

$[\![a,b]\!]$, the integer interval $[a,b]\cap\mathbb{Z}$ is defined. Given vectors $\boldsymbol{a}_i,\ i\in[\![1,n]\!]$, we refer to their vertical concatenation as $\boldsymbol{a}=\left[\boldsymbol{a}_1;\ldots;\boldsymbol{a}_n\right]=\left[\{\boldsymbol{a}_i\}_{i\in[\![1,n]\!]}\right]$. A diagonal matrix made up of scalars $a_i, i\in[\![1,n]\!]$, is denoted with $\mathrm{diag}(a_1,\ldots,a_n)\in\mathbb{R}^{n\times n}$, while a block diagonal matrix constructed by $\boldsymbol{A}_i$, $i\in[\![1,n]\!]$, is denoted with $\mathrm{bdiag}(\boldsymbol{A}_1,\ldots,\boldsymbol{A}_n)$. The ℓ_2-norm of $\boldsymbol{a}=\begin{bmatrix}a_1&\ldots&a_n\end{bmatrix}\in\mathbb{R}^n$ is $\|\boldsymbol{a}\|_2=\sqrt{\boldsymbol{a}^{\mathrm{T}}\boldsymbol{a}}=\sqrt{\sum_{i=1}^n a_i^2}$. We also define the weighted norm $\|\boldsymbol{a}\|_{\boldsymbol{W}}=\|\boldsymbol{W}^{1/2}\boldsymbol{a}\|_2=\sqrt{\boldsymbol{a}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{a}}$ for any symmetric positive definite matrix $\boldsymbol{W}$. Furthermore, given a set $\mathcal{X}$, its cardinality is provided by $|\mathcal{X}|$. Finally, the projection of a vector $\boldsymbol{x}$ onto a set $\mathcal{C}$ is denoted with $\Pi_{\mathcal{C}}(\boldsymbol{x})$, while $\mathcal{I}_{\mathcal{C}}(\boldsymbol{x})$ defines an indicator function such that $\mathcal{I}_{\mathcal{C}}(\boldsymbol{x})=0$ if $\boldsymbol{x}\in\mathcal{C}$ and $\mathcal{I}_{\mathcal{C}}(\boldsymbol{x})=+\infty$, otherwise.

B. Differential Dynamic Programming

A brief introduction to DDP is initially provided in its well-known centralized/single-agent unconstrained form [23]. Let us consider the following discrete-time nonlinear dynamics

$$\boldsymbol{x}_{k+1}=\boldsymbol{f}(\boldsymbol{x}_k,\boldsymbol{u}_k),\quad \boldsymbol{x}_0:\text{given}, \tag{1}$$

where $\boldsymbol{x}_k\in\mathbb{R}^p$, $\boldsymbol{u}_k\in\mathbb{R}^q$ denote the state and control at time instant k, respectively, and $\boldsymbol{f}:\mathbb{R}^p\times\mathbb{R}^q\to\mathbb{R}^p$ is the dynamics function. Given a finite time horizon K, the state and control trajectories are denoted with $\boldsymbol{x}=\left[\boldsymbol{x}_0;\ldots;\boldsymbol{x}_K\right]$ and $\boldsymbol{u}=\left[\boldsymbol{u}_0;\ldots;\boldsymbol{u}_{K-1}\right]$. The finite-horizon discrete-time optimal control problem can be formulated as finding the optimal control sequence $\boldsymbol{u}^*$ such that

$$\boldsymbol{u}^*=\arg\min_{\boldsymbol{u}} J(\boldsymbol{x},\boldsymbol{u})=\sum_{k=0}^{K-1}\ell(\boldsymbol{x}_k,\boldsymbol{u}_k)+\phi(\boldsymbol{x}_K) \tag{2a}$$

$$\text{s.t.}\quad \boldsymbol{x}_{k+1}=\boldsymbol{f}(\boldsymbol{x}_k,\boldsymbol{u}_k),\quad \boldsymbol{x}_0:\text{given}, \tag{2b}$$

with $J:\mathbb{R}^{(K+1)p}\times\mathbb{R}^{Kq}\to\mathbb{R}$, $\ell:\mathbb{R}^p\times\mathbb{R}^q\to\mathbb{R}$ and $\phi:\mathbb{R}^p\to\mathbb{R}$ being the total, running and terminal costs, respectively. Both the dynamics and the cost are assumed to be twice differentiable. Problem (2) can be addressed with dynamic programming [57] since, given $\boldsymbol{x}_k$ at time k, the optimality of the future controls is independent of the past states and controls. The value function that describes the optimal "cost-to-go" starting from $\boldsymbol{x}_k$ is provided by:

$$V_k(\boldsymbol{x}_k):=\min_{\boldsymbol{u}_k,\ldots,\boldsymbol{u}_{K-1}}\sum_{\kappa=k}^{K-1}\ell(\boldsymbol{x}_\kappa,\boldsymbol{u}_\kappa)+\phi(\boldsymbol{x}_K). \tag{3}$$

Hence, solving (2) is equivalent with finding a control sequence $\boldsymbol{u}$ such that $J(\boldsymbol{x},\boldsymbol{u})=V_0(\boldsymbol{x}_0)$. Bellman's Principle of Optimality [57] states that $V_k(\boldsymbol{x}_k)$ can be found through the following backpropagation rule

$$V_k(\boldsymbol{x}_k)=\min_{\boldsymbol{u}_k}Q_k(\boldsymbol{x}_k,\boldsymbol{u}_k), \tag{4a}$$

$$Q_k(\boldsymbol{x}_k,\boldsymbol{u}_k)=\ell(\boldsymbol{x}_k,\boldsymbol{u}_k)+V_{k+1}(\boldsymbol{x}_{k+1}), \tag{4b}$$

with terminal condition $V_K(\boldsymbol{x}_K)=\phi(\boldsymbol{x}_K)$.

DDP is a second-order method that solves problem (2) locally through (4a), by taking an approximation of Q_k. The algorithm operates in an iterative backward and forward pass fashion. During the *backward* pass, both sides of (4a) are expanded around some nominal trajectories $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{u}}$. In particular, by defining the deviations $\delta\boldsymbol{x}_k=\boldsymbol{x}_k-\bar{\boldsymbol{x}}_k$, $\delta\boldsymbol{u}_k=\boldsymbol{u}_k-\bar{\boldsymbol{u}}_k$, the quadratic expansion of Q_k is given by

$$\begin{aligned}Q_k(\boldsymbol{x}_k,\boldsymbol{u}_k)\approx{}&Q_{\boldsymbol{x},k}^{\mathrm{T}}\delta\boldsymbol{x}_k+Q_{\boldsymbol{u},k}^{\mathrm{T}}\delta\boldsymbol{u}_k+\frac{1}{2}\delta\boldsymbol{x}_k^{\mathrm{T}}Q_{\boldsymbol{xx},k}\delta\boldsymbol{x}_k\\&+\delta\boldsymbol{x}_kQ_{\boldsymbol{xu},k}\delta\boldsymbol{u}_k+\frac{1}{2}\delta\boldsymbol{u}_k^{\mathrm{T}}Q_{\boldsymbol{uu},k}\delta\boldsymbol{u}_k,\end{aligned} \tag{5}$$

with

$$Q_{\boldsymbol{xx},k}=\ell_{\boldsymbol{xx}}+\boldsymbol{f}_{\boldsymbol{x}}^{\mathrm{T}}V_{\boldsymbol{xx},k+1}\boldsymbol{f}_{\boldsymbol{x}}+V_{\boldsymbol{x},k+1}\cdot\boldsymbol{f}_{\boldsymbol{xx}}, \tag{6a}$$

$$Q_{\boldsymbol{xu},k}=\ell_{\boldsymbol{xu}}+\boldsymbol{f}_{\boldsymbol{x}}^{\mathrm{T}}V_{\boldsymbol{xx},k+1}\boldsymbol{f}_{\boldsymbol{u}}+V_{\boldsymbol{x},k+1}\cdot\boldsymbol{f}_{\boldsymbol{xu}}, \tag{6b}$$

$$Q_{\boldsymbol{uu},k}=\ell_{\boldsymbol{uu}}+\boldsymbol{f}_{\boldsymbol{u}}^{\mathrm{T}}V_{\boldsymbol{xx},k+1}\boldsymbol{f}_{\boldsymbol{u}}+V_{\boldsymbol{x},k+1}\cdot\boldsymbol{f}_{\boldsymbol{uu}}, \tag{6c}$$

$$Q_{\boldsymbol{x},k}=\ell_{\boldsymbol{x}}+\boldsymbol{f}_{\boldsymbol{x}}^{\mathrm{T}}V_{\boldsymbol{x},k+1}, \tag{6d}$$

$$Q_{\boldsymbol{u},k}=\ell_{\boldsymbol{u}}+\boldsymbol{f}_{\boldsymbol{u}}^{\mathrm{T}}V_{\boldsymbol{x},k+1}, \tag{6e}$$

where the cost and dynamics derivatives are evaluated at $\bar{\boldsymbol{x}}_k$ and $\bar{\boldsymbol{u}}_k$. The last terms in (6a)-(6c) are vector-tensor products that appear in DDP if the second-order approximation of the dynamics is considered. By only taking the first-order dynamics expansion, these terms are omitted and DDP coincides with iLQR [28]. Using (6), the minimization of the RHS of (4a) yields the following optimal control deviations

$$\delta\boldsymbol{u}_k^*=\boldsymbol{k}_k+\boldsymbol{K}_k\delta\boldsymbol{x}_k, \tag{7}$$

with $\boldsymbol{k}_k=-Q_{\boldsymbol{uu},k}^{-1}Q_{\boldsymbol{u},k}$ and $\boldsymbol{K}_k=-Q_{\boldsymbol{uu},k}^{-1}Q_{\boldsymbol{ux},k}$ being the feed-forward and feedback gains, respectively. By also expanding V_k quadratically, (4a) leads to the following backpropagation rules

$$V_k=-\frac{1}{2}Q_{\boldsymbol{u},k}^{\mathrm{T}}Q_{\boldsymbol{uu},k}^{-1}Q_{\boldsymbol{u},k}, \tag{8a}$$

$$V_{\boldsymbol{x},k}=Q_{\boldsymbol{x},k}-Q_{\boldsymbol{ux},k}^{\mathrm{T}}Q_{\boldsymbol{uu},k}^{-1}Q_{\boldsymbol{u},k}, \tag{8b}$$

$$V_{\boldsymbol{xx},k}=Q_{\boldsymbol{xx},k}-Q_{\boldsymbol{ux},k}^{\mathrm{T}}Q_{\boldsymbol{uu},k}^{-1}Q_{\boldsymbol{ux},k}, \tag{8c}$$

with $V_K=\phi(\boldsymbol{x}_K)$, $V_{\boldsymbol{x},K}=\phi_{\boldsymbol{x}}(\boldsymbol{x}_K)$ and $V_{\boldsymbol{xx},K}=\phi_{\boldsymbol{xx}}(\boldsymbol{x}_K)$.

After completing the backward pass, the new trajectories $\boldsymbol{x}$ and $\boldsymbol{u}$ are obtained during the *forward* pass as follows

$$\boldsymbol{u}_k=\bar{\boldsymbol{u}}_k+\boldsymbol{k}_k+\boldsymbol{K}_k(\boldsymbol{x}_k-\bar{\boldsymbol{x}}_k), \tag{9a}$$

$$\boldsymbol{x}_{k+1}=\boldsymbol{f}(\boldsymbol{x}_k,\boldsymbol{u}_k),\quad \boldsymbol{x}_0:\text{given}. \tag{9b}$$

These new trajectories will be used as the nominal ones during the next backward pass, and so on. This iterative procedure continues until a predefined termination criterion is satisfied.

While the original DDP method was developed for unconstrained optimal control problems, several variations have been proposed for handling control and/or state constraints such as [32]–[38]. Out of them, methods that incorporate constraints through an AL on the cost have been shown to be particularly effective [37], [38]. In this paper, we will use the recently suggested AL-DDP combination [38] as a baseline for addressing constrained optimal control problems with DDP.

C. Alternating Direction Method of Multipliers

Subsequently, we provide a brief overview of ADMM. For more details, the reader is referred to [44]. The standard (two-block) version of ADMM considers an optimization problem of the following form

$$\min_{\boldsymbol{x},\boldsymbol{z}}\ f(\boldsymbol{x})+g(\boldsymbol{z})\quad\text{s.t. }\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{z}=\boldsymbol{c}, \tag{10}$$

where $\boldsymbol{x}\in\mathbb{R}^p$, $\boldsymbol{z}\in\mathbb{R}^q$ are the primal variables, $f:\mathbb{R}^p\to\mathbb{R}$, $g:\mathbb{R}^q\to\mathbb{R}$, $\boldsymbol{A}\in\mathbb{R}^{r\times p}$, $\boldsymbol{B}\in\mathbb{R}^{r\times q}$ and $\boldsymbol{c}\in\mathbb{R}^r$. For problem

(10), the AL is given by

$$\mathcal{L}(\boldsymbol{x},\boldsymbol{z},\boldsymbol{\lambda}) = f(\boldsymbol{x}) + g(\boldsymbol{z}) + \boldsymbol{\lambda}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{z}-\boldsymbol{c}) + \frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{z}-\boldsymbol{c}\|_2^2,$$

where $\boldsymbol{\lambda} \in \mathbb{R}^r$ is the dual variable for the constraint $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c} = 0$ and $\rho > 0$ is a penalty parameter. Classical ADMM consists of the following sequential updates

$$\boldsymbol{x}^{n+1} = \arg\min_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{z}^n, \boldsymbol{\lambda}^n) \tag{11a}$$

$$\boldsymbol{z}^{n+1} = \arg\min_{\boldsymbol{z}} \mathcal{L}(\boldsymbol{x}^{n+1}, \boldsymbol{z}, \boldsymbol{\lambda}^n) \tag{11b}$$

$$\boldsymbol{\lambda}^{n+1} = \boldsymbol{\lambda}^n + \rho(\boldsymbol{A}\boldsymbol{x}^{n+1} + \boldsymbol{B}\boldsymbol{z}^{n+1} - \boldsymbol{c}), \tag{11c}$$

where n denotes the algorithm iteration. The method can be extended to a multi-block version with variables $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, $N \geq 3$, that are updated in a sequential manner [58], [59].

III. Multi-Agent Optimal Control Problem Formulation

Let us consider a team of M agents given by the set $\mathcal{V} = \{1, \ldots, M\}$. We first introduce the notion of *neighbors*.

Definition 1 (Neighborhood Set): The "neighborhood" set $\mathcal{N}_i \subseteq \mathcal{V}$ of an agent $i \in \mathcal{V}$ is the set that contains all agents $j \in \mathcal{V}$ that are neighbors of i (including i).

Definition 2 (Neighbor-of Set): The "neighbor-of" set $\mathcal{P}_i \subseteq \mathcal{V}$ of an agent $i \in \mathcal{V}$ is defined as $\mathcal{P}_i = \{j \in \mathcal{V} : i \in \mathcal{N}_j\}$. In other words, it is the set that contains all agents $j \in \mathcal{V}$ that consider i as a neighbor.

Assumption 1: The sets $\mathcal{N}_i,\ i \in \mathcal{V}$, are time-invariant. It follows that $\mathcal{P}_i,\ i \in \mathcal{V}$, are also time-invariant.

In a multi-robot setting, we would typically have $j \in \mathcal{N}_i$ if agent j is within a close distance from i. Note that there is no assumption of mutual neighbors, i.e., it is not required that $\mathcal{N}_i = \mathcal{P}_i$. Furthermore, we only require local communication between the agents through the following assumption.

Assumption 2: Each agent $i \in \mathcal{V}$ is able to exchange information only with agents $j \in \mathcal{N}_i \cup \mathcal{P}_i$.

The dynamics of the i-th agent are provided by the following discrete-time nonlinear equations

$$\boldsymbol{x}_{i,k+1} = \boldsymbol{f}_i(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}), \quad \boldsymbol{x}_{i,0} : \text{given}, \tag{12}$$

where $\boldsymbol{x}_{i,k} \in \mathbb{R}^{p_i}$, $\boldsymbol{u}_{i,k} \in \mathbb{R}^{q_i}$ are the state and control input of agent i at time k. Let $\boldsymbol{x}_i = [\boldsymbol{x}_{i,0}; \ldots; \boldsymbol{x}_{i,K}]$ and $\boldsymbol{u}_i = [\boldsymbol{u}_{i,0}; \ldots; \boldsymbol{u}_{i,K-1}]$ be the state and control trajectories of agent i over a time horizon K. The global cost function that all agents aim to collectively minimize is

$$J = \sum_{i=1}^{M} J_i(\boldsymbol{x}_i, \boldsymbol{u}_i) \tag{13}$$

where each local component $J_i : \mathbb{R}^{(K+1)p_i} \times \mathbb{R}^{Kq_i} \to \mathbb{R}$ has the form

$$J_i = \sum_{k=0}^{K-1} \Big[\ell_i(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k})\Big] + \phi_i(\boldsymbol{x}_{i,K}), \tag{14}$$

with $\ell_i : \mathbb{R}^{p_i} \times \mathbb{R}^{q_i} \to \mathbb{R}$ and $\phi_i : \mathbb{R}^{p_i} \to \mathbb{R}$ being the running and terminal costs, respectively. All agents are subject to the following *single-agent* control and state constraints

$$\boldsymbol{b}_{i,k}(\boldsymbol{u}_{i,k}) \leq 0,\ k \in [\![0, K-1]\!], \tag{15}$$

$$\boldsymbol{g}_{i,k}(\boldsymbol{x}_{i,k}) \leq 0,\ k \in [\![0, K]\!]. \tag{16}$$

In multi-robot problems, the former usually correspond to actuation limitations, while the latter often represent position, velocity, obstacle avoidance constraints, etc. Finally, the following *inter-agent* state constraints must also be satisfied between any agent $i \in \mathcal{V}$ and its neighbors,

$$\boldsymbol{h}_{ij,k}(\boldsymbol{x}_{i,k}, \boldsymbol{x}_{j,k}) \leq 0,\ k \in [\![0, K]\!],\ j \in \mathcal{N}_i \backslash \{i\}. \tag{17}$$

Such constraints could enforce collision avoidance or connectivity maintenance between neighbors. Note that the use of the latter supports introducing Assumption 1 since neighboring agents would always stay within a close distance. Consequently, we proceed with formulating the multi-agent optimal control problem.

Problem 1 (Multi-Agent Optimal Control Problem): Find the optimal control sequences $\boldsymbol{u}_i^*,\ \forall i \in \mathcal{V}$, such that

$$\begin{aligned} \{\boldsymbol{u}_i^*\}_{i\in\mathcal{V}} = \arg\min & \sum_{i=1}^{M} J_i(\boldsymbol{x}_i, \boldsymbol{u}_i) \\ \text{s.t.} \quad & \boldsymbol{x}_{i,k+1} = \boldsymbol{f}_i(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}),\ \boldsymbol{x}_{i,0} : \text{given}, \\ & \boldsymbol{b}_{i,k}(\boldsymbol{u}_{i,k}) \leq 0,\ \boldsymbol{g}_{i,k}(\boldsymbol{x}_{i,k}) \leq 0, \\ & \boldsymbol{h}_{ij,k}(\boldsymbol{x}_{i,k}, \boldsymbol{x}_{j,k}) \leq 0,\ j \in \mathcal{N}_i \backslash \{i\},\ i \in \mathcal{V}. \end{aligned}$$

IV. Nested Distributed DDP

The first proposed architecture (Fig. 2) uses an ADMM layer for enforcing a consensus between the decisions of all agents (Level 1). Next, each agent's neighborhood constraints are captured through a local AL layer (Level 2). Finally, the resulting problems are solved by each agent through DDP (Level 3). Due to the "nested" structure of this framework, we refer to it as Nested Distributed DDP (ND-DDP).

A. Problem Transformation

Prior to the method derivation, it is necessary that Problem 1 is transformed to an equivalent form that is suitable for distributed optimization. The coupling that prohibits directly solving it in a decentralized manner is the one induced by the inter-agent constraints (17). To address this, we propose that each agent contains copy variables regarding the states and controls of their neighbors. In other words, we define the variables $\boldsymbol{x}_{j,k}^i \in \mathbb{R}^{p_j},\ \boldsymbol{u}_{j,k}^i \in \mathbb{R}^{q_j},\ j \in \mathcal{N}_i,\ i \in \mathcal{V}$, where each $\boldsymbol{x}_{j,k}^i$ (or $\boldsymbol{u}_{j,k}^i$) refers to what would be safe for agent j from the *point of view* of agent i. Of course, the variables $\boldsymbol{x}_{i,k}^i$ and $\boldsymbol{u}_{i,k}^i$ coincide with $\boldsymbol{x}_{i,k}$ and $\boldsymbol{u}_{i,k}$, respectively. Thus, the following augmented state and control variables can be defined for each agent:

$$\boldsymbol{x}_{i,k}^{\mathrm{a}} = [\{\boldsymbol{x}_{j,k}^i\}_{j\in\mathcal{N}_i}] \in \mathbb{R}^{\tilde{p}_i},\ \tilde{p}_i = \sum_{j\in\mathcal{N}_i} p_j, \tag{18}$$

$$\boldsymbol{u}_{i,k}^{\mathrm{a}} = [\{\boldsymbol{u}_{j,k}^i\}_{j\in\mathcal{N}_i}] \in \mathbb{R}^{\tilde{q}_i},\ \tilde{q}_i = \sum_{j\in\mathcal{N}_i} q_j. \tag{19}$$

The trajectories $\boldsymbol{x}_j^i, \boldsymbol{u}_j^i, \boldsymbol{x}_i^{\mathrm{a}}, \boldsymbol{u}_i^{\mathrm{a}}$ are defined as $\boldsymbol{x}_i, \boldsymbol{u}_i$ earlier. Thanks to introducing the copy variables, the inter-agent

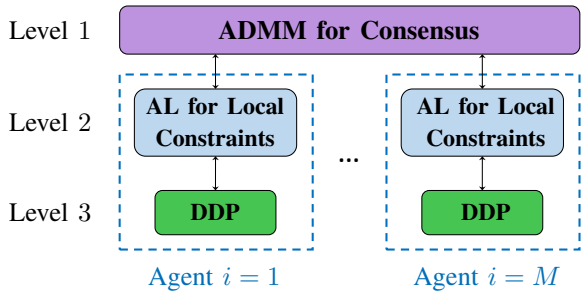

Fig. 2: The ND-DDP ("nested") architecture. Level 1: ADMM for consensus between different agents; Level 2: AL for local constraints; Level 3: DDP.

constraints (17) can be rewritten from the perspective of each agent i as $\boldsymbol{h}_{ij,k}(\boldsymbol{x}_{i,k}, \boldsymbol{x}^i_{j,k}) \leq 0$, or more compactly as

$$\boldsymbol{h}^{\mathrm{a}}_{i,k}(\boldsymbol{x}^{\mathrm{a}}_{i,k}) \leq 0,\ i \in \mathcal{V}, \tag{20}$$

where $\boldsymbol{h}^{\mathrm{a}}_{i,k}(\boldsymbol{x}^{\mathrm{a}}_{i,k}) = \big[\{\boldsymbol{h}_{ij,k}(\boldsymbol{x}_{i,k}, \boldsymbol{x}^i_{j,k})\}_{j\in\mathcal{N}_i\setminus\{i\}}\big]$.

Nevertheless, since there might be multiple variables referring to the same agent, it becomes necessary to enforce a consensus between such variables. To achieve this, the following global state and control variables are also introduced

$$\boldsymbol{z}_k = [\{\boldsymbol{z}_{i,k}\}_{i\in\mathcal{V}}] \in \mathbb{R}^p,\ p = \sum_{i\in\mathcal{V}} p_i,\ k \in [\![0, K]\!], \tag{21}$$

$$\boldsymbol{w}_k = [\{\boldsymbol{w}_{i,k}\}_{i\in\mathcal{V}}] \in \mathbb{R}^q,\ q = \sum_{i\in\mathcal{V}} q_i,\ k \in [\![0, K-1]\!], \tag{22}$$

with the trajectories $\boldsymbol{z}$, $\boldsymbol{w}$, $\boldsymbol{z}_i$ and $\boldsymbol{w}_i$ defined accordingly. Thus, we can impose the following consensus constraints

$$\boldsymbol{x}^i_j = \boldsymbol{z}_j,\ \boldsymbol{u}^i_j = \boldsymbol{w}_j,\ j \in \mathcal{N}_i,\ i \in \mathcal{V}, \tag{23}$$

which can be written in a more compact form as

$$\boldsymbol{x}^{\mathrm{a}}_i = \boldsymbol{z}^{\mathrm{a}}_i,\ \boldsymbol{u}^{\mathrm{a}}_i = \boldsymbol{w}^{\mathrm{a}}_i,\ i \in \mathcal{V}, \tag{24}$$

with $\boldsymbol{z}^{\mathrm{a}}_{i,k} = \big[\{\boldsymbol{z}_{j,k}\}_{j\in\mathcal{N}_i}\big] \in \mathbb{R}^{\tilde{p}_i}$ and $\boldsymbol{w}^{\mathrm{a}}_{i,k} = \big[\{\boldsymbol{w}_{j,k}\}_{j\in\mathcal{N}_i}\big] \in \mathbb{R}^{\tilde{q}_i}$. Let us also introduce the augmented dynamics $\boldsymbol{x}^{\mathrm{a}}_{i,k+1} = \boldsymbol{f}^{\mathrm{a}}_i(\boldsymbol{x}^{\mathrm{a}}_{i,k}, \boldsymbol{u}^{\mathrm{a}}_{i,k})$ and constraints $\boldsymbol{b}^{\mathrm{a}}_{i,k}(\boldsymbol{u}^{\mathrm{a}}_{i,k}) \leq 0$, $\boldsymbol{g}^{\mathrm{a}}_{i,k}(\boldsymbol{x}^{\mathrm{a}}_{i,k}) \leq 0$, accordingly. Finally, we suggest the new local cost functions:

$$J^{\mathrm{a}}_i(\boldsymbol{x}^{\mathrm{a}}_i, \boldsymbol{u}^{\mathrm{a}}_i) = \sum_{j\in\mathcal{N}_i} \frac{1}{|\mathcal{P}_j|} J_j(\boldsymbol{x}^i_j, \boldsymbol{u}^i_j), \tag{25}$$

for every agent $i \in \mathcal{V}$. Therefore, Problem 1 can be transformed to the following equivalent form.

Problem 2 (Multi-Agent Optimal Control Problem II): Find the optimal control sequences $\boldsymbol{u}^{\mathrm{a}*}_i,\ \forall i \in \mathcal{V}$, such that

$$\begin{aligned}
&\{\boldsymbol{u}^{\mathrm{a}*}_i\}_{i\in\mathcal{V}} = \arg\min \sum_{i=1}^{M} J^{\mathrm{a}}_i(\boldsymbol{x}^{\mathrm{a}}_i, \boldsymbol{u}^{\mathrm{a}}_i)\\
\text{s.t.}\quad &\boldsymbol{x}^{\mathrm{a}}_{i,k+1} = \boldsymbol{f}^{\mathrm{a}}_i(\boldsymbol{x}^{\mathrm{a}}_{i,k}, \boldsymbol{u}^{\mathrm{a}}_{i,k}),\ \boldsymbol{x}^{\mathrm{a}}_{i,0} : \text{given},\\
&\boldsymbol{b}^{\mathrm{a}}_{i,k}(\boldsymbol{u}^{\mathrm{a}}_{i,k}) \leq 0,\ \boldsymbol{g}^{\mathrm{a}}_{i,k}(\boldsymbol{x}^{\mathrm{a}}_{i,k}) \leq 0,\ \boldsymbol{h}^{\mathrm{a}}_{i,k}(\boldsymbol{x}^{\mathrm{a}}_{i,k}) \leq 0,\\
&\boldsymbol{x}^{\mathrm{a}}_i = \boldsymbol{z}^{\mathrm{a}}_i,\ \boldsymbol{u}^{\mathrm{a}}_i = \boldsymbol{w}^{\mathrm{a}}_i,\ i \in \mathcal{V}.
\end{aligned}$$

Note that in (25), each J_j is multiplied with a factor $1/|\mathcal{P}_j|$ so that the cost functions in Problems 1 and 2 are equivalent.

B. Method Derivation

Subsequently, the derivation of the ND-DDP method is presented. Let us start with rewriting Problem 2 by replacing part of the constraints with their indicator functions as follows

$$\begin{aligned}
\min \sum_{i=1}^{M} &J^{\mathrm{a}}_i(\boldsymbol{x}^{\mathrm{a}}_i, \boldsymbol{u}^{\mathrm{a}}_i) + \mathcal{I}_{\boldsymbol{f}^{\mathrm{a}}_i}(\boldsymbol{x}^{\mathrm{a}}_i, \boldsymbol{u}^{\mathrm{a}}_i) + \mathcal{I}_{\boldsymbol{b}^{\mathrm{a}}_i}(\boldsymbol{u}^{\mathrm{a}}_i)\\
&+ \mathcal{I}_{\boldsymbol{g}^{\mathrm{a}}_i}(\boldsymbol{x}^{\mathrm{a}}_i) + \mathcal{I}_{\boldsymbol{h}^{\mathrm{a}}_i}(\boldsymbol{x}^{\mathrm{a}}_i)\\
\text{s.t.}\quad &\boldsymbol{x}^{\mathrm{a}}_i = \boldsymbol{z}^{\mathrm{a}}_i,\ \boldsymbol{u}^{\mathrm{a}}_i = \boldsymbol{w}^{\mathrm{a}}_i,\ i \in \mathcal{V}.
\end{aligned} \tag{26}$$

The AL for problem (26) can be formulated as

$$\begin{aligned}
\mathcal{L} = \sum_{i=1}^{M} &J^{\mathrm{a}}_i(\boldsymbol{x}^{\mathrm{a}}_i, \boldsymbol{u}^{\mathrm{a}}_i) + \mathcal{I}_{\boldsymbol{f}^{\mathrm{a}}_i}(\boldsymbol{x}^{\mathrm{a}}_i, \boldsymbol{u}^{\mathrm{a}}_i) + \mathcal{I}_{\boldsymbol{b}^{\mathrm{a}}_i}(\boldsymbol{u}^{\mathrm{a}}_i)\\
&+ \mathcal{I}_{\boldsymbol{g}^{\mathrm{a}}_i}(\boldsymbol{x}^{\mathrm{a}}_i) + \boldsymbol{\lambda}^{\mathrm{T}}_i(\boldsymbol{x}^{\mathrm{a}}_i - \boldsymbol{z}^{\mathrm{a}}_i) + \boldsymbol{y}^{\mathrm{T}}_i(\boldsymbol{u}^{\mathrm{a}}_i - \boldsymbol{w}^{\mathrm{a}}_i)\\
&+ \frac{\rho}{2}\|\boldsymbol{x}^{\mathrm{a}}_i - \boldsymbol{z}^{\mathrm{a}}_i\|^2_2 + \frac{\mu}{2}\|\boldsymbol{u}^{\mathrm{a}}_i - \boldsymbol{w}^{\mathrm{a}}_i\|^2_2,
\end{aligned} \tag{27}$$

where $\boldsymbol{\lambda}_i, \boldsymbol{y}_i$ are the corresponding dual variables of the consensus constraints and $\rho, \mu > 0$ are penalty parameters. Next, the ADMM update rules are derived. In the following, ADMM iterations are denoted with n.

Optimization Step 1: Local AL-DDP Updates. In the first block of updates, the AL (27) is minimized w.r.t. the local variables $\boldsymbol{x}^{\mathrm{a}}_i, \boldsymbol{u}^{\mathrm{a}}_i$, i.e.,

$$\{\boldsymbol{x}^{\mathrm{a}}_i, \boldsymbol{u}^{\mathrm{a}}_i\}^{n+1} = \arg\min \mathcal{L}(\boldsymbol{x}^{\mathrm{a}}_i, \boldsymbol{u}^{\mathrm{a}}_i, \boldsymbol{z}^n, \boldsymbol{w}^n, \boldsymbol{\lambda}^n_i, \boldsymbol{y}^n_i),$$

for all agents $i \in \mathcal{V}$. This results to the following M subproblems, that can be solved in parallel from all agents

$$\begin{aligned}
\{\boldsymbol{x}^{\mathrm{a}}_i, \boldsymbol{u}^{\mathrm{a}}_i\}^{n+1} = \arg\min &\sum_{k=0}^{K-1} \Big[\hat{\ell}_i(\boldsymbol{x}^{\mathrm{a}}_{i,k}, \boldsymbol{u}^{\mathrm{a}}_{i,k})\Big] + \hat{\phi}_i(\boldsymbol{x}^{\mathrm{a}}_{i,K}),\\
\text{s.t.}\quad &\boldsymbol{x}^{\mathrm{a}}_{i,k+1} = \boldsymbol{f}^{\mathrm{a}}_{i,k}(\boldsymbol{x}^{\mathrm{a}}_{i,k}, \boldsymbol{u}^{\mathrm{a}}_{i,k}),\ \boldsymbol{x}^{\mathrm{a}}_{i,0} : \text{given},\\
&\boldsymbol{b}^{\mathrm{a}}_{i,k}(\boldsymbol{u}^{\mathrm{a}}_{i,k}) \leq 0,\ \boldsymbol{g}^{\mathrm{a}}_{i,k}(\boldsymbol{x}^{\mathrm{a}}_{i,k}) \leq 0,\ \boldsymbol{h}^{\mathrm{a}}_{i,k}(\boldsymbol{x}^{\mathrm{a}}_{i,k}) \leq 0,
\end{aligned} \tag{28}$$

with the running and terminal costs being

$$\begin{aligned}
\hat{\ell}_i(\boldsymbol{x}^{\mathrm{a}}_{i,k}, \boldsymbol{u}^{\mathrm{a}}_{i,k}) = &\sum_{j\in\mathcal{N}_i} \frac{1}{|\mathcal{P}_j|} \ell_j(\boldsymbol{x}^i_{j,k}, \boldsymbol{u}^i_{j,k})\\
&+ \frac{\rho}{2}\left\|\boldsymbol{x}^{\mathrm{a}}_{i,k} - \boldsymbol{z}^{\mathrm{a}}_{i,k} + \frac{\boldsymbol{\lambda}_{i,k}}{\rho}\right\|^2_2 + \frac{\mu}{2}\left\|\boldsymbol{u}^{\mathrm{a}}_{i,k} - \boldsymbol{w}^{\mathrm{a}}_{i,k} + \frac{\boldsymbol{y}_{i,k}}{\mu}\right\|^2_2,\\
\hat{\phi}_i(\boldsymbol{x}^{\mathrm{a}}_{i,K}) = &\sum_{j\in\mathcal{N}_i} \frac{1}{|\mathcal{P}_j|} \phi_j(\boldsymbol{x}^i_{j,K}) + \frac{\rho}{2}\left\|\boldsymbol{x}^{\mathrm{a}}_{i,K} - \boldsymbol{z}^{\mathrm{a}}_{i,K} + \frac{\boldsymbol{\lambda}_{i,K}}{\rho}\right\|^2_2.
\end{aligned}$$

Each of these subproblems is solved by incorporating the local constraints into the cost through an AL approach as follows

$$\begin{aligned}
\{\boldsymbol{x}^{\mathrm{a}}_i, \boldsymbol{u}^{\mathrm{a}}_i\}^{n+1} = \arg\min &\sum_{k=0}^{K-1} \Big[\hat{\ell}_i(\boldsymbol{x}^{\mathrm{a}}_{i,k}, \boldsymbol{u}^{\mathrm{a}}_{i,k})\Big] + \hat{\phi}_i(\boldsymbol{x}^{\mathrm{a}}_{i,K})\\
&+ \sum_{k=0}^{K} C_{i,k}(\boldsymbol{x}^{\mathrm{a}}_{i,k}, \boldsymbol{u}^{\mathrm{a}}_{i,k})\\
\text{s.t.}\quad &\boldsymbol{x}^{\mathrm{a}}_{i,k+1} = \boldsymbol{f}^{\mathrm{a}}_{i,k}(\boldsymbol{x}^{\mathrm{a}}_{i,k}, \boldsymbol{u}^{\mathrm{a}}_{i,k}),\ \boldsymbol{x}^{\mathrm{a}}_{i,0} : \text{given},
\end{aligned} \tag{29}$$

where

$$\begin{aligned}
C_{i,k}(\boldsymbol{x}^{\mathrm{a}}_{i,k}, \boldsymbol{u}^{\mathrm{a}}_{i,k}) &= P\big(\boldsymbol{w}_{i,k}, \beta_{i,k}, \boldsymbol{s}^{\mathrm{a}}_{i,k}(\boldsymbol{x}^{\mathrm{a}}_{i,k}, \boldsymbol{u}^{\mathrm{a}}_{i,k})\big),\\
\boldsymbol{s}^{\mathrm{a}}_{i,k}(\boldsymbol{x}^{\mathrm{a}}_{i,k}, \boldsymbol{u}^{\mathrm{a}}_{i,k}) &= \big[\boldsymbol{b}^{\mathrm{a}}_{i,k}(\boldsymbol{u}^{\mathrm{a}}_{i,k});\ \boldsymbol{g}^{\mathrm{a}}_{i,k}(\boldsymbol{x}^{\mathrm{a}}_{i,k});\ \boldsymbol{h}^{\mathrm{a}}_{i,k}(\boldsymbol{x}^{\mathrm{a}}_{i,k})\big],
\end{aligned}$$

and $\boldsymbol{w}_{i,k}$, $\beta_{i,k}$ are the Lagrange multipliers and penalty parameters for the constraint $\boldsymbol{s}^{\mathrm{a}}_{i,k}(\boldsymbol{x}^{\mathrm{a}}_{i,k}, \boldsymbol{u}^{\mathrm{a}}_{i,k}) \leq 0$, respectively. Details on the form of $P(\cdot)$ and on the updates of $\boldsymbol{w}_{i,k}$, $\beta_{i,k}$ are provided in Appendix A. During each local AL iteration $l = 1, \ldots, L$, problem (29) is solved with DDP. The backward

Algorithm 1 Nested Distributed DDP (ND-DDP)

1: $\boldsymbol{x}_i', \boldsymbol{u}_i' \leftarrow$ Solve single-agent problems in parallel $\forall\ i \in \mathcal{V}$.
2: *Each agent* $i \in \mathcal{V}$ *receives* $\boldsymbol{x}_j', \boldsymbol{u}_j'$ *from all* $j \in \mathcal{N}_i \backslash \{i\}$.
3: **Initialize:** $\boldsymbol{x}_i^{\mathrm{a}} \leftarrow [\{\boldsymbol{x}_j'\}_{j\in\mathcal{N}_i}]$, $\boldsymbol{u}_i^{\mathrm{a}} \leftarrow [\{\boldsymbol{u}_j'\}_{j\in\mathcal{N}_i}]$, $\boldsymbol{z}_i^{\mathrm{a}} \leftarrow \boldsymbol{x}_i^{\mathrm{a}}$, $\boldsymbol{w}_i^{\mathrm{a}} \leftarrow \boldsymbol{u}_i^{\mathrm{a}}$, $\boldsymbol{\lambda}_i \leftarrow 0$, $\boldsymbol{y}_i \leftarrow 0$.
4: **while** $n \leq N$ **do**
5: **while** $l \leq L$ **do** (in parallel $\forall\ i \in \mathcal{V}$)
6: $\boldsymbol{x}_i^{\mathrm{a}}, \boldsymbol{u}_i^{\mathrm{a}} \leftarrow$ Solve (29) in parallel $\forall\ i \in \mathcal{V}$.
7: $\boldsymbol{w}_{i,k}, \beta_{i,k} \leftarrow$ Update in parallel $\forall\ i \in \mathcal{V}, \forall\ k \in$
8: $[\![0, K-1]\!]$.
9: **if** $\|\boldsymbol{s}_{i,k}^{\mathrm{a}}\|_2 \leq \epsilon_{\mathrm{AL}}$ **then**
10: **break**
11: *Each agent* $i \in \mathcal{V}$ *receives* $\boldsymbol{x}_i^j, \boldsymbol{u}_i^j$ *from all* $j \in \mathcal{P}_i \backslash \{i\}$.
12: $\boldsymbol{z}_i, \boldsymbol{w}_i \leftarrow$ Update with (30a), (30b) in parallel $\forall\ i \in \mathcal{V}$.
13: *Each agent* $i \in \mathcal{V}$ *receives* $\boldsymbol{z}_j, \boldsymbol{w}_j$ *from all* $j \in \mathcal{N}_i \backslash \{i\}$.
14: $\boldsymbol{\lambda}_i, \boldsymbol{y}_i \leftarrow$ Update with (31a), (31b) in parallel $\forall\ i \in \mathcal{V}$.

pass rules for DDP will obtain the following form

$$\begin{aligned}
Q_{\boldsymbol{x}_i^{\mathrm{a}}\boldsymbol{x}_i^{\mathrm{a}},k} &= \left(\sum_{j\in\mathcal{N}_i}\frac{1}{|\mathcal{P}_j|}\ell_j\right)_{\boldsymbol{x}_i^{\mathrm{a}}\boldsymbol{x}_i^{\mathrm{a}}} + \rho\boldsymbol{I} + C_{\boldsymbol{x}_i^{\mathrm{a}}\boldsymbol{x}_i^{\mathrm{a}},k} \\
&\quad + \boldsymbol{f}_{\boldsymbol{x}_i^{\mathrm{a}}}^{\mathrm{a}\,\mathrm{T}} V_{\boldsymbol{x}_i^{\mathrm{a}}\boldsymbol{x}_i^{\mathrm{a}},k+1} \boldsymbol{f}_{\boldsymbol{x}_i^{\mathrm{a}}}^{\mathrm{a}}, \\
Q_{\boldsymbol{x}_i^{\mathrm{a}}\boldsymbol{u}_i^{\mathrm{a}},k} &= \left(\sum_{j\in\mathcal{N}_i}\frac{1}{|\mathcal{P}_j|}\ell_j\right)_{\boldsymbol{x}_i^{\mathrm{a}}\boldsymbol{u}_i^{\mathrm{a}}} + \boldsymbol{f}_{\boldsymbol{x}_i^{\mathrm{a}}}^{\mathrm{a}\,\mathrm{T}} V_{\boldsymbol{x}_i^{\mathrm{a}}\boldsymbol{x}_i^{\mathrm{a}},k+1} \boldsymbol{f}_{\boldsymbol{u}_i^{\mathrm{a}}}^{\mathrm{a}}, \\
Q_{\boldsymbol{u}_i^{\mathrm{a}}\boldsymbol{u}_i^{\mathrm{a}},k} &= \left(\sum_{j\in\mathcal{N}_i}\frac{1}{|\mathcal{P}_j|}\ell_j\right)_{\boldsymbol{u}_i^{\mathrm{a}}\boldsymbol{u}_i^{\mathrm{a}}} + \mu\boldsymbol{I} + C_{\boldsymbol{u}_i^{\mathrm{a}}\boldsymbol{u}_i^{\mathrm{a}},k} \\
&\quad + \boldsymbol{f}_{\boldsymbol{u}_i^{\mathrm{a}}}^{\mathrm{a}\,\mathrm{T}} V_{\boldsymbol{x}_i^{\mathrm{a}}\boldsymbol{x}_i^{\mathrm{a}},k+1} \boldsymbol{f}_{\boldsymbol{u}_i^{\mathrm{a}}}^{\mathrm{a}}, \\
Q_{\boldsymbol{x}_i^{\mathrm{a}},k} &= \left(\sum_{j\in\mathcal{N}_i}\frac{1}{|\mathcal{P}_j|}\ell_j\right)_{\boldsymbol{x}_i^{\mathrm{a}}} + \rho(\boldsymbol{x}_{i,k}^{\mathrm{a}} - \boldsymbol{z}_{i,k}^{\mathrm{a}}) + \boldsymbol{\lambda}_{i,k} \\
&\quad + C_{\boldsymbol{x}_i^{\mathrm{a}},k} + \boldsymbol{f}_{\boldsymbol{x}_i^{\mathrm{a}}}^{\mathrm{a}\,\mathrm{T}} V_{\boldsymbol{x}_i^{\mathrm{a}},k+1}, \\
Q_{\boldsymbol{u}_i^{\mathrm{a}},k} &= \left(\sum_{j\in\mathcal{N}_i}\frac{1}{|\mathcal{P}_j|}\ell_j\right)_{\boldsymbol{u}_i^{\mathrm{a}}} + \mu(\boldsymbol{u}_{i,k}^{\mathrm{a}} - \boldsymbol{w}_{i,k}^{\mathrm{a}}) + \boldsymbol{y}_{i,k} \\
&\quad + C_{\boldsymbol{u}_i^{\mathrm{a}},k} + \boldsymbol{f}_{\boldsymbol{u}_i^{\mathrm{a}}}^{\mathrm{a}\,\mathrm{T}} V_{\boldsymbol{x}_i^{\mathrm{a}},k+1},
\end{aligned}$$

where, for simplicity, only the first-order dynamics expansion is considered. The terminal conditions for the value functions and their derivatives will be given by

$$\begin{aligned}
V_{i,K} &= \sum_{j\in\mathcal{N}_i}\frac{1}{|\mathcal{P}_j|}\phi_j(\boldsymbol{x}_{j,K}^i) + \frac{\rho}{2}\left\|\boldsymbol{x}_{i,K}^{\mathrm{a}} - \boldsymbol{z}_{i,K}^{\mathrm{a}} + \frac{\boldsymbol{\lambda}_{i,K}}{\rho}\right\|_2^2 \\
&\quad + C_{i,K}, \\
V_{\boldsymbol{x}_i^{\mathrm{a}},K} &= \left(\sum_{j\in\mathcal{N}_i}\frac{1}{|\mathcal{P}_j|}\phi_j(\boldsymbol{x}_{j,K}^i)\right)_{\boldsymbol{x}_i^{\mathrm{a}}} + \rho(\boldsymbol{x}_{i,K}^{\mathrm{a}} - \boldsymbol{z}_{i,K}^{\mathrm{a}}) \\
&\quad + \boldsymbol{\lambda}_{i,K} + C_{\boldsymbol{x}_i^{\mathrm{a}},K}, \\
V_{\boldsymbol{x}_i^{\mathrm{a}}\boldsymbol{x}_i^{\mathrm{a}},K} &= \left(\sum_{j\in\mathcal{N}_i}\frac{1}{|\mathcal{P}_j|}\phi_j(\boldsymbol{x}_{j,K}^i)\right)_{\boldsymbol{x}_i^{\mathrm{a}}\boldsymbol{x}_i^{\mathrm{a}}} + \rho\boldsymbol{I} + C_{\boldsymbol{x}_i^{\mathrm{a}}\boldsymbol{x}_i^{\mathrm{a}},K}.
\end{aligned}$$

Remark 1: The new local problems (29) include four significant modifications compared to their unconstrained single-agent counterparts. First, each agent optimizes for its augmented state and control which allows for handling inter-agent constraints - from their own perspective. Second, all agents partially take into account the objectives of their neighbors. Third, all neighborhood constraints are incorporated through the local AL for each agent. Finally, the new costs include extra terms that encourage a consensus between local and global variables, and therefore, a consensus between all agents.

Optimization Step 2: Global Updates. In the second block, the global variables $\boldsymbol{z}, \boldsymbol{w}$ are updated, i.e.,

$$\{\boldsymbol{z}, \boldsymbol{w}\}^{n+1} = \arg\min \mathcal{L}(\boldsymbol{x}_i^{\mathrm{a},n+1}, \boldsymbol{u}_i^{\mathrm{a},n+1}, \boldsymbol{z}, \boldsymbol{w}, \boldsymbol{\lambda}_i^n, \boldsymbol{y}_i^n),$$

which leads to

$$\begin{aligned}
\{\boldsymbol{z}, \boldsymbol{w}\}^{n+1} = \arg\min \sum_{i=1}^{M}\sum_{j\in\mathcal{N}_i} &\frac{\rho}{2}\|\boldsymbol{x}_j^i - \boldsymbol{z}_j\|_2^2 + \frac{\mu}{2}\|\boldsymbol{u}_j^i - \boldsymbol{w}_j\|_2^2 \\
&- \boldsymbol{\lambda}_j^{i\,\mathrm{T}}\boldsymbol{z}_j - \boldsymbol{y}_j^{i\,\mathrm{T}}\boldsymbol{w}_j.
\end{aligned}$$

This minimization can be decomposed for all $\boldsymbol{z}_i, \boldsymbol{w}_i,\ i \in \mathcal{V}$, leading to the following update rules

$$\boldsymbol{z}_i^{n+1} = \frac{1}{|\mathcal{P}_i|}\sum_{j\in\mathcal{P}_i}\boldsymbol{x}_i^{j,n+1} + \frac{1}{\rho}\boldsymbol{\lambda}_i^{j,n}, \tag{30a}$$

$$\boldsymbol{w}_i^{n+1} = \frac{1}{|\mathcal{P}_i|}\sum_{j\in\mathcal{P}_i}\boldsymbol{u}_i^{j,n+1} + \frac{1}{\mu}\boldsymbol{y}_i^{j,n}. \tag{30b}$$

For $n > 0$, the parts involving the dual variables in (30), will become zero with a similar argument as in [44, Section 7.2]. Therefore, the global updates are essentially taking an average of all variables (actual or copy ones) that refer to agent i. Finally, the dual variables are updated as follows

$$\boldsymbol{\lambda}_i^{n+1} = \boldsymbol{\lambda}_i^n + \rho(\boldsymbol{x}_i^{\mathrm{a},n+1} - \boldsymbol{z}_i^{\mathrm{a},n+1}), \tag{31a}$$

$$\boldsymbol{y}_i^{n+1} = \boldsymbol{y}_i^n + \mu(\boldsymbol{u}_i^{\mathrm{a},n+1} - \boldsymbol{w}_i^{\mathrm{a},n+1}). \tag{31b}$$

C. Algorithm

The ND-DDP algorithm is presented in Alg. 1. To warmstart the variables (Line 1), each agent $i \in \mathcal{V}$ first obtains $\boldsymbol{x}_i', \boldsymbol{u}_i'$ through solving once with DDP their unconstrained single-agent problems which only take into account the dynamics constraints (12). Next, every agent $j \in \mathcal{N}_i \backslash \{i\}$ sends $\boldsymbol{x}_j', \boldsymbol{u}_j'$ to agent i (Line 2). These are used by each agent i to initialize $\boldsymbol{x}_i^{\mathrm{a}}$, $\boldsymbol{u}_i^{\mathrm{a}}$, $\boldsymbol{z}_i^{\mathrm{a}}$ and $\boldsymbol{w}_i^{\mathrm{a}}$ (Line 3). Subsequently, the iterative ADMM algorithm starts. In Lines 5-10, $\boldsymbol{x}_i^{\mathrm{a}}$ and $\boldsymbol{u}_i^{\mathrm{a}}$ are obtained by solving (28) with AL-DDP in parallel for all agents. This involves L loops, wherein $\boldsymbol{x}_i^{\mathrm{a}}$, $\boldsymbol{u}_i^{\mathrm{a}}$ (Line 6) and $\boldsymbol{w}_{i,k}, \beta_{i,k}$ (Line 7) are updated iteratively. The loop terminates early if the residuals of the local neighborhood constraints get below some threshold, i.e., $\|\boldsymbol{s}_{i,k}^{\mathrm{a}}\|_2 \leq \epsilon_{\mathrm{AL}}$. After that, each agent $j \in \mathcal{P}_i \backslash \{i\}$ sends $\boldsymbol{x}_i^j, \boldsymbol{u}_i^j$ to agent i (Line 11), so that the latter is able to perform the global updates (Line 12) using (30). Next, every agent $j \in \mathcal{N}_i \backslash \{i\}$ sends $\boldsymbol{z}_j, \boldsymbol{w}_j$ to agent i, so that the latter can construct $\boldsymbol{z}_i^{\mathrm{a}}$, $\boldsymbol{w}_i^{\mathrm{a}}$ (Line 13). Thus, every agent i can now perform its dual updates (Line 14) with (31). This iterative procedure (Lines 4-14) continues until n reaches to the maximum number of ADMM iterations N.

Remark 2: All computations (Lines 1,5-10,12,14) can be performed in parallel by every agent i. Furthermore, all necessary communication steps (Lines 2,11,13) only take place in a neighborhood level. These two attributes characterize the *fully decentralized* nature of ND-DDP from both *computational* and *communication* aspects.

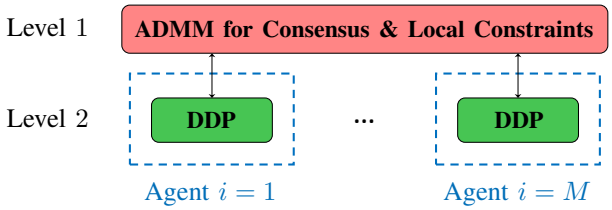

Fig. 3: The MD-DDP ("merged") architecture. Level 1: ADMM for consensus and local constraints; Level 2: DDP.

Remark 3: It is possible to use a termination criterion that would check whether the ADMM primal and dual residuals get below some threshold [44, Section 3.3]. Nevertheless, this would require collecting information from all agents at every ADMM iteration, thus violating the fully distributed form of the algorithm. For this reason, we empirically set a maximum number of iterations N that leads to sufficient convergence.

Remark 4 (Computational Complexity of ND-DDP): The most computationally demanding part of ND-DDP are the local AL-DDP updates (Step 1). Let us denote the DDP iterations with D. The computational bottleneck of the DDP algorithm itself is the matrix inversion $Q_{\boldsymbol{uu},k}^{-1}$ required for (7) which is performed K times per DDP iteration. Therefore, ND-DDP has the following computational complexity

$$O\big(N \cdot L \cdot D \cdot K \cdot \tilde{q}_i^3\big). \tag{32}$$

Note that applying AL-DDP directly on Problem 1 in a centralized fashion would lead to the following computational complexity

$$O\big(L \cdot D \cdot K \cdot (Mq_i)^3\big), \tag{33}$$

since the matrix inversion in (7) would have the dimensionality of the concatenated control vector of all agents. Thus, it is clear that ND-DDP can achieve dramatic computational gains against centralized AL-DDP as the number of agents M increases.

Remark 5: Any other constrained DDP method such as [34]–[38] could also be used instead of the AL-DDP combination (Lines 5-10) for solving problems (28) within ND-DDP.

V. Merged Distributed DDP

In the second proposed architecture, we wish to take advantage of the fact that ADMM can also be used for handling the neighborhood constraints locally rather than only the consensus ones. This would lead into "merging" the consensus and local constraints levels of ND-DDP into a single level (Fig. 3). For this reason, we label this framework as Merged Distributed DDP (MD-DDP).

A. Problem Transformation

In order to handle the local state and control constraints with ADMM, it is necessary to first introduce the "safe" copy variables $\tilde{\boldsymbol{x}}_{i,k} \in \mathbb{R}^{p_i},\ \tilde{\boldsymbol{u}}_{i,k} \in \mathbb{R}^{q_i},\ i \in \mathcal{V}$. These variables must satisfy the local single-agent constraints, i.e., $\boldsymbol{b}_{i,k}(\tilde{\boldsymbol{u}}_{i,k}) \leq 0$ and $\boldsymbol{g}_{i,k}(\tilde{\boldsymbol{x}}_{i,k}) \leq 0$, while of course being in consensus with the actual states and controls

$$\tilde{\boldsymbol{x}}_{i,k} = \boldsymbol{x}_{i,k},\ \tilde{\boldsymbol{u}}_{i,k} = \boldsymbol{u}_{i,k},\ i \in \mathcal{V}. \tag{34}$$

Moreover, each agent will also contain the following safe copy variables for its neighbors, $\tilde{\boldsymbol{x}}_{j,k}^i \in \mathbb{R}^{p_j},\ j \in \mathcal{N}_i \backslash \{i\},\ i \in \mathcal{V}$. Thus, in a similar manner as in Section IV, we can introduce the augmented "safe" states as follows

$$\tilde{\boldsymbol{x}}_{i,k}^{\mathrm{a}} = [\{\tilde{\boldsymbol{x}}_{j,k}^i\}_{j \in \mathcal{N}_i}] \in \mathbb{R}^{\tilde{p}_i},\ i \in \mathcal{V}, \tag{35}$$

for which we impose that $\boldsymbol{h}_{i,k}^{\mathrm{a}}(\tilde{\boldsymbol{x}}_{i,k}^{\mathrm{a}}) \leq 0$. The trajectories $\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j^i, \tilde{\boldsymbol{x}}_i^{\mathrm{a}}$ are defined accordingly. Finally, we define a global state $\boldsymbol{z}_k$ as in (21) and enforce the consensus constraints

$$\tilde{\boldsymbol{x}}_i^{\mathrm{a}} = \boldsymbol{z}_i^{\mathrm{a}},\ i \in \mathcal{V}. \tag{36}$$

As a result, we can formulate the following problem which is equivalent to Problem 1.

Problem 3 (Multi-Agent Optimal Control Problem III): Find the optimal control sequences $\boldsymbol{u}_i^*,\ \forall i \in \mathcal{V}$, such that

$$\begin{aligned} &\{\boldsymbol{u}_i^*\}_{i \in \mathcal{V}} = \arg\min \sum_{i=1}^{M} J_i(\boldsymbol{x}_i, \boldsymbol{u}_i) \\ \text{s.t.}\quad &\boldsymbol{x}_{i,k+1} = \boldsymbol{f}_i(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}),\ \boldsymbol{x}_{i,0} : \text{given}, \\ &\boldsymbol{b}_{i,k}(\tilde{\boldsymbol{u}}_{i,k}) \leq 0,\ \boldsymbol{g}_{i,k}(\tilde{\boldsymbol{x}}_{i,k}) \leq 0,\ \boldsymbol{h}_{i,k}^{\mathrm{a}}(\tilde{\boldsymbol{x}}_{i,k}^{\mathrm{a}}) \leq 0, \\ &\boldsymbol{u}_i = \tilde{\boldsymbol{u}}_i,\ \boldsymbol{x}_i = \tilde{\boldsymbol{x}}_i,\ \tilde{\boldsymbol{x}}_i^{\mathrm{a}} = \boldsymbol{z}_i^{\mathrm{a}},\ i \in \mathcal{V}. \end{aligned}$$

B. Method Derivation

The derivation of the MD-DDP updates follows. First, let us restate Problem 3 as

$$\begin{aligned} \min \sum_{i=1}^{M} &J_i(\boldsymbol{x}_i, \boldsymbol{u}_i) + \mathcal{I}_{\boldsymbol{f}_i}(\boldsymbol{x}_i, \boldsymbol{u}_i) + \mathcal{I}_{\boldsymbol{b}_i}(\tilde{\boldsymbol{u}}_i) \\ &+ \mathcal{I}_{\boldsymbol{g}_i}(\tilde{\boldsymbol{x}}_i) + \mathcal{I}_{\boldsymbol{h}_i^{\mathrm{a}}}(\tilde{\boldsymbol{x}}_i^{\mathrm{a}}) \\ \text{s.t.}\quad &\boldsymbol{u}_i = \tilde{\boldsymbol{u}}_i,\ \boldsymbol{x}_i = \tilde{\boldsymbol{x}}_i,\ \tilde{\boldsymbol{x}}_i^{\mathrm{a}} = \boldsymbol{z}_i^{\mathrm{a}},\ i \in \mathcal{V}. \end{aligned} \tag{37}$$

For this problem, the AL can be written as

$$\begin{aligned} \mathcal{L} = \sum_{i=1}^{M} &J_i(\boldsymbol{x}_i, \boldsymbol{u}_i) + \mathcal{I}_{\boldsymbol{f}_i}(\boldsymbol{x}_i, \boldsymbol{u}_i) + \mathcal{I}_{\boldsymbol{b}_i}(\tilde{\boldsymbol{u}}_i) + \mathcal{I}_{\boldsymbol{g}_i}(\tilde{\boldsymbol{x}}_i) \\ &+ \mathcal{I}_{\boldsymbol{h}_i^{\mathrm{a}}}(\tilde{\boldsymbol{x}}_i^{\mathrm{a}}) + \boldsymbol{\xi}_i^{\mathrm{T}}(\boldsymbol{u}_i - \tilde{\boldsymbol{u}}_i) + \boldsymbol{\lambda}_i^{\mathrm{T}}(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i) \\ &+ \boldsymbol{y}_i^{\mathrm{T}}(\tilde{\boldsymbol{x}}_i^{\mathrm{a}} - \boldsymbol{z}_i^{\mathrm{a}}) + \frac{\tau}{2}\|\boldsymbol{u}_i - \tilde{\boldsymbol{u}}_i\|_2^2 + \frac{\rho}{2}\|\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\|_2^2 \\ &+ \frac{\mu}{2}\|\tilde{\boldsymbol{x}}_i^{\mathrm{a}} - \boldsymbol{z}_i^{\mathrm{a}}\|_2^2. \end{aligned} \tag{38}$$

Optimization Step 1: Local DDP Updates. In the first block of updates, we minimize the AL w.r.t. the local variables $\boldsymbol{x}_i, \boldsymbol{u}_i$, i.e.,

$$\{\boldsymbol{x}_i, \boldsymbol{u}_i\}^{n+1} = \arg\min \mathcal{L}(\boldsymbol{x}_i, \boldsymbol{u}_i, \tilde{\boldsymbol{x}}_i^{\mathrm{a},n}, \tilde{\boldsymbol{u}}_i^n, \boldsymbol{z}^n, \boldsymbol{\xi}_i^n, \boldsymbol{\lambda}_i^n, \boldsymbol{y}_i^n),$$

for all agents $i \in \mathcal{V}$. This results to the following M subproblems

$$\begin{aligned} \{\boldsymbol{x}_i, \boldsymbol{u}_i\}^{n+1} = \arg\min &\sum_{k=0}^{K-1} \Big[\hat{\ell}_i(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k})\Big] + \hat{\phi}_i(\boldsymbol{x}_{i,K}), \\ \text{s.t.}\quad &\boldsymbol{x}_{i,k+1} = \boldsymbol{f}_i(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}),\ \boldsymbol{x}_{i,0} : \text{given}, \end{aligned} \tag{39}$$

with

$$\begin{aligned} \hat{\ell}_{i,k}(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}) = \ell_{i,k}(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}) + \frac{\rho}{2}\left\|\boldsymbol{x}_{i,k} - \tilde{\boldsymbol{x}}_{i,k} + \frac{\boldsymbol{\lambda}_{i,k}}{\rho}\right\|_2^2 \\ + \frac{\tau}{2}\left\|\boldsymbol{u}_{i,k} - \tilde{\boldsymbol{u}}_{i,k} + \frac{\boldsymbol{\xi}_{i,k}}{\tau}\right\|_2^2, \end{aligned}$$

$$\hat{\phi}_i(\boldsymbol{x}_{i,K}) = \phi_i(\boldsymbol{x}_{i,K}) + \frac{\rho}{2}\left\|\boldsymbol{x}_{i,K} - \tilde{\boldsymbol{x}}_{i,K} + \frac{\boldsymbol{\lambda}_{i,K}}{\rho}\right\|_2^2.$$

These problems are solved in parallel by every agent $i \in \mathcal{V}$, with DDP. The backward pass rules will obtain the following form

$$\begin{aligned}
Q_{\boldsymbol{x}_i\boldsymbol{x}_i,k} &= \ell_{\boldsymbol{x}_i\boldsymbol{x}_i} + \rho\boldsymbol{I} + \boldsymbol{f}_{\boldsymbol{x}_i}^{\mathrm{T}} V_{\boldsymbol{x}_i\boldsymbol{x}_i,k+1}\boldsymbol{f}_{\boldsymbol{x}_i},\\
Q_{\boldsymbol{x}_i\boldsymbol{u}_i,k} &= \ell_{\boldsymbol{x}_i\boldsymbol{u}_i} + \boldsymbol{f}_{\boldsymbol{x}_i}^{\mathrm{T}} V_{\boldsymbol{x}_i\boldsymbol{x}_i,k+1}\boldsymbol{f}_{\boldsymbol{u}_i},\\
Q_{\boldsymbol{u}_i\boldsymbol{u}_i,k} &= \ell_{\boldsymbol{u}_i\boldsymbol{u}_i} + \tau\boldsymbol{I} + \boldsymbol{f}_{\boldsymbol{u}_i}^{\mathrm{T}} V_{\boldsymbol{x}_i\boldsymbol{x}_i,k+1}\boldsymbol{f}_{\boldsymbol{u}_i},\\
Q_{\boldsymbol{x}_i,k} &= \ell_{\boldsymbol{x}_i} + \rho(\boldsymbol{x}_{i,k} - \tilde{\boldsymbol{x}}_{i,k}) + \boldsymbol{\lambda}_{i,k} + \boldsymbol{f}_{\boldsymbol{x}_i}^{\mathrm{T}} V_{\boldsymbol{x}_i,k+1},\\
Q_{\boldsymbol{u}_i,k} &= \ell_{\boldsymbol{u}_i} + \tau(\boldsymbol{u}_{i,k} - \tilde{\boldsymbol{u}}_{i,k}) + \boldsymbol{\xi}_{i,k} + \boldsymbol{f}_{\boldsymbol{u}_i}^{\mathrm{T}} V_{\boldsymbol{x}_i,k+1},
\end{aligned}$$

where only the first-order dynamics approximation is considered for brevity. The terminal conditions for the value functions and their derivatives will be

$$\begin{aligned}
V_{i,K} &= \phi_i(\boldsymbol{x}_{i,K}) + \frac{\rho}{2}\left\|\boldsymbol{x}_{i,K} - \tilde{\boldsymbol{x}}_{i,K} + \frac{\boldsymbol{\lambda}_{i,K}}{\rho}\right\|_2^2,\\
V_{\boldsymbol{x}_i,K} &= \phi_{\boldsymbol{x}_i}(\boldsymbol{x}_{i,K}) + \rho(\boldsymbol{x}_{i,K} - \tilde{\boldsymbol{x}}_{i,K}) + \boldsymbol{\lambda}_{i,K},\\
V_{\boldsymbol{x}_i\boldsymbol{x}_i,K} &= \phi_{\boldsymbol{x}_i\boldsymbol{x}_i}(\boldsymbol{x}_{i,K}) + \rho\boldsymbol{I}.
\end{aligned}$$

Optimization Step 2: Local Safe Updates. The second block yields the following update rules

$$\{\tilde{\boldsymbol{x}}_i^{\mathrm{a}}, \tilde{\boldsymbol{u}}_i\}^{n+1} = \arg\min \mathcal{L}(\boldsymbol{x}_i^{n+1}, \boldsymbol{u}_i^{n+1}, \tilde{\boldsymbol{x}}_i^{\mathrm{a}}, \tilde{\boldsymbol{u}}_i, \boldsymbol{z}^n, \boldsymbol{\xi}_i^n, \boldsymbol{\lambda}_i^n, \boldsymbol{y}_i^n),$$

which leads to the following M (for every agent $i = 1,\dots,M$) state subproblems

$$\begin{aligned}
\tilde{\boldsymbol{x}}_i^{\mathrm{a},n+1} = \arg\min\ & \frac{\rho}{2}\left\|\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i + \frac{\boldsymbol{\lambda}_i}{\rho}\right\|_2^2 + \frac{\mu}{2}\left\|\tilde{\boldsymbol{x}}_i^{\mathrm{a}} - \boldsymbol{z}_i^{\mathrm{a}} + \frac{\boldsymbol{y}_i}{\mu}\right\|_2^2\\
\text{s.t.}\quad & \boldsymbol{g}_{i,k}(\tilde{\boldsymbol{x}}_{i,k}) \le 0,\ \tilde{\boldsymbol{h}}_{i,k}(\tilde{\boldsymbol{x}}_{i,k}^{\mathrm{a}}) \le 0,\ k \in [\![0,K]\!], \quad (40)
\end{aligned}$$

and M control subproblems

$$\begin{aligned}
\tilde{\boldsymbol{u}}_i^{n+1} = \arg\min\ & \frac{\tau}{2}\left\|\boldsymbol{u}_i - \tilde{\boldsymbol{u}}_i + \frac{\boldsymbol{\xi}_i}{\tau}\right\|_2^2\\
\text{s.t.}\quad & \boldsymbol{b}_{i,k}(\tilde{\boldsymbol{u}}_{i,k}) \le 0,\ k \in [\![0,K-1]\!]. \quad (41)
\end{aligned}$$

Note that each agent's state subproblem (40) can be further decomposed into $K+1$ smaller subproblems (for each time instant $k = 0,\dots,K$), i.e.,

$$\begin{aligned}
\tilde{\boldsymbol{x}}_{i,k}^{\mathrm{a},n+1} = \arg\min\ & \frac{\rho}{2}\left\|\boldsymbol{x}_{i,k} - \tilde{\boldsymbol{x}}_{i,k} + \frac{\boldsymbol{\lambda}_{i,k}}{\rho}\right\|_2^2\\
& + \frac{\mu}{2}\left\|\tilde{\boldsymbol{x}}_{i,k}^{\mathrm{a}} - \boldsymbol{z}_{i,k}^{\mathrm{a}} + \frac{\boldsymbol{y}_{i,k}}{\mu}\right\|_2^2\\
\text{s.t.}\quad & \boldsymbol{g}_{i,k}(\tilde{\boldsymbol{x}}_{i,k}) \le 0,\ \boldsymbol{h}_{i,k}^{\mathrm{a}}(\tilde{\boldsymbol{x}}_{i,k}^{\mathrm{a}}) \le 0, \quad (42)
\end{aligned}$$

that can be solved in parallel as well. The same holds for the control subproblems, which can be decomposed into the following K subproblems (for $k = 0,\dots,K-1$)

$$\begin{aligned}
\tilde{\boldsymbol{u}}_{i,k}^{n+1} = \arg\min\ & \frac{\tau}{2}\left\|\boldsymbol{u}_{i,k} - \tilde{\boldsymbol{u}}_{i,k} + \frac{\boldsymbol{\xi}_{i,k}}{\tau}\right\|_2^2\\
\text{s.t.}\quad & \boldsymbol{b}_{i,k}(\tilde{\boldsymbol{u}}_{i,k}) \le 0. \quad (43)
\end{aligned}$$

Moreover, note that (42) and (43) are actually projection steps that admit trivial solutions in the case of box constraints. For example, if we have $\boldsymbol{u}_{i,\min} \le \tilde{\boldsymbol{u}}_{i,k} \le \boldsymbol{u}_{i,\max}$, then (43) yields

$$\tilde{\boldsymbol{u}}_{i,k}^{n+1} = \Pi_{[\boldsymbol{u}_{i,\min}, \boldsymbol{u}_{i,\max}]}\left(\boldsymbol{u}_{i,k} + \frac{\boldsymbol{\xi}_{i,k}}{\tau}\right), \quad (44)$$

which can be computed by clamping all elements of $\boldsymbol{u}_{i,k} + \boldsymbol{\xi}_{i,k}/\tau$, so that they are within $[\boldsymbol{u}_{i,\min}, \boldsymbol{u}_{i,\max}]$.

Algorithm 2 Merged Distributed DDP (MD-DDP)

1: $\boldsymbol{x}_i', \boldsymbol{u}_i' \leftarrow$ Solve single-agent problems in parallel $\forall\ i \in \mathcal{V}$.
2: *Each agent $i \in \mathcal{V}$ receives $\boldsymbol{x}_j'$ from all $j \in \mathcal{N}_i \backslash \{i\}$.*
3: **Initialize:** $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i'$, $\boldsymbol{u}_i \leftarrow \boldsymbol{u}_i'$, $\tilde{\boldsymbol{x}}_i^{\mathrm{a}} \leftarrow [\{\boldsymbol{x}_j'\}_{j\in\mathcal{N}_i}]$, $\tilde{\boldsymbol{u}}_i \leftarrow \boldsymbol{u}_i'$, $\boldsymbol{z}_i^{\mathrm{a}} \leftarrow \tilde{\boldsymbol{x}}_i^{\mathrm{a}}$, $\boldsymbol{\xi}_i \leftarrow 0$, $\boldsymbol{\lambda}_i \leftarrow 0$, $\boldsymbol{y}_i \leftarrow 0$.
4: **while** $n \le N$ **do**
5: $\quad \boldsymbol{x}_i, \boldsymbol{u}_i \leftarrow$ Solve (39) in parallel $\forall\ i \in \mathcal{V}$.
6: $\quad \tilde{\boldsymbol{x}}_i^{\mathrm{a}} \leftarrow$ Solve (42) in parallel $\forall\ i \in \mathcal{V}$, $\forall\ k \in [\![0,K]\!]$.
7: $\quad \tilde{\boldsymbol{u}}_i \leftarrow$ Solve (43) in parallel $\forall\ i \in \mathcal{V}$, $\forall\ k \in [\![0,K-1]\!]$.
8: $\quad$ *Each agent $i \in \mathcal{V}$ receives $\tilde{\boldsymbol{x}}_i^j$ from all $j \in \mathcal{P}_i \backslash \{i\}$.*
9: $\quad \boldsymbol{z}_i \leftarrow$ Update with (45) in parallel $\forall\ i \in \mathcal{V}$.
10: $\quad$ *Each agent $i \in \mathcal{V}$ receives $\boldsymbol{z}_j$ from all $j \in \mathcal{N}_i \backslash \{i\}$.*
11: $\quad \boldsymbol{\xi}_i, \boldsymbol{\lambda}_i, \boldsymbol{y}_i \leftarrow$ Update (46a)-(46c) in parallel $\forall\ i \in \mathcal{V}$.

Remark 6: It follows that Step 2 of MD-DDP is a *highly parallelizable* step, since it can be decomposed w.r.t. i) all agents, ii) states and controls and iii) all time instants. Therefore, it results into $M(2K+1)$ low-dimensional subproblems that can be solved in parallel.

Optimization Step 3: Global Updates. Similarly with Step 2 of ND-DDP, the global updates will be given by

$$\boldsymbol{z}_i^{n+1} = \frac{1}{|\mathcal{P}_i|}\sum_{j\in\mathcal{P}_i} \tilde{\boldsymbol{x}}_i^{j,n+1} + \frac{1}{\mu}\boldsymbol{y}_i^{j,n}. \quad (45)$$

Finally, the dual updates are provided by

$$\boldsymbol{\xi}_i^{n+1} = \boldsymbol{\xi}_i^n + \tau(\boldsymbol{u}_i^{n+1} - \tilde{\boldsymbol{u}}_i^{n+1}), \quad (46a)$$
$$\boldsymbol{\lambda}_i^{n+1} = \boldsymbol{\lambda}_i^n + \rho(\boldsymbol{x}_i^{n+1} - \tilde{\boldsymbol{x}}_i^{n+1}), \quad (46b)$$
$$\boldsymbol{y}_i^{n+1} = \boldsymbol{y}_i^n + \mu(\tilde{\boldsymbol{x}}_i^{\mathrm{a},n+1} - \boldsymbol{z}_i^{\mathrm{a},n+1}). \quad (46c)$$

C. Algorithm

The MD-DDP algorithm is illustrated in Alg. 2. As in ND-DDP, the single-agent unconstrained problems are initially solved once (Line 1), so that $\boldsymbol{x}_i$, $\boldsymbol{u}_i$, $\tilde{\boldsymbol{x}}_i^{\mathrm{a}}$, $\tilde{\boldsymbol{u}}_i$, $\boldsymbol{z}_i^{\mathrm{a}}$ are warmstarted (Line 3). During every ADMM iteration, all agents first solve problems (39) in parallel with DDP, to obtain $\boldsymbol{x}_i, \boldsymbol{u}_i$ (Line 5). Subsequently, problems (42),(43) are solved in parallel for all agents and all time instants (Lines 6,7). Afterwards, every agent $i \in \mathcal{V}$ receives $\tilde{\boldsymbol{x}}_i^j$ from all agents $j \in \mathcal{P}_i \backslash \{i\}$ (Line 8), so that the global updates (45) are executed (Line 9). Finally, each agent i gets $\boldsymbol{z}_j$ from all $j \in \mathcal{N}_i \backslash \{i\}$ (Line 10), so that the variables $\boldsymbol{z}_i^{\mathrm{a}}$ are constructed and the dual updates can be performed (Line 11). Lines 5-11 repeat until n reaches to N.

Remark 7: As with ND-DDP, the MD-DDP algorithm is *fully decentralized* since all computations can be performed in parallel among all agents (Lines 1,5-7,9,11), while the required communication steps (Lines 2,8,10) take place in a neighborhood level.

Remark 8 (Computational Complexity of MD-DDP): Using a similar justification as in Remark 4, the computational complexity of MD-DDP is given by

$$O\big(N \cdot D \cdot K \cdot q_i^3\big). \quad (47)$$

Note that this is a major computational improvement even compared to ND-DDP, since the AL loop has been eliminated

and the DDP subproblems only have a single-agent dimensionality, i.e., the matrix inversion in (7) has a $O(q_i^3)$ cost.

VI. Architectures Improvements and Details

In this section, we propose two improvements for enhancing the performance of ND-DDP and MD-DDP. First, we show how Nesterov acceleration can be incorporated through the ADMM updates to accelerate the convergence of the algorithms. Second, we propose a decentralized *"agent-specific"* scheme for adapting the ADMM penalty parameters that is suitable for multi-agent optimal control. For the sake of brevity, both are being presented only in the context of MD-DDP. Further details on the algorithms are also provided.

A. Nesterov Acceleration

A Nesterov accelerated version of MD-DDP is presented here. For more details on ADMM with Nesterov acceleration, the reader is referred to [59], [60]. Let us introduce the Nesterov duplicate variables $\bar{\boldsymbol{x}}_i^{\mathrm{a}}$, $\bar{\boldsymbol{u}}_i$, $\bar{\boldsymbol{z}}$, $\bar{\boldsymbol{\xi}}_i$, $\bar{\boldsymbol{\lambda}}_i$ and $\bar{\boldsymbol{y}}_i$. The updates of the modified MD-DDP algorithm are carried out as detailed below. In Step 1 (Local DDP updates), the variables $\boldsymbol{x}_i, \boldsymbol{u}_i$ are updated through

$$\{\boldsymbol{x}_i, \boldsymbol{u}_i\}^{n+1} = \arg\min \mathcal{L}(\boldsymbol{x}_i, \boldsymbol{u}_i, \bar{\boldsymbol{x}}_i^{\mathrm{a},n}, \bar{\boldsymbol{u}}_i^n, \bar{\boldsymbol{z}}^n, \bar{\boldsymbol{\xi}}_i^n, \bar{\boldsymbol{\lambda}}_i^n, \bar{\boldsymbol{y}}_i^n)$$

which leads to problems (39), but with $\bar{\boldsymbol{x}}_i$, $\bar{\boldsymbol{u}}_i$, $\bar{\boldsymbol{\lambda}}_i$, $\bar{\boldsymbol{\xi}}_i$ in place of $\tilde{\boldsymbol{x}}_i$, $\tilde{\boldsymbol{u}}_i$, $\boldsymbol{\lambda}_i$, $\boldsymbol{\xi}_i$, respectively. Step 2 (Local Safe updates) yields the following updates for $\tilde{\boldsymbol{x}}_i^{\mathrm{a}}$, $\tilde{\boldsymbol{u}}_i$

$$\{\tilde{\boldsymbol{x}}_i^{\mathrm{a}}, \tilde{\boldsymbol{u}}_i\}^{n+1} = \arg\min \mathcal{L}(\boldsymbol{x}_i^{n+1}, \boldsymbol{u}_i^{n+1}, \tilde{\boldsymbol{x}}_i^{\mathrm{a}}, \tilde{\boldsymbol{u}}_i, \bar{\boldsymbol{z}}^n, \bar{\boldsymbol{\xi}}_i^n, \bar{\boldsymbol{\lambda}}_i^n, \bar{\boldsymbol{y}}_i^n)$$

which results to (42), (43) using $\bar{\boldsymbol{z}}_{i,k}^{\mathrm{a}}$, $\bar{\boldsymbol{\xi}}_{i,k}$, $\bar{\boldsymbol{\lambda}}_{i,k}$, $\bar{\boldsymbol{y}}_{i,k}$ instead of $\boldsymbol{z}_{i,k}^{\mathrm{a}}$, $\boldsymbol{\xi}_{i,k}$, $\boldsymbol{\lambda}_{i,k}$, $\boldsymbol{y}_{i,k}$, respectively. Subsequently, Step 3 (Global updates) will have the following form

$$\bar{\boldsymbol{z}}_i^{n+1} = \frac{1}{|\mathcal{P}_i|} \sum_{j \in \mathcal{P}_i} \tilde{\boldsymbol{x}}_i^{j,n+1} + \frac{1}{\mu} \bar{\boldsymbol{y}}_i^{j,n}. \tag{48}$$

The dual updates are provided by

$$\bar{\boldsymbol{\xi}}_i^{n+1} = \bar{\boldsymbol{\xi}}_i^n + \tau(\boldsymbol{u}_i^{n+1} - \tilde{\boldsymbol{u}}_i^{n+1}), \tag{49a}$$
$$\bar{\boldsymbol{\lambda}}_i^{n+1} = \bar{\boldsymbol{\lambda}}_i^n + \rho(\boldsymbol{x}_i^{n+1} - \tilde{\boldsymbol{x}}_i^{n+1}), \tag{49b}$$
$$\bar{\boldsymbol{y}}_i^{n+1} = \bar{\boldsymbol{y}}_i^n + \mu(\tilde{\boldsymbol{x}}_i^{\mathrm{a},n+1} - \boldsymbol{z}_i^{\mathrm{a},n+1}). \tag{49c}$$

Finally, the Nesterov duplicate variables are updated as follows

$$\bar{\boldsymbol{x}}_i^{\mathrm{a},n+1} = \tilde{\boldsymbol{x}}_i^{\mathrm{a},n+1} + \gamma_n(\tilde{\boldsymbol{x}}_i^{\mathrm{a},n+1} - \tilde{\boldsymbol{x}}_i^{\mathrm{a},n}), \tag{50a}$$
$$\bar{\boldsymbol{u}}_i^{n+1} = \tilde{\boldsymbol{u}}_i^{n+1} + \gamma_n(\tilde{\boldsymbol{u}}_i^{n+1} - \tilde{\boldsymbol{u}}_i^n), \tag{50b}$$
$$\bar{\boldsymbol{z}}_i^{n+1} = \boldsymbol{z}_i^{n+1} + \gamma_n(\boldsymbol{z}_i^{n+1} - \boldsymbol{z}_i^n), \tag{50c}$$
$$\bar{\boldsymbol{\xi}}_i^{n+1} = \boldsymbol{\xi}_i^{n+1} + \gamma_n(\boldsymbol{\xi}_i^{n+1} - \boldsymbol{\xi}_i^n), \tag{50d}$$
$$\bar{\boldsymbol{\lambda}}_i^{n+1} = \boldsymbol{\lambda}_i^{n+1} + \gamma_n(\boldsymbol{\lambda}_i^{n+1} - \boldsymbol{\lambda}_i^n), \tag{50e}$$
$$\bar{\boldsymbol{y}}_i^{n+1} = \boldsymbol{y}_i^{n+1} + \gamma_n(\boldsymbol{y}_i^{n+1} - \boldsymbol{y}_i^n), \tag{50f}$$

where $\gamma_n = \eta \dfrac{\alpha_n - 1}{\alpha_{n+1}}$, $\{\alpha_n\}_{n=0,1,\dots}$ is the Nesterov sequence

$$\alpha_{n+1} = \frac{1 + \sqrt{1 + 4\alpha_n^2}}{2}, \quad \alpha_1 = 1,$$

and $\eta \in [0, 1)$ is a tuning parameter. For $\eta = 0$, the algorithm collapses to vanilla MD-DDP.

B. Decentralized Penalty Parameters Adaptation

Proper selection of the penalty parameters in ADMM algorithms is significant since it affects how fast they will converge to a satisfying solution, i.e., the primal and dual residuals will reach below some sufficient thresholds [44]. In general, these parameters are treated as tuning parameters, although several adaptation schemes have been proposed in the literature such as [61]–[64]. Most of them rely on steering the total primal and dual residuals ratio to some desired value [61], [62], since high or low values of the parameters prioritize the reduction of the primal or dual residuals, respectively. Of course, computing the total residuals of the global problem would require a central node, sacrificing the fully decentralized form of ND-DDP and MD-DDP. In [63], the residual balancing idea is extended by using different penalty parameters per node/agent.

Our observations have shown that schemes of this nature can be particularly useful in optimal control for multi-agent systems, while also admitting an intuitive interpretation. In fact, we take this idea one step further and propose (in the context of MD-DDP) that each agent contains the following diagonal *penalty parameter matrices*

$$\boldsymbol{T}_i \in \mathbb{R}^{q_i \times q_i},\ \boldsymbol{P}_i \in \mathbb{R}^{p_i \times p_i},\ \boldsymbol{M}_i \in \mathbb{R}^{\tilde{p}_i \times \tilde{p}_i},\ i \in \mathcal{V}, \tag{51}$$

with every diagonal element corresponding to a particular control/state component. The intuition here is that not only each agent should be penalized differently based on whether its local constraints are satisfied or not, but moreover, each control/state component should also be penalized separately. This is crucial in the case where the original costs (14) assign significantly different weights to these components, so the penalty parameters should be attuned to such discrepancies.

The resulting modified updates for MD-DDP are provided in Appendix B. At each ADMM iteration, the penalty parameter matrices are updated as follows

$$\boldsymbol{T}_i^{n+1} = a_{1,i}^{n+1} \boldsymbol{T}_i^0, \tag{52a}$$
$$\boldsymbol{P}_i^{n+1} = a_{2,i}^{n+1} \boldsymbol{P}_i^0, \tag{52b}$$
$$\boldsymbol{M}_i^{n+1} = a_{3,i}^{n+1} \boldsymbol{M}_i^0, \tag{52c}$$

where $\boldsymbol{T}_i^0$, $\boldsymbol{P}_i^0$, $\boldsymbol{M}_i^0$ are their initially assigned values. The update rules for the parameters $a_{b,i}^{n+1}$, $b = 1, 2, 3$, depend on the corresponding primal and dual residuals ratios, with detailed expressions provided in Appendix C.

Of particular interest is the quite common case, where the costs (14) are quadratic, i.e.,

$$J_i = \sum_{k=0}^{K-1} \Big[(\boldsymbol{x}_{i,k} - \boldsymbol{x}_i^{\mathrm{g}})^{\mathrm{T}} \boldsymbol{Q}_i (\boldsymbol{x}_{i,k} - \boldsymbol{x}_i^{\mathrm{g}}) + \boldsymbol{u}_{i,k}^{\mathrm{T}} \boldsymbol{R}_i \boldsymbol{u}_{i,k} \Big] + (\boldsymbol{x}_{i,K} - \boldsymbol{x}_i^{\mathrm{g}})^{\mathrm{T}} \boldsymbol{Q}_i^{\mathrm{f}} (\boldsymbol{x}_{i,K} - \boldsymbol{x}_i^{\mathrm{g}}), \tag{53}$$

with $\boldsymbol{Q}_i$, $\boldsymbol{R}_i$ being diagonal matrices and $\boldsymbol{x}_i^{\mathrm{g}}$ being the target state of the i-th agent. In this case, a well-suited initialization is $\boldsymbol{T}_i^0 = c_1 \cdot \boldsymbol{R}_i$, $\boldsymbol{P}_i^0 = c_2 \cdot \boldsymbol{Q}_i$, $\boldsymbol{M}_i^0 = c_3 \cdot \mathrm{bdiag}(\{\boldsymbol{Q}_j\}_{j \in \mathcal{N}_i})$, where $c_1, c_2, c_3 > 0$ with reasonable values, say 0.1 to 10. Choosing values $c_1, c_2, c_3 > 1$ would assign more importance to consensus in the MD-DDP updates (see Appendix B), while the opposite would assign more importance to the original costs. The intuition behind this choice is that each state/control component of each agent should be penalized by a symmetric

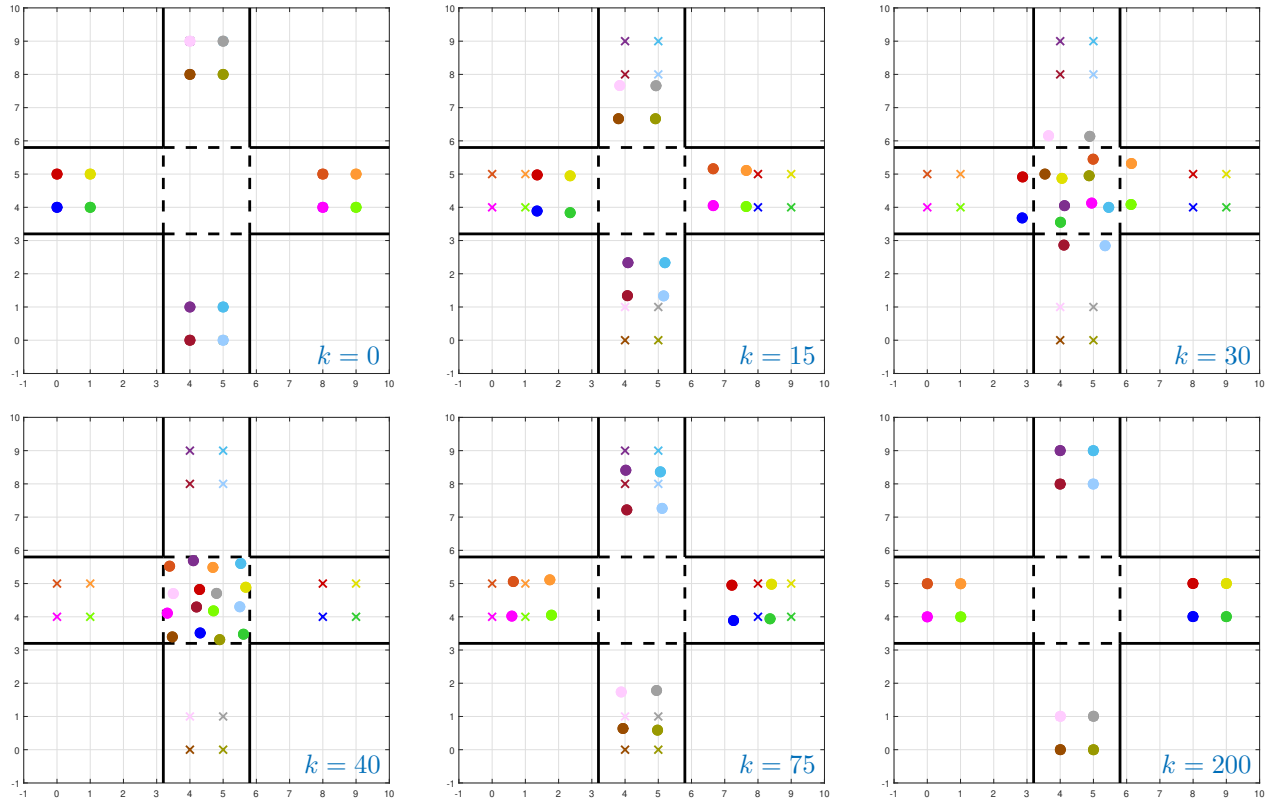

Fig. 4: Multi-vehicle intersection task with MD-DDP including 16 cars: Snapshots at different time instants. Each 'x' marker indicates the target of the car with the same color. The black lines indicate the lane bounds of each team of cars.

amount as in the original costs. Subsequently, depending on which constraints are violated or not, the updates (52) will lead the penalty parameters to appropriate values, while still respecting the original relative weighting that was assigned to each state/control component by the cost matrices $\boldsymbol{Q}_i$, $\boldsymbol{R}_i$. It is straightforward to apply the same ideas in ND-DDP as well, by replacing the parameters ρ and μ in Section IV, with matrices $\boldsymbol{P}_i \in \mathbb{R}^{\tilde{p}_i \times \tilde{p}_i}$ and $\boldsymbol{M}_i \in \mathbb{R}^{\tilde{q}_i \times \tilde{q}_i}$ for every agent i.

C. Convergence

DDP is known to admit a quadratic convergence rate to a locally optimal solution [29], [30]. In addition, convergence guarantees for ADMM are well-established in convex optimization problems [44]. While convergence results on problems with nonconvex constraints are limited in the literature and require significant modifications that reduce computational efficiency [52], [65], [66], extensive computational experience has shown that ADMM works well in practice for nonconvex problems [67]–[73]. In fact, after thorough testing, no divergence cases have been observed for both vanilla ND-DDP and MD-DDP. For their Nesterov accelerated versions, a sufficient tuning of parameter η is necessary to ensure that the algorithms will not diverge [59].

D. DDP Details

The performance of DDP can be further improved with two main modifications, which we adopt in the proposed frameworks. First, during the backward pass of DDP, we perform a sufficient regularization on the $\boldsymbol{Q}_{\boldsymbol{uu}}$ matrix such that it is ensured that it is a positive definite matrix using the adaptation rule proposed in [29]. Second, at every DDP forward pass, we use the following modified version of (7),

$$\delta \boldsymbol{u}_k^* = \alpha \boldsymbol{k}_k + \boldsymbol{K}_k \delta \boldsymbol{x}_k, \tag{54}$$

where a line search is performed to find the optimal α that leads to the highest cost reduction [23]. In all cases, DDP is warmstarted with the previous solution of each agent.

VII. Simulation Results

In this section, the efficacy of the proposed methods is verified through extensive simulation results. We start by demonstrating their performance in various challenging multi-vehicle problems and gradually increase to large-scale scenarios with thousands of vehicles. Subsequently, the capability of the algorithms to handle more complex dynamics in 3D space is illustrated by successfully testing them in multi-UAV control problems with up to hundreds of drones. An ablative analysis which shows how the improvements proposed in Section VI can enhance performance is provided afterwards. Finally, the superior scalability of ND-DDP and MD-DDP against popular multi-agent optimal control alternatives such as centralized/decentralized SQP and centralized/semi-centralized DDP is highlighted for large-scale problems. In all results, only the first-order expansions of the dynamics are used. For all tasks, the results of both MD-DDP and ND-DDP

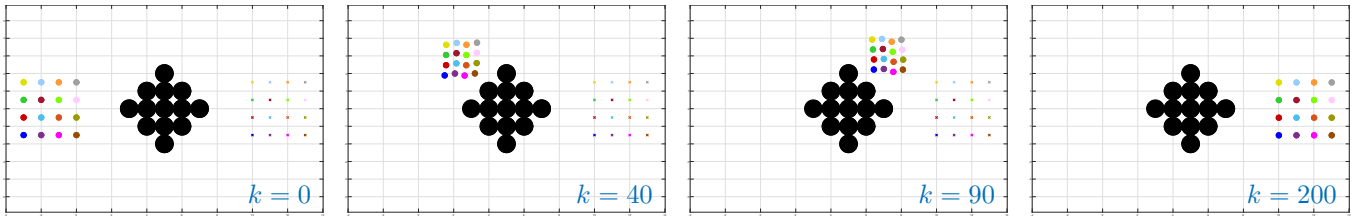

Fig. 5: Multi-vehicle "communication maintenance" task with MD-DDP (16 cars). Snapshots at different time instants.

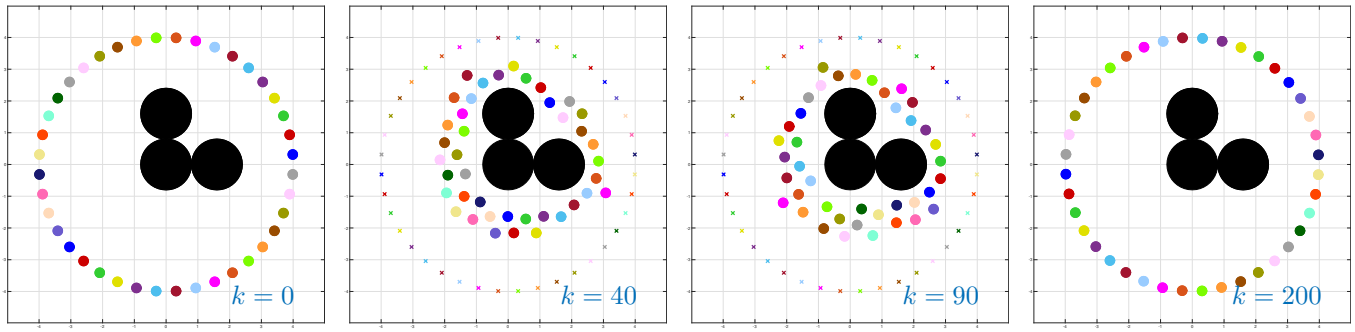

Fig. 6: Multi-vehicle "circle swapping" task with 40 cars and three unsymmetrical obstacles (black), solved with MD-DDP. Snapshots at different time instants.

are included in the supplementary video. In the main paper, due to space limitations, only snapshots using MD-DDP are presented.

A. Multi-Vehicle Control

We consider a problem setup where each agent i has Dubin's car dynamics with state $\boldsymbol{x}_{i,k} = \begin{bmatrix} \mathrm{x}_{i,k} & \mathrm{y}_{i,k} & \theta_{i,k} & v_{i,k} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^4$ and control input $\boldsymbol{u}_{i,k} = \begin{bmatrix} a_{i,k} & \omega_{i,k} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^2$, where $(\mathrm{x}_{i,k}, \mathrm{y}_{i,k})$ is the 2D position, $\theta_{i,k}$ is the angle, $v_{i,k}$, $\omega_{i,k}$ are the linear and angular velocities and $a_{i,k}$ is the linear acceleration of vehicle i at time instant k. For all tasks, we use time step $dt = 0.02$s and time horizon $K = 200$, unless specified otherwise. Each agent's cost is of the form (53) with $\boldsymbol{Q}_i = \mathrm{diag}(30, 30, 0, 6)$, $\boldsymbol{R}_i = \mathrm{diag}(0.5, 0.5)$ and $\boldsymbol{Q}_i^{\mathrm{f}} = \mathrm{diag}(100, 100, 0, 100)$. All cars are subject to the following box control constraints

$$-a_{\max} \leq a_{i,k} \leq a_{\max}, \quad -\omega_{\max} \leq \omega_{i,k} \leq \omega_{\max}, \tag{55}$$

with $a_{\max} = 10\mathrm{m/s}^2$ and $\omega_{\max} = 30^\circ/\mathrm{s}$. The following box state constraints must also be satisfied

$$\mathrm{x}_{\min} \leq \mathrm{x}_{i,k} \leq \mathrm{x}_{\max}, \quad \mathrm{y}_{\min} \leq \mathrm{y}_{i,k} \leq \mathrm{y}_{\max}, \tag{56}$$

$$-v_{\max} \leq v_{i,k} \leq v_{\max}, \tag{57}$$

where $\mathrm{x}_{\min/\max}$, $\mathrm{y}_{\min/\max}$ are the field bounds for each task and $v_{\max} = 10\mathrm{m/s}$. Furthermore, all vehicles are subject to the obstacle avoidance constraints

$$\|\boldsymbol{p}_{i,k} - \boldsymbol{p}_o\|_2 \geq r_o + d_o, \ o \in \mathcal{O}, \ i \in \mathcal{V}, \tag{58}$$

where $\boldsymbol{p}_{i,k} = \begin{bmatrix} \mathrm{x}_{i,k}; \mathrm{y}_{i,k} \end{bmatrix}$, while $\boldsymbol{p}_o$, r_o and d_o are the position, radius and desired minimum distance from obstacle $o \in \mathcal{O}$, and $\mathcal{O}$ is the set of all obstacles. For all obstacles, we set $d_o = 0.3$m. Lastly, the following inter-agent collision avoidance and connectivity maintenance constraints are also enforced between each vehicle i and its neighbors

$$\|\boldsymbol{p}_{i,k} - \boldsymbol{p}_{j,k}\|_2 \geq d_{\mathrm{col}}, \ j \in \mathcal{N}_i \backslash \{i\}, \ i \in \mathcal{V}, \tag{59}$$

$$\|\boldsymbol{p}_{i,k} - \boldsymbol{p}_{j,k}\|_2 \leq d_{\mathrm{con}}, \ j \in \mathcal{N}_i \backslash \{i\}, \ i \in \mathcal{V}, \tag{60}$$

with $d_{\mathrm{col}} = 0.3$m and $d_{\mathrm{con}} = 2.0$m. While constraints (59) ensure that the cars will not collide with each other, the ones in (60) force neighboring cars to stay within a close distance. Note that (58)-(60) can be handled directly through the local ALs in ND-DDP. For MD-DDP, we opt to linearize them [74, Section III.A] at every ADMM iteration n, around the trajectories $\boldsymbol{x}_i^n$ obtained from Step 1. Hence, problems (42) will be small-dimensional QPs that can be solved very fast - especially considering Remark 6. An additional communication step will be needed in MD-DDP between Steps 1 and 2 where every agent i must send $\boldsymbol{x}_i^n$ to agents $j \in \mathcal{P}_i \backslash \{i\}$, so that subsequently, each agent i linearizes the constraints (59) and (60) around the trajectories $\boldsymbol{x}_j^n$, $j \in \mathcal{N}_i$. In Sections VII-A and VII-B, we use the vanilla versions of ND-DDP and MD-DDP with constant agent-specific penalty parameter matrices - but without parameter adaptation or Nesterov acceleration. For ND-DDP, we set $\boldsymbol{M}_i = 2 \cdot \mathrm{bdiag}(\{\boldsymbol{R}_i\}_{j \in \mathcal{N}_i})$, $\boldsymbol{P}_i = 8 \cdot \mathrm{bdiag}(\{\boldsymbol{Q}_j\}_{j \in \mathcal{N}_i})$ and for MD-DDP, we assign $\boldsymbol{T}_i = 2 \cdot \boldsymbol{R}_i$, $\boldsymbol{P}_i = 8 \cdot \boldsymbol{Q}_i$, $\boldsymbol{M}_i = 8 \cdot \mathrm{bdiag}(\{\boldsymbol{Q}_j\}_{j \in \mathcal{N}_i})$ for every vehicle. The maximum number of ADMM iterations is set to $N = 50$ for ND-DDP and $N = 200$ for MD-DDP. In all experiments, the neighborhood sets are selected with a closest distance criterion based on the initial positions of the agents.

In the first task (Fig. 4), four groups of 4 cars are initialized with velocities $v_{i,0} = 3$m/s, while moving towards an intersection with a high chance of colliding. Except for avoiding

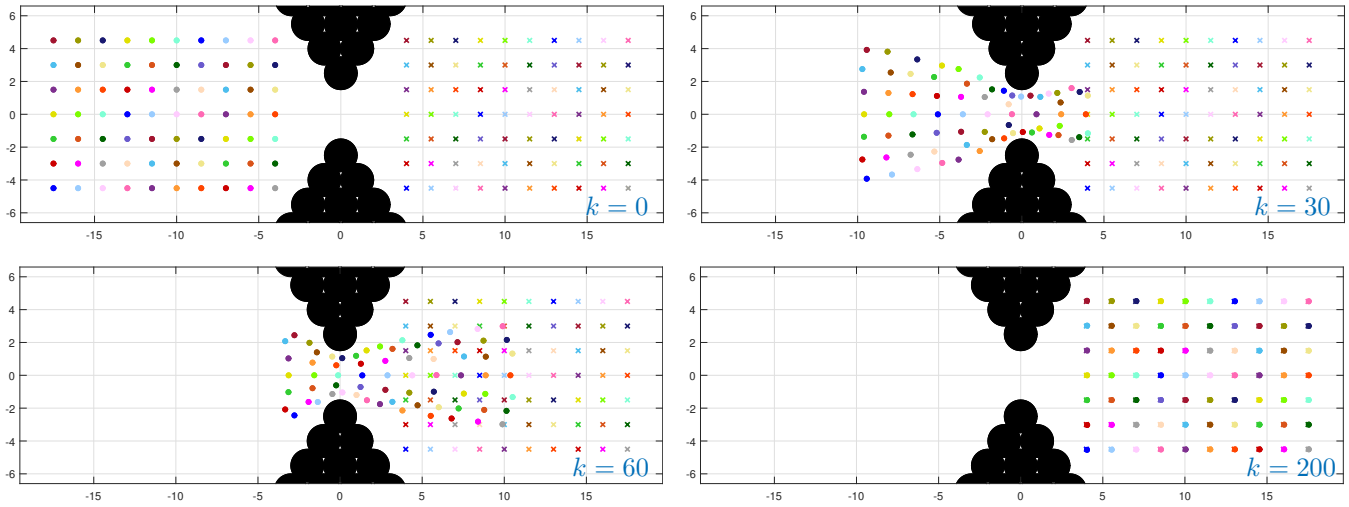

Fig. 7: Multi-vehicle "bottleneck" task with MD-DDP including 70 cars: Snapshots at different time instants.

collisions with all other cars ($\mathcal{N}_i = \mathcal{V}$) and reaching their targets, the vehicles are also required to remain within their lanes. Since all cars are neighbors here, constraints (60) can be omitted. Figure 4 shows six snapshots of the vehicles motion using the MD-DDP optimal solutions. All cars are able to reach their goals while maintaining their safety. Note that in cases where the neighborhood size is close to the total number of agents, the computational benefit of ND-DDP against centralized AL-DDP decreases. On the contrary, MD-DDP is immune to such choices, since the local DDP problems in Step 1 always maintain a single-agent dimensionality, irrespective of the neighborhood size (see Remark 8).

The second task includes a team of 16 vehicles that is required to surpass a cluster of obstacles while staying close to each other so that communication is maintained. The neighborhood size for each car is set to $|\mathcal{N}_i| = 9$. As illustrated in Fig. 5, the cars are able to safely reach to their target positions while not colliding and maintaining communication.

In the next task (Fig. 6), 40 cars are initialized in a circle formation. Their goal is to swap positions with the diametrically opposite ones while avoiding each other and three unsymmetrical obstacles at the center. We set $|\mathcal{N}_i| = 5$ for all cars. The optimal trajectories acquired with MD-DDP are shown in Fig. 6. All cars successfully rotate around the obstacles and reach their targets while staying safe.

The capabilities of the methods are further demonstrated in a problem where a team of 70 cars has to pass through a narrow "bottleneck" while avoiding collisions in order to reach their targets. Figure 7 illustrates how the cars are able to safely complete this task using MD-DDP. Neighborhoods of size of $|\mathcal{N}_i| = 9$ are used here.

Finally, to exhibit the scalability of both frameworks to large-scale problems, we consider a task where a team of vehicles has to move from one square grid to another while again avoiding collisions with each other and multiple obstacles. Figure 8 shows a scenario where 1,024 cars successfully complete the task with MD-DDP. The supplementary video includes results with 256 cars using ND-DDP and with up to 4,096 cars using MD-DDP. The 256 vehicles case includes 1,024 states and 512 control inputs. With time horizon $K = 200$, this is a problem with 307,200 optimization variables in total. The 4,096 vehicles problem, that MD-DDP is able to solve, includes 16,384 states and 8,192 controls, which using a time horizon $K = 800$, results to a 19,660,800-dimensional optimization problem. To our best knowledge, ND-DDP and MD-DDP are the first decentralized DDP-based methods to exhibit such a scalability for large-scale multi-robot control.

B. Multi-UAV Control

Next, the proposed algorithms are tested in multi-UAV problems that involve more complex dynamics and a 3D environment. Each drone i has a state $\boldsymbol{x}_{i,k} = \left[\boldsymbol{p}_{i,k};\ \boldsymbol{v}_{i,k};\ \boldsymbol{\eta}_{i,k};\ \boldsymbol{\nu}_{i,k}\right] \in \mathbb{R}^{12}$ and control input $\boldsymbol{u}_{i,k} = \left[F_{i,k}^{(1)};\ F_{i,k}^{(2)};\ F_{i,k}^{(3)};\ F_{i,k}^{(4)}\right] \in \mathbb{R}^4$, where $\boldsymbol{p}_{i,k} \in \mathbb{R}^3$ is the center of mass absolute linear position, $\boldsymbol{v}_{i,k} \in \mathbb{R}^3$ are the linear velocities, $\boldsymbol{\eta}_{i,k} \in \mathbb{R}^3$ are the Euler angles, $\boldsymbol{\nu}_{i,k} \in \mathbb{R}^3$ are the angular velocities and $F_{i,k}^{(c)},\ c \in [\![1, 4]\!]$ is the thrust force of the c-th rotor. The drones dynamics and parameters can be found in [75]. The discrete dynamics are obtained through Euler discretization with time step $dt = 0.02$s. Each drone has a quadratic cost (53) with $\boldsymbol{Q}_i = \text{diag}(150, 150, 150, 50, \ldots, 50)$, $\boldsymbol{R}_i = \text{diag}(1, \ldots, 1)$ and $\boldsymbol{Q}_i^{\text{f}} = \boldsymbol{Q}_i$. The following thrust forces limits must be satisfied

$$0 \leq F_{i,k}^{(c)} \leq F_{\max},\ c \in [\![1, 4]\!], \tag{61}$$

with $F_{\max} = 30$N. Position constraints similar to (56), are imposed in 3D so that the drones remain inbound. Collision avoidance (59) and connectivity maintenance constraints (60) must also be satisfied with $d_{\text{col}} = 0.5$m and $d_{\text{con}} = 2.0$m. The constraints are handled by each method as in Section VII-A and the drones are always initialized at hovering state. The same amounts of iterations N as in Section VII-A are used. Finally, neighbors are selected with a closest distance criterion.

The first task (Fig. 9) requires 12 drones to reach their targets while passing through the green "gate". Given a time

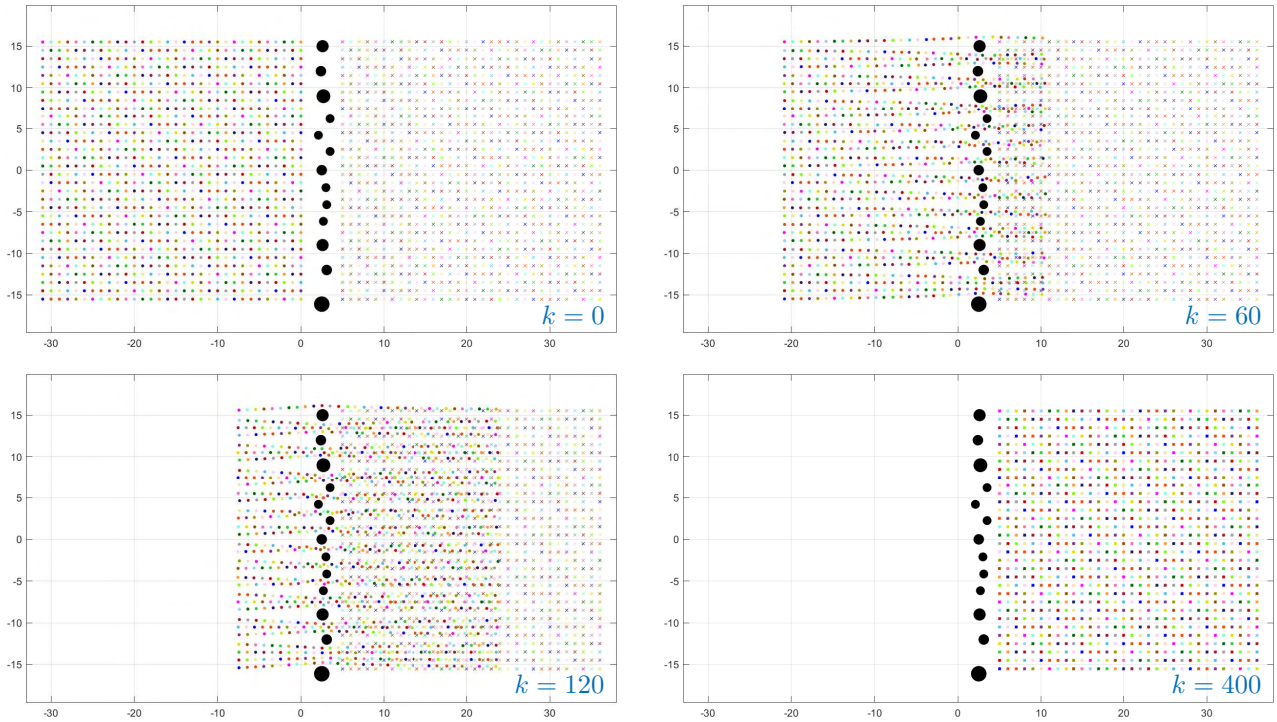

Fig. 8: Multi-vehicle formation task with MD-DDP including $1,024$ cars: Snapshots at different time instants.

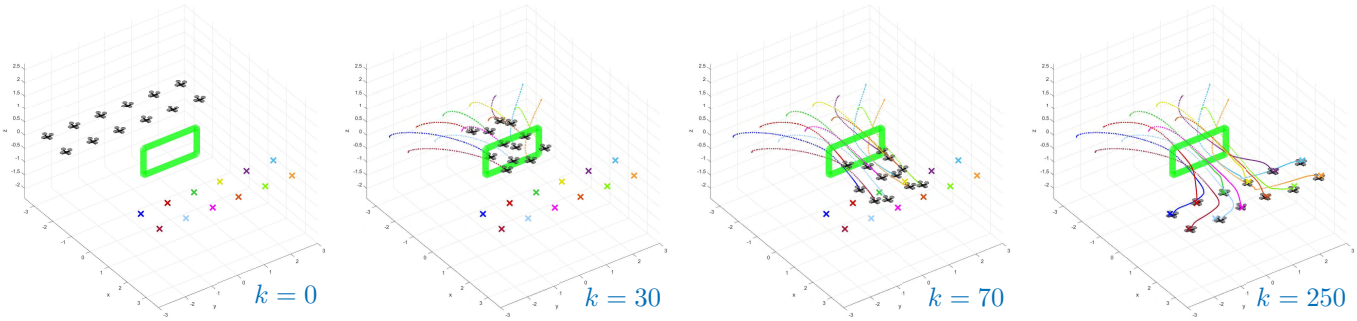

Fig. 9: Multi-UAV "gate" task with MD-DDP including 12 drones: Snapshots at different time instants.

horizon $K = 250$, the additional constraint is given by

$$\mathrm{y}^{\mathrm{l}}_{\mathrm{gate}} \leq \mathrm{y}_{i,k} \leq \mathrm{y}^{\mathrm{h}}_{\mathrm{gate}},\ k \in [\![k_1, k_2]\!], \tag{62a}$$

$$\mathrm{z}^{\mathrm{l}}_{\mathrm{gate}} \leq \mathrm{z}_{i,k} \leq \mathrm{z}^{\mathrm{h}}_{\mathrm{gate}},\ k \in [\![k_1, k_2]\!], \tag{62b}$$

with $\mathrm{y}^{\mathrm{l}}_{\mathrm{gate}} = -1.0\mathrm{m}$, $\mathrm{y}^{\mathrm{h}}_{\mathrm{gate}} = 1.0\mathrm{m}$, $\mathrm{z}^{\mathrm{l}}_{\mathrm{gate}} = -0.5\mathrm{m}$, $\mathrm{z}^{\mathrm{h}}_{\mathrm{gate}} = 0.5\mathrm{m}$, $k_1 = 30$ and $k_2 = 100$. Each drone has a neighborhood of size $|\mathcal{N}_i| = 6$. As illustrated in Fig. 9, with MD-DDP all drones reach their goals while passing through the gate and avoiding collisions. In the supplementary video, results for both MD-DDP and ND-DDP are provided.

Next, a task where a large-scale team of drones must move from an initial formation to a target one, is presented. Except for avoiding each other, the drones must also avoid spherical obstacles, which is enforced with constraints (58) in 3D. We set $|\mathcal{N}_i| = 9$. In Fig. 10, a task with 64 drones using MD-DDP is shown, where all drones successfully reach their targets while staying safe. The video includes results with 64 drones using ND-DDP and with 64 and 256 drones using MD-DDP.

C. Ablative Analysis

Here, we show how the improvements proposed in Section VI (Nesterov acceleration and decentralized penalty parameters adaptation), can increase the performance of the algorithms. For brevity, only a comparison for MD-DDP is presented, as the same patterns are observed for ND-DDP as well. The 256 cars formation task is used for all tests. Since we wish to investigate whether the improvements could allow for earlier termination, the algorithms are no longer terminated at maximum ADMM iterations, but once the total primal and dual residuals norms reach below the following thresholds

$$\|r^{\mathrm{pri}}_{b,\mathrm{total}}\|_2 \leq \epsilon^{\mathrm{pri}}_b, \quad \|r^{\mathrm{dual}}_{\mathrm{total}}\|_2 \leq \epsilon^{\mathrm{dual}}_b, \quad b = 1, 2, 3 \tag{63}$$

where $r^{\mathrm{pri}}_{b,\mathrm{total}} = \left[r^{\mathrm{pri}}_{b,1}; \ldots; r^{\mathrm{pri}}_{b,M}\right]$, $r^{\mathrm{dual}}_{b,\mathrm{total}} = \left[r^{\mathrm{dual}}_{b,1}; \ldots; r^{\mathrm{dual}}_{b,M}\right]$. The expressions for all "per-agent" residuals $r^{\mathrm{pri}}_{b,i}$, $r^{\mathrm{dual}}_{b,i}$, $i \in [\![1, M]\!]$, are provided in Appendix C. The threshold values are set to $\epsilon^{\mathrm{pri}}_1 = 5$, $\epsilon^{\mathrm{pri}}_{\{2,3\}} = 10$ and $\epsilon^{\mathrm{dual}}_1 = 50$, $\epsilon^{\mathrm{dual}}_{\{2,3\}} = 10^3$.

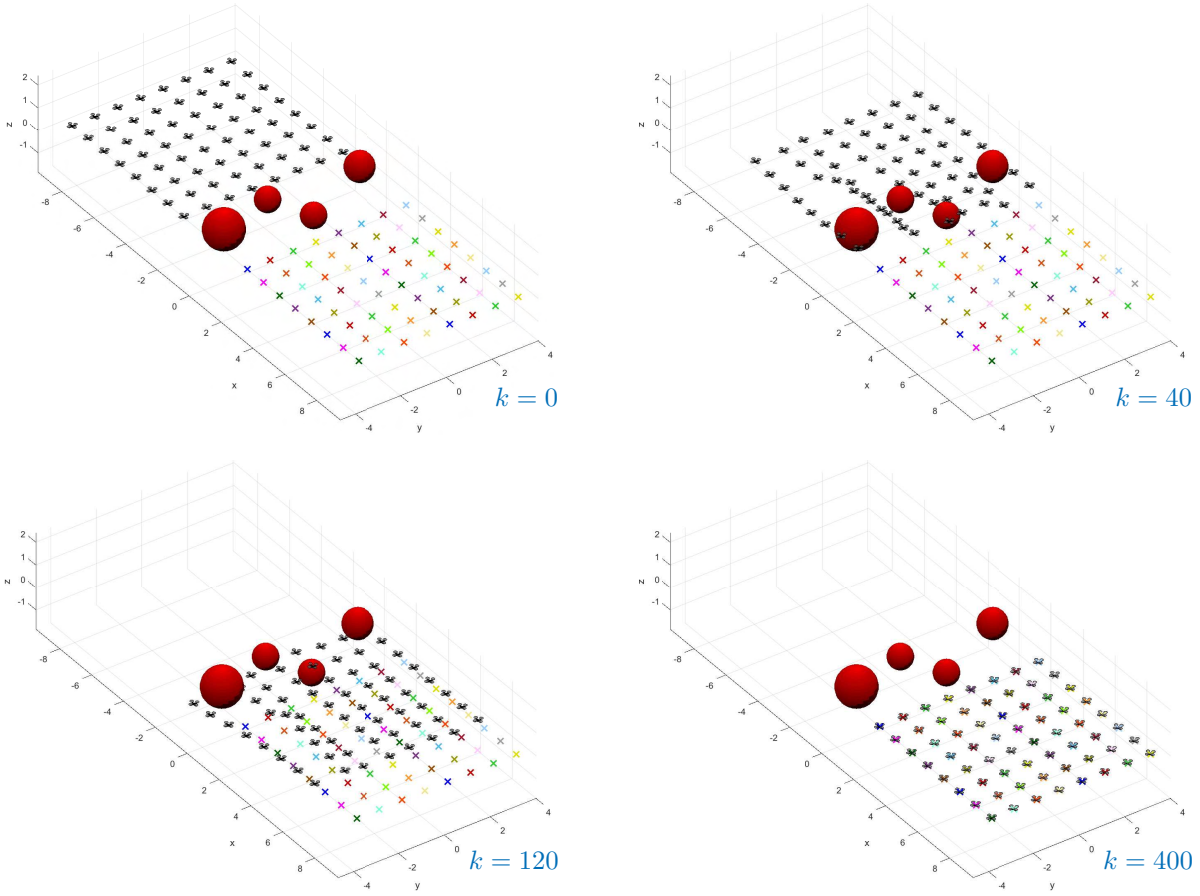

Fig. 10: Multi-UAV formation task with MD-DDP including 64 drones: Snapshots at different time instants.

Method	η	Iterations	+/- %
Vanilla MD-DDP	-	127	-
Nesterov accelerated MD-DDP	0.05	119	-6.3%
	0.1	116	-8.7%
	0.15	108	-15.0%
	0.2	**101**	**-20.5%**
	0.25	104	-18.1%
	0.3	113	-11.0%
	0.35	122	-3.9%
	0.4	136	+7.1%

TABLE I: Comparison of ADMM iterations between vanilla MD-DDP and Nesterov accelerated MD-DDP for different values of the parameter η.

Method	n_{adapt}	Iterations	+/- %
Without adaptation	-	184	-
Centralized adaptation	1	172	-6.5%
	5	159	-13.6%
	10	131	-28.8%
	20	138	-25.0%
Decentralized adaptation	1	171	-7.1%
	5	151	-17.9%
	10	**123**	**-33.2%**
	20	130	-29.3%

TABLE II: Comparison of ADMM iterations between MD-DDP with decentralized, centralized or no adaptation for the penalty parameters.

Initially, a comparison between vanilla and Nesterov accelerated MD-DDP is presented. Table I shows the amount of ADMM iterations required for the criteria (63) to be met, for different values of η. For the accelerated algorithm to achieve a computational benefit, the parameter η must be within the region $[0.05, 0.35]$. The iterations percentage decrease varies from -3.9% to -20.5%, with the largest reduction observed for $\eta = 0.2$. Therefore, the results show that Nesterov acceleration can indeed improve performance, with the appropriate tuning of parameter η.

Subsequently, the advantages of the proposed decentralized penalty parameter adaptation scheme are illustrated. MD-DDP using the proposed scheme is compared against MD-DDP without adaptation and with a centralized adaptation scheme as in [44, Section 3.4.1] using the total residuals norms of the global problem $\|r^{\text{pri}}_{b,\text{total}}\|_2$, $\|r^{\text{dual}}_{b,\text{total}}\|_2$ in the criteria. We use $\chi^{\text{incr}} = \chi^{\text{decr}} = 2$, $\sigma_1^{\text{incr}} = \sigma_2^{\text{incr}} = 1/200$, $\sigma_3^{\text{incr}} = 1/20$, $\sigma_1^{\text{decr}} = \sigma_2^{\text{decr}} = 1/50$ and $\sigma_3^{\text{decr}} = 1/5$ for both the centralized and decentralized schemes. To verify the adaptation capabilities of the proposed approach, we initialize the penalty parameter matrices with $\boldsymbol{T}_i^0 = \boldsymbol{R}_i$, $\boldsymbol{P}_i^0 = \boldsymbol{Q}_i$, $\boldsymbol{M}_i^0 = \text{bdiag}(\{\boldsymbol{Q}_j\}_{j\in\mathcal{N}_i})$ which are substantially different than the values used in Section VII-A after proper tuning. Table II shows the required ADMM iterations until convergence for each case, while also varying how frequently we perform an adaptation, i.e., n_{adapt}. The results indicate that both the centralized and decentralized schemes will adapt the parameters such that convergence is achieved faster than without adaptation. The best performance

Method	$M=2$	$M=4$	$M=8$	$M=16$	$M=32$	$M=64$	$M=256$	$M=1024$	$M=4096$
ND-DDP (Proposed)	36s	1m 11s	2m 58s	6m 4s	7m 35s	8m 18s	10m 38s	-	-
MD-DDP (Proposed)	40s	48s	**1m 15s**	**1m 41s**	**1m 49s**	**1m 56s**	**2m 2s**	**2m 24s**	**2m 46s**
Semi-centralized DDP	53s	1m 41s	4m 32s	8m 56s	18m 21s	37m 4s	2h 42m	-	-
Centralized AL-DDP	**11s**	**34s**	2m 3s	5m 2s	25m 56s	2h 47m	-	-	-
Decentralized SQP	8m 28s	21m 51s	42m 37s	1h 47m	-	-	-	-	-
Centralized SQP	2m 8s	7m 30s	31m 45s	2h 32m	-	-	-	-	-

TABLE III: Computational times comparison between ND-DDP, MD-DDP, Semi-centralized DDP, Centralized AL-DDP, Decentralized SQP and Centralized SQP for an increasing number of agents M in the multi-car formation task.

is achieved using the decentralized scheme with $n_{\text{adapt}} = 10$, while in all cases, it performs either better or equally well with the centralized one. Therefore, the proposed scheme not only maintains the decentralized nature of the algorithms, but also demonstrates a favorable performance against the equivalent centralized one.

D. Scalability Comparison with Other Methods

Next, the scalability of ND-DDP and MD-DDP is compared against centralized DDP [38], semi-centralized DDP [54], and centralized/decentralized SQP [10]. All simulations were performed in Matlab R2021b using a laptop computer with an 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz and a 32GB RAM memory. For the SQP methods, the NLP solver `fmincon` of the Matlab Optimization Toolbox is used. In centralized DDP and SQP, Problem 1 is directly provided to the solvers. For decentralized SQP, we solve the Step 1 problems of ND-DDP with SQP instead of AL-DDP. The computational times for decentralized methods are measured by considering the "slowest" agent at every step. ND-DDP, semi-centralized DDP and decentralized SQP are terminated after $N = 50$ ADMM iterations and MD-DDP after $N = 200$ iterations.

All algorithms were tested on the multi-vehicle formation task for an increasing number of agents M. The computational times of all methods are displayed in Table III. For $M = 2, 4, 8$, we used $|\mathcal{N}_i| = 2, 3, 5$, respectively, while for $M \geq 16$, we set $|\mathcal{N}_i| = 9$. As M increases, MD-DDP and ND-DDP demonstrate a superior scalability against the other methods. As predicted by its computational complexity (Remark 8), the computational demands of MD-DDP do not increase significantly as M grows. The same applies for ND-DDP once the neighborhood size $|\mathcal{N}_i|$ (and thus $\tilde{q}_i$) remains fixed. The SQP methods present significantly higher computational demands which is mainly attributed to the fact that all states/controls are stacked into one high-dimensional vector for the entire time horizon K. On the other hand, DDP scales linearly w.r.t. K, which explains its favorable performance for long time horizons. Nevertheless, centralized AL-DDP still demonstrates an inferior scalability compared to ND-DDP and MD-DDP, since it has a cubic complexity in M (see Remark 4). In addition, superior scalability compared to a previous (semi-centralized) distributed DDP approach [54] is also shown. This stems from the fact that in the latter method, the updates in the second ADMM block cannot be distributed computationally among the agents as they address the global problem including all agents. This leads to more computationally demanding optimization subproblems as the scale of the multi-agent system increases. Furthermore, note that semi-centralized DDP requires the communication of all agents with a central node during every ADMM iteration - and thus cannot be considered as fully decentralized. Finally, note that the Step 2 computations in MD-DDP were performed sequentially here (and not in parallel; see Remark 6), so the capabilities of the method were not fully exploited.

E. Limitations

Although the proposed algorithms have demonstrated substantial success and applicability for a large variety of multi-agent problems, we also note limitations of the current frameworks that are to be addressed in future work. First, despite the fact that ND-DDP and MD-DDP greatly outperform the SQP methods in computational efficiency, we have observed that decentralized SQP is more effective in generating feasible trajectories as the difficulty/complexity of the tasks increases. This is due to the fact that SQP handles the local constraints as hard ones - in contrast with ND-DDP and MD-DDP where these constraints are soft - thus facilitating reaching to a consensus. Second, as shown in Remarks 4 and 8, the computational complexity of ND-DDP and MD-DDP depends on the control dimension of the local problems. Therefore, if the local problems - and thus the matrix inversions $Q_{\boldsymbol{uu},i,k}^{-1}$ required for (7) - are high-dimensional, then computational demands will increase. To address this limitation, we wish to also explore using first-order local trajectory optimizers in future work. Finally, the proposed methods currently rely on the assumption that neighborhoods are fixed throughout the time horizon. In practice, such an assumption might require for example predicting which agents would be subject to collisions, losing communication, etc. In the future, we wish to extend ND-DDP and MD-DDP in a model predictive control (MPC) setup which would allow the algorithms to account for time-varying neighborhoods and more dynamic environments.

VIII. Hardware Experiments

To illustrate their applicability on real systems, ND-DDP and MD-DDP were employed on the Robotarium platform [56] for controlling a multi-robot team. All robots have unicycle dynamics with states $\boldsymbol{x}_{i,k} = \begin{bmatrix} \mathrm{x}_{i,k}; & \mathrm{y}_{i,k}; & \theta_{i,k} \end{bmatrix} \in \mathbb{R}^3$ and control inputs $\boldsymbol{u}_{i,k} = \begin{bmatrix} v_{i,k}; & \omega_{i,k} \end{bmatrix} \in \mathbb{R}^2$, where $(\mathrm{x}_{i,k}, \mathrm{y}_{i,k})$ is the 2D position, $\theta_{i,k}$ is the angle and $v_{i,k},\ \omega_{i,k}$ are the linear and angular velocities of the i-th robot at time k. The time step and time horizon are set to $dt = 0.033$s and $K = 900$. Constraints on the controls of the robots appear through bounding their wheel speeds,

$$-\boldsymbol{d}_{\max} \leq \boldsymbol{C}\boldsymbol{u}_i \leq \boldsymbol{d}_{\max}, \tag{64}$$

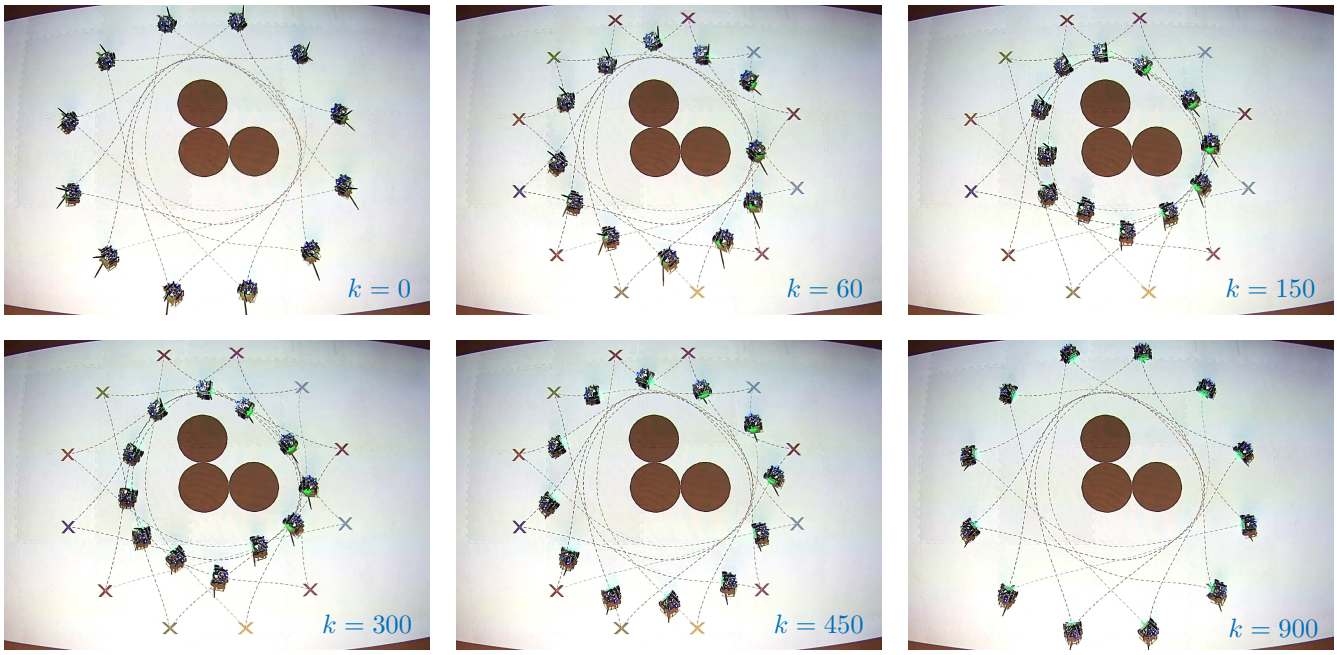

Fig. 11: Hardware experiment: "Circle Swapping" task with three unsymmetrical obstacles in the center, solved with MD-DDP (12 robots). Snapshots at different time instants.

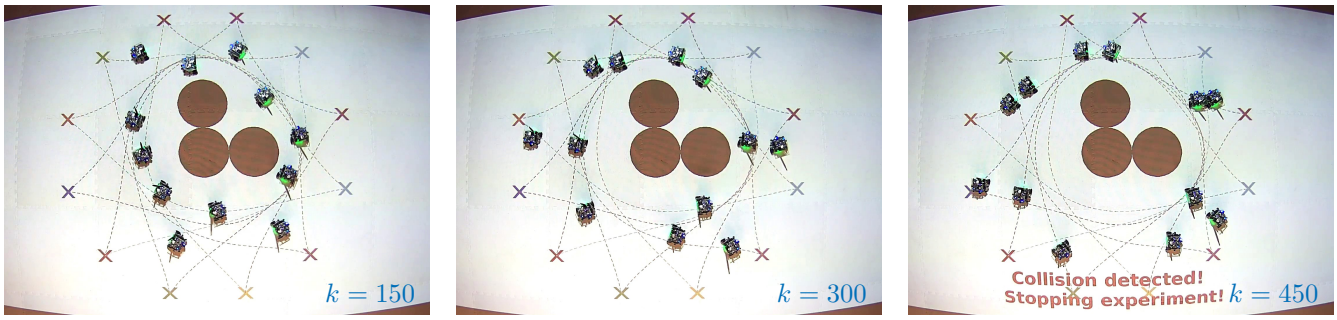

Fig. 12: Hardware experiment: "Circle Swapping" task with MD-DDP (12 robots) using open-loop policies. Without any feedback, the robots end up colliding and the experiment is terminated.

with

$$\boldsymbol{C} = \frac{1}{2R}\begin{bmatrix} 2 & L \\ 2 & -L \end{bmatrix}, \quad \boldsymbol{d}_{\max} = \begin{bmatrix} v_{\text{wheel, max}} \\ v_{\text{wheel, max}} \end{bmatrix}, \tag{65}$$

where $R = 0.016$m and $L = 0.11$m are the wheel radius and axle length of each robot, respectively, while $v_{\text{wheel, max}} = 12.5$ rad/s is the maximum wheel speed. Moreover, the robots are subject to the collision avoidance and connectivity maintenance constraints (58)-(60) with $d_o = 0.15$m, $d_{\text{col}} = 0.2$m and $d_{\text{con}} = 0.5$m. Each robot's cost is quadratic as in (53) with $\boldsymbol{Q}_i = \text{diag}(100, 100, 0)$, $\boldsymbol{R}_i = \text{diag}(100, 10)$ and $\boldsymbol{Q}_i^{\text{f}} = \text{diag}(300, 300, 30)$. To enhance the robustness of the proposed methods against model uncertainty, we utilize the feedback terms provided by DDP and apply the following closed-loop policies

$$\boldsymbol{u}_{i,k} = \boldsymbol{u}_{i,k}^* + \boldsymbol{K}_{i,k}(\boldsymbol{x}_{i,k}^{\text{meas}} - \boldsymbol{x}_{i,k}^*) \tag{66}$$

where $\boldsymbol{x}_{i,k}^*, \boldsymbol{u}_{i,k}^*$ are the computed optimal states and controls, $\boldsymbol{x}_{i,k}^{\text{meas}}$ are the measured states and $\boldsymbol{K}_{i,k}$ are the computed feedback gains of the i-th robot for time k. The methods are tested for the same tasks as in Section VII-A. Results for both ND-DDP and MD-DDP are provided in the supplementary video, while the figures displayed in the main paper correspond to the latter.

The first task that is presented is the "circle swapping" one with 12 robots and three unsymmetrical obstacles in the center (Fig. 11). All robots successfully reach to their targets while avoiding collisions with each other and the obstacles. In Fig. 12, the experiment is repeated using the open-loop policies $\boldsymbol{u}_{i,k} = \boldsymbol{u}_{i,k}^*$, instead of the closed-loop ones (66). Without any use of feedback, the robots are unable to follow their optimal state trajectories, which leads to unsafe behavior as collisions occur. This highlights the advantage of DDP methods for providing feedback control policies explicitly, in addition to the optimal control and state sequences.

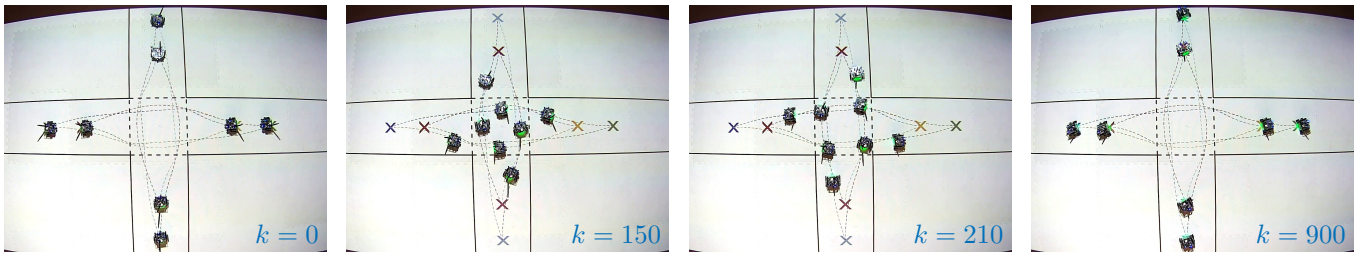

Fig. 13: Hardware experiment: "Intersection" task with MD-DDP (8 robots). Snapshots at different time instants.

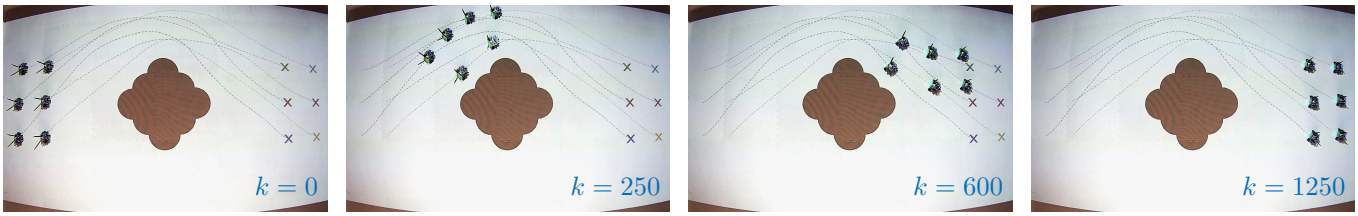

Fig. 14: Hardware experiment: Communication maintenance task with MD-DDP (6 robots). Snapshots at different time instants.

Subsequently, we demonstrate 8 robots performing the "intersection" task. As shown in Fig. 13, the robots successfully cross the intersection while staying safe and within their lanes. Next, 6 robots are required to complete the communication maintenance" task. As illustrated in Fig. 14, all robots are able to reach to their targets while staying close to each other and avoiding collisions. In Fig. 15, the "bottleneck" task with 8 robots is presented. The robots are able to pass through the gap while avoiding collisions and reach to their goals. Finally, a formation task with 20 robots (maximum available number of robots in the Robotarium) is also shown in Fig. 16. All robots are able to reach their desired targets while avoiding the obstacles and each other.

IX. Conclusion

In this paper, we propose two fully distributed DDP methods based on ADMM. The first one, ND-DDP, is a three-level architecture that utilizes ADMM for consensus, an AL approach for local constraints and DDP as the local trajectory optimizer. The second one, MD-DDP, is a two-level framework that merges the consensus and local constraints levels by treating them both with ADMM, to further improve computational efficiency. Both methods are successfully tested in simulation on a wide variety of multi-car and multi-UAV control problems. To our best knowledge, ND-DDP and MD-DDP are the first fully decentralized DDP methods that exhibit a scalability to multi-robot systems of such a large scale. We also showcase the applicability of the algorithms on hardware systems by successfully employing them on a multi-robot platform.

Future directions include further theoretical and algorithmic investigations on the two methods. From a theoretical perspective, we wish to establish conditions for the convergence of both schemes. On the algorithmic side, both architectures can be extended to the cases where uncertainty is represented using Gaussian processes [76] or generalized polynomial chaos theory [77], [78]. Such extensions will result into a new set of distributed optimization algorithms that are capable of handling epistemic and aleatoric uncertainty. Another interesting direction could incorporate a min-max differential game theoretic formulation of DDP [79], to address disturbances with an adversarial nature. In addition, generalizations of DDP such as Maximum Entropy DDP [80] and Parameterized DDP [81] can further expand the capabilities of the ND-DDP and MD-DDP architectures, in terms of avoiding local minima and handling parametric uncertainty, respectively. Finally, to further increase the applicability of the methods, we aspire to investigate their application in a MPC setting to account for time-vaying neighborhoods and more dynamic environments.

Appendix A
Local AL-DDP Updates in ND-DDP

Additional details about the local AL-DDP updates in ND-DDP are given here. Each term $C_{i,k}(\boldsymbol{x}^{\mathrm{a}}_{i,k}, \boldsymbol{u}^{\mathrm{a}}_{i,k})$ is of the form

$$C_{i,k}(\boldsymbol{x}^{\mathrm{a}}_{i,k}, \boldsymbol{u}^{\mathrm{a}}_{i,k}) = \sum_g P\big(w_{k,g}, \beta_{k,g}, s^{\mathrm{a}}_{i,k,g}(\boldsymbol{x}^{\mathrm{a}}_{i,k}, \boldsymbol{u}^{\mathrm{a}}_{i,k})\big) \quad (67)$$

where g is the index of the g-th component of the constraint $\boldsymbol{s}^{\mathrm{a}}_{i,k}(\boldsymbol{x}^{\mathrm{a}}_{i,k}, \boldsymbol{u}^{\mathrm{a}}_{i,k}) \leq 0$. The function P is selected to be

$$P = \frac{(w_{k,g})^2}{\beta_{k,g}} \phi\left(\frac{\beta_{k,g}}{w_{k,g}} s^{\mathrm{a}}_{i,k,g}(\boldsymbol{x}^{\mathrm{a}}_{i,k}, \boldsymbol{u}^{\mathrm{a}}_{i,k})\right) \quad (68)$$

where the function $\phi(t)$ is a smooth approximation to the Powell Hestenes-Rockafellar method [82] given by

$$\phi(t) = \begin{cases} \frac{1}{2}t^2 + t, & \text{if } t \geq -\frac{1}{2} \\ -\frac{1}{4}\log(-2t), & \text{otherwise.} \end{cases} \quad (69)$$

The variables $w_{k,g}$ and parameters $\beta_{k,g}$ are updated with

$$w^{l+1}_{k,g} = w^{l}_{k,g} \phi\left(\frac{\beta_{k,g}}{w_{k,g}} s^{\mathrm{a}}_{i,k,g}(\boldsymbol{x}^{\mathrm{a}}_{i,k}, \boldsymbol{u}^{\mathrm{a}}_{i,k})\right) \quad (70)$$

and $\beta^{l+1}_{k,g} = \beta^{\mathrm{coeff}} \beta^{l}_{k,g}$, where $\beta^{\mathrm{coeff}} = 1$ if sufficient improvement on the constraint satisfaction has been achieved and $\beta^{\mathrm{coeff}} > 1$, otherwise.

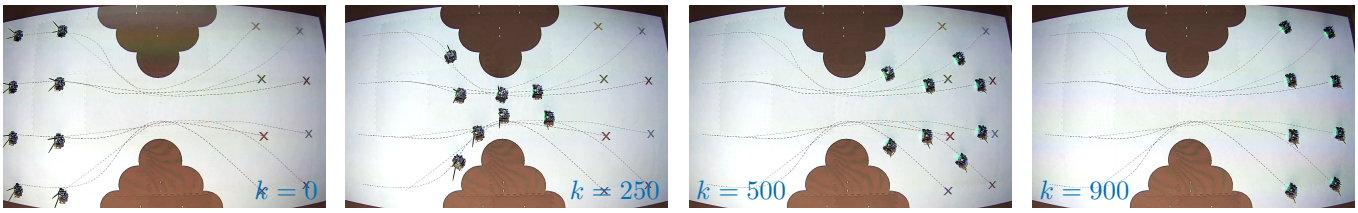

Fig. 15: Hardware experiment: "Bottleneck" task with MD-DDP (8 robots). Snapshots at different time instants.

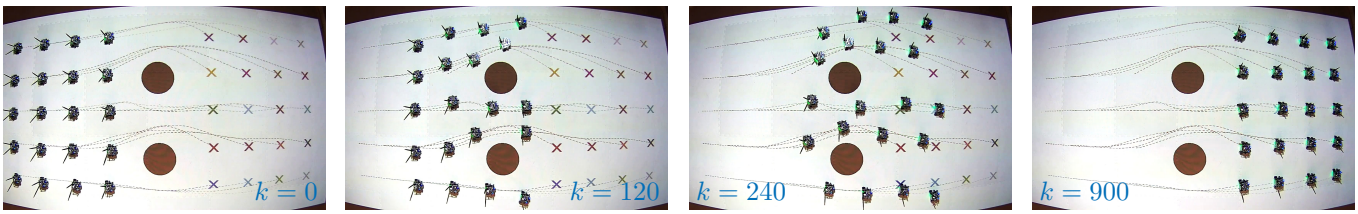

Fig. 16: Hardware experiment: Formation task with MD-DDP (20 robots). Snapshots at different time instants.

APPENDIX B
MD-DDP UPDATES WITH DECENTRALIZED PENALTY PARAMETER ADAPTATION

Here, we provide the MD-DDP updates when the penalty parameter adaptation proposed in Section VI-B is used. The Step 1 DDP updates will be given by (39), but with costs

$$\hat{\ell}_{i,k}(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}) = \ell_{i,k}(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}) + \frac{1}{2}\left\|\boldsymbol{x}_{i,k} - \tilde{\boldsymbol{x}}_{i,k}\right\|^2_{\boldsymbol{P}_i} + \boldsymbol{\lambda}^{\mathrm{T}}_{i,k}\boldsymbol{x}_{i,k} + \frac{1}{2}\left\|\boldsymbol{u}_{i,k} - \tilde{\boldsymbol{u}}_{i,k}\right\|^2_{\boldsymbol{T}_i} + \boldsymbol{\xi}^{\mathrm{T}}_{i,k}\boldsymbol{u}_{i,k},$$

$$\hat{\phi}_i(\boldsymbol{x}_{i,K}) = \phi_i(\boldsymbol{x}_{i,K}) + \frac{1}{2}\left\|\boldsymbol{x}_{i,K} - \tilde{\boldsymbol{x}}_{i,K}\right\|^2_{\boldsymbol{P}_i} + \boldsymbol{\lambda}^{\mathrm{T}}_{i,K}\boldsymbol{x}_{i,K}.$$

In Step 2, the safe state updates will be provided by

$$\tilde{\boldsymbol{x}}^{\mathrm{a},n+1}_{i,k} = \arg\min \frac{1}{2}\left\|\boldsymbol{x}_{i,k} - \tilde{\boldsymbol{x}}_{i,k}\right\|^2_{\boldsymbol{P}_i} - \boldsymbol{\lambda}^{\mathrm{T}}_{i,k}\tilde{\boldsymbol{x}}_{i,k} + \frac{1}{2}\left\|\tilde{\boldsymbol{x}}^{\mathrm{a}}_{i,k} - \boldsymbol{z}^{\mathrm{a}}_{i,k}\right\|^2_{\boldsymbol{M}_i} + \boldsymbol{y}^{\mathrm{T}}_i\tilde{\boldsymbol{x}}^{\mathrm{a}}_i$$
$$\text{s.t.} \quad \boldsymbol{g}_{i,k}(\tilde{\boldsymbol{x}}_{i,k}) \leq 0, \ \boldsymbol{h}^{\mathrm{a}}_{i,k}(\tilde{\boldsymbol{x}}^{\mathrm{a}}_{i,k}) \leq 0, \tag{71}$$

while the safe control updates will take the following form

$$\tilde{\boldsymbol{u}}^{n+1}_{i,k} = \arg\min \frac{1}{2}\left\|\boldsymbol{u}_{i,k} - \tilde{\boldsymbol{u}}_{i,k}\right\|^2_{\boldsymbol{T}_i} - \boldsymbol{\xi}^{\mathrm{T}}_{i,k}\tilde{\boldsymbol{u}}_{i,k}$$
$$\text{s.t.} \quad \boldsymbol{b}_{i,k}(\tilde{\boldsymbol{u}}_{i,k}) \leq 0. \tag{72}$$

The Step 3 global updates will be given by

$$\boldsymbol{z}^{n+1}_{i,k} = \Big(\sum_{j\in\mathcal{P}_i}\boldsymbol{M}_j\Big)^{-1}\sum_{j\in\mathcal{P}_i}\boldsymbol{M}_j\tilde{\boldsymbol{x}}^{i,n+1}_{j,k}. \tag{73}$$

Finally, the dual updates will be

$$\boldsymbol{\xi}^{n+1}_{i,k} = \boldsymbol{\xi}^{n}_{i,k} + \boldsymbol{T}_i(\boldsymbol{u}^{n+1}_{i,k} - \tilde{\boldsymbol{u}}^{n+1}_{i,k}), \tag{74}$$
$$\boldsymbol{\lambda}^{n+1}_{i,k} = \boldsymbol{\lambda}^{n}_{i,k} + \boldsymbol{P}_i(\boldsymbol{x}^{n+1}_{i,k} - \tilde{\boldsymbol{x}}^{n+1}_{i,k}), \tag{75}$$
$$\boldsymbol{y}^{n+1}_{i,k} = \boldsymbol{y}^{n}_{i,k} + \boldsymbol{M}_i(\tilde{\boldsymbol{x}}^{\mathrm{a},n+1}_{i,k} - \boldsymbol{z}^{\mathrm{a},n+1}_{i,k}). \tag{76}$$

For $\boldsymbol{T}_i = \mathrm{diag}(\tau, \ldots, \tau)$, $\boldsymbol{P}_i = \mathrm{diag}(\rho, \ldots, \rho)$ and $\boldsymbol{M}_i = \mathrm{diag}(\mu, \ldots, \mu)$, the vanilla MD-DDP updates are recovered.

APPENDIX C
PENALTY PARAMETERS ADAPTATION DETAILS

Detailed expressions for the adaptation rules of the scheme presented in Section VI-B are provided here. First, the "per-agent" primal and dual residuals of the MD-DDP algorithm at iteration n are given by

$$r^{\mathrm{pri},n}_{1,i} = \boldsymbol{u}^n_i - \tilde{\boldsymbol{u}}^n_i, \qquad r^{\mathrm{dual},n}_{1,i} = \bar{\boldsymbol{T}}_i(\tilde{\boldsymbol{u}}^n_i - \tilde{\boldsymbol{u}}^{n-1}_i),$$
$$r^{\mathrm{pri},n}_{2,i} = \boldsymbol{x}^n_i - \tilde{\boldsymbol{x}}^n_i, \qquad r^{\mathrm{dual},n}_{2,i} = \bar{\boldsymbol{P}}_i(\tilde{\boldsymbol{x}}^n_i - \tilde{\boldsymbol{x}}^{n-1}_i)$$
$$r^{\mathrm{pri},n}_{3,i} = \tilde{\boldsymbol{x}}^{\mathrm{a},n}_i - \boldsymbol{z}^{\mathrm{a},n}_i, \qquad r^{\mathrm{dual},n}_{3,i} = \bar{\boldsymbol{M}}_i(\boldsymbol{z}^{\mathrm{a},n}_i - \boldsymbol{z}^{\mathrm{a},n-1}_i),$$

where $\bar{\boldsymbol{T}}_i = \mathrm{bdiag}(\boldsymbol{T}_i, \ldots, \boldsymbol{T}_i) \in \mathbb{R}^{Kq_i \times Kq_i}$, $\bar{\boldsymbol{P}}_i = \mathrm{bdiag}(\boldsymbol{P}_i, \ldots, \boldsymbol{P}_i) \in \mathbb{R}^{(K+1)p_i \times (K+1)p_i}$ and $\bar{\boldsymbol{M}}_i = \mathrm{bdiag}(\boldsymbol{M}_i, \ldots, \boldsymbol{M}_i) \in \mathbb{R}^{(K+1)\tilde{p}_i \times (K+1)\tilde{p}_i}$. For a detailed explanation on how such expressions are derived for ADMM methods, see [44, Section 3.3]. Given the residuals, we propose the following update rules

$$a^{n+1}_{b,i} = \begin{cases} \chi^{\mathrm{incr}} a^n_{b,i} & \text{if } \|r^{\mathrm{pri},n}_{b,i}\|_2 > \sigma^{\mathrm{incr}}_b \|r^{\mathrm{dual},n}_{b,i}\|_2 \\ a^n_{b,i}/\chi^{\mathrm{decr}} & \text{if } \|r^{\mathrm{pri},n}_{b,i}\|_2 < \sigma^{\mathrm{decr}}_b \|r^{\mathrm{dual},n}_{b,i}\|_2 \\ a^n_{b,i} & \text{otherwise,} \end{cases} \tag{77}$$

for $b = 1, 2, 3$ with $\sigma^{\mathrm{incr}}_b > \sigma^{\mathrm{decr}}_b > 0$, $\chi^{\mathrm{incr}} > 1$ and $\chi^{\mathrm{decr}} > 1$. Lower and upper bounds $a^n_{i,b,\min}$, $a^n_{i,b,\max}$ $\forall b \in \{1, 2, 3\}$ can also be specified. The rules are of similar form with [44, Section 3.4.1], [62] but in a "per-agent" format.

REFERENCES

[1] B. Chalaki and A. A. Malikopoulos, "Time-optimal coordination for connected and automated vehicles at adjacent intersections," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–16, 2021.
[2] W. Xiao and C. G. Cassandras, "Decentralized optimal merging control for connected and automated vehicles with safety constraint guarantees," *Automatica*, vol. 123, p. 109333, 2021.
[3] K. M. Kabore and S. Güler, "Distributed formation control of drones with onboard perception," *IEEE/ASME Transactions on Mechatronics*, pp. 1–11, 2021.
[4] P. Yu and D. V. Dimarogonas, "Distributed motion coordination for multirobot systems under LTL specifications," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 1047–1062, 2022.

[5] W. Zhang, M. Kamgarpour, D. Sun, and C. J. Tomlin, "Decentralized flight path planning for air traffic management," in *Proceedings of the 2011 American Control Conference*, 2011, pp. 2137–2142.

[6] A. E. Turgut, H. Çelikkanat, F. Gökçe, and E. Şahin, "Self-organized flocking in mobile robot swarms," *Swarm Intelligence*, vol. 2, no. 2, pp. 97–120, 2008.

[7] P. Zhu, C. Liu, and S. Ferrari, "Adaptive online distributed optimal control of very-large-scale robotic systems," *IEEE Transactions on Control of Network Systems*, vol. 8, no. 2, pp. 678–689, 2021.

[8] A. Abdulghafoor and E. Bakolas, "Multi-agent distributed optimal control for tracking large-scale multi-target systems in dynamic environments," 2022.

[9] Z. Zhou and H. Xu, "A novel mean-field-game-type optimal control for very large-scale multiagent systems," *IEEE Transactions on Cybernetics*, vol. 52, no. 6, pp. 5197–5208, 2022.

[10] M. Diehl, H. J. Ferreau, and N. Haverbeke, *Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 391–417.

[11] O. Von Stryk and R. Bulirsch, "Direct and indirect methods for trajectory optimization," *Annals of operations research*, vol. 37, no. 1, pp. 357–373, 1992.

[12] G. Wagner and H. Choset, "M*: A complete multirobot path planning algorithm with performance bounds," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 3260–3267.

[13] T. Standley and R. Korf, "Complete algorithms for cooperative pathfinding problems," in *IJCAI*. Citeseer, 2011, pp. 668–673.

[14] M. Čáp, P. Novák, J. Vokrínek, and M. Pěchouček, "Multi-agent rrt: sampling-based cooperative pathfinding," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 2013, pp. 1263–1264.

[15] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[16] A. Nowé, P. Vrancx, and Y.-M. De Hauwere, "Game theory and multi-agent reinforcement learning," *Reinforcement Learning: State-of-the-Art*, pp. 441–470, 2012.

[17] D. V. Dimarogonas and E. Frazzoli, "Analysis of decentralized potential field based multi-agent navigation via primal-dual lyapunov theory," in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 1215–1220.

[18] M. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1257–1270, 2006.

[19] P. Robuffo Giordano, A. Franchi, C. Secchi, and H. H. Bülthoff, "A passivity-based decentralized strategy for generalized connectivity maintenance," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 299–323, 2013.

[20] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 285–292.

[21] L. Riegger, M. Carlander, N. Lidander, N. Murgovski, and J. Sjöberg, "Centralized mpc for autonomous intersection crossing," in *2016 IEEE 19th international conference on intelligent transportation systems (ITSC)*. IEEE, 2016, pp. 1372–1377.

[22] K. Mombaur, "Using optimization to create self-stable human-like running," *Robotica*, vol. 27, no. 3, pp. 321–330, 2009.

[23] D. H. Jacobson and D. Q. Mayne, *Differential dynamic programming*. Elsevier Publishing Company, 1970, no. 24.

[24] J. Morimoto, G. Zeglin, and C. Atkeson, "Minimax differential dynamic programming: application to a biped walking robot," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 2, 2003, pp. 1927–1932 vol.2.

[25] V. Kumar, E. Todorov, and S. Levine, "Optimal control with learned local models: Application to dexterous manipulation," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 378–383.

[26] R. Budhiraja, J. Carpentier, C. Mastalli, and N. Mansard, "Differential dynamic programming for multi-phase rigid contact dynamics," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, 2018, pp. 1–9.

[27] M. D. Houghton, A. B. Oshin, M. J. Acheson, E. A. Theodorou, and I. M. Gregory, "Path planning: Differential dynamic programming and model predictive path integral control on vtol aircraft," in *AIAA SCITECH 2022 Forum*, 2022, p. 0624.

[28] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proceedings of the 2005, American Control Conference, 2005.*, 2005, pp. 300–306 vol. 1.

[29] L.-Z. Liao and C. Shoemaker, "Convergence in unconstrained discrete-time differential dynamic programming," *IEEE Transactions on Automatic Control*, vol. 36, no. 6, pp. 692–706, 1991.

[30] S. Yakowitz and B. Rutherford, "Computational aspects of discrete-time optimal control," *Applied Mathematics and Computation*, vol. 15, no. 1, pp. 29–45, 1984.

[31] L.-z. Liao and C. A. Shoemaker, *Advantages of differential dynamic programming over Newton's method for discrete-time optimal control problems*. Cornell University, 1992.

[32] D. M. Murray and S. J. Yakowitz, "Constrained differential dynamic programming and its application to multireservoir control," *Water Resources Research*, vol. 15, no. 5, pp. 1017–1027, 1979.

[33] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 1168–1175.

[34] Z. Xie, C. K. Liu, and K. Hauser, "Differential dynamic programming with nonlinear constraints," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 695–702.

[35] A. Pavlov, I. Shames, and C. Manzie, "Interior point differential dynamic programming," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 6, pp. 2720–2727, 2021.

[36] W. Jallet, A. Bambade, N. Mansard, and J. Carpentier, "Constrained differential dynamic programming: A primal-dual augmented lagrangian approach," 2022.

[37] T. A. Howell, B. E. Jackson, and Z. Manchester, "ALTRO: A fast solver for constrained trajectory optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 7674–7679.

[38] Y. Aoyama, G. Boutselis, A. Patel, and E. A. Theodorou, "Constrained differential dynamic programming revisited," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 9738–9744.

[39] H. Almubarak, K. Stachowicz, N. Sadegh, and E. A. Theodorou, "Safety embedded differential dynamic programming using discrete barrier states," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2755–2762, 2022.

[40] K. Cao, M. Cao, S. Yuan, and L. Xie, "DIRECT: A differential dynamic programming based framework for trajectory generation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2439–2446, 2022.

[41] T. Kavuncu, A. Yaraneri, and N. Mehr, "Potential iLQR: A Potential-Minimizing Controller for Planning Multi-Agent Interactive Trajectories," in *Proceedings of Robotics: Science and Systems*, Virtual, July 2021.

[42] O. So, K. Stachowicz, and E. A. Theodorou, "Multimodal maximum entropy dynamic games," *arXiv preprint arXiv:2201.12925*, 2022.

[43] Y. Wang and K. Tsumura, "Preconditioned distributed trajectory optimization algorithm using differential dynamic programming," in *2020 59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 2985–2991.

[44] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[45] A. D. Saravanos, A. Tsolovikos, E. Bakolas, and E. Theodorou, "Distributed Covariance Steering with Consensus ADMM for Stochastic Multi-Agent Systems," in *Proceedings of Robotics: Science and Systems*, Virtual, July 2021.

[46] M. A. Pereira, A. D. Saravanos, O. So, and E. A. Theodorou, "Decentralized Safe Multi-agent Stochastic Optimal Control using Deep FBSDEs and ADMM," in *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022.

[47] F. Rey, Z. Pan, A. Hauswirth, and J. Lygeros, "Fully decentralized admm for coordination and collision avoidance," in *2018 European Control Conference (ECC)*, 2018, pp. 825–830.

[48] O. Shorinwa, T. Halsted, and M. Schwager, "Scalable distributed optimization with separable variables in multi-agent networks," in *2020 American Control Conference (ACC)*, 2020, pp. 3619–3626.

[49] T. Halsted, O. Shorinwa, J. Yu, and M. Schwager, "A survey of distributed optimization methods for multi-robot systems," *arXiv preprint arXiv:2103.12840*, 2021.

[50] O. Shorinwa and M. Schwager, "Scalable collaborative manipulation with distributed trajectory planning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 9108–9115.

[51] ——, "Distributed contact-implicit trajectory optimization for collaborative manipulation," in *2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2021, pp. 56–65.
[52] W. Tang and P. Daoutidis, "Fast and stable nonconvex constrained distributed optimization: the ELLADA algorithm," *Optimization and Engineering*, pp. 1–43, 2021.
[53] X. Zhang, Z. Cheng, J. Ma, S. Huang, F. L. Lewis, and T. H. Lee, "Semidefinite relaxation-based ADMM for cooperative planning and control of connected autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 9240–9251, 2022.
[54] X. Zhang, Z. Cheng, J. Ma, L. Zhao, C. Xiang, and T. H. Lee, "Parallel collaborative motion planning with alternating direction method of multipliers," in *IECON 2021 – 47th Annual Conference of the IEEE Industrial Electronics Society*, 2021, pp. 1–6.
[55] S.-S. Park, Y. Min, J.-S. Ha, D.-H. Cho, and H.-L. Choi, "A distributed ADMM approach to non-myopic path planning for multi-target tracking," *IEEE Access*, vol. 7, pp. 163 589–163 603, 2019.
[56] S. Wilson, P. Glotfelter, L. Wang, S. Mayya, G. Notomista, M. Mote, and M. Egerstedt, "The robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems," *IEEE Control Systems Magazine*, vol. 40, no. 1, pp. 26–44, 2020.
[57] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.
[58] M. Tao and X. Yuan, "Recovering low-rank and sparse components of matrices from incomplete and noisy observations," *SIAM Journal on Optimization*, vol. 21, no. 1, pp. 57–81, 2011.
[59] W. Tang and P. Daoutidis, "Distributed nonlinear model predictive control through accelerated parallel ADMM," in *2019 American Control Conference (ACC)*, 2019, pp. 1406–1411.
[60] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk, "Fast alternating direction optimization methods," *SIAM Journal on Imaging Sciences*, vol. 7, no. 3, pp. 1588–1623, 2014.
[61] R. T. Rockafellar, "Monotone operators and the proximal point algorithm," *SIAM Journal on Control and Optimization*, vol. 14, no. 5, pp. 877–898, 1976.
[62] B. He, H. Yang, and S. Wang, "Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities," *Journal of Optimization Theory and applications*, vol. 106, no. 2, pp. 337–356, 2000.
[63] C. Song, S. Yoon, and V. Pavlovic, "Fast ADMM algorithm for distributed optimization with adaptive penalty," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, Feb. 2016.
[64] Z. Xu, G. Taylor, H. Li, M. A. T. Figueiredo, X. Yuan, and T. Goldstein, "Adaptive consensus ADMM for distributed optimization," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 3841–3850.
[65] K. Sun and X. A. Sun, "A two-level distributed algorithm for nonconvex constrained optimization," *arXiv preprint arXiv:1902.07654*, 2019.
[66] X. Zhang, J. Ma, Z. Cheng, S. Huang, C. W. de Silva, and T. H. Lee, "Improved hierarchical ADMM for nonconvex cooperative distributed model predictive control," *arXiv preprint arXiv:2011.00463*, 2020.
[67] M. Fortin and R. Glowinski, *Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems*. Elsevier, 2000.
[68] R. Glowinski and P. Le Tallec, *Augmented Lagrangian and operator-splitting methods in nonlinear mechanics*. SIAM, 1989.
[69] B. Jiang, S. Ma, and S. Zhang, "Alternating direction method of multipliers for real and complex polynomial optimization models," *Optimization*, vol. 63, no. 6, pp. 883–898, 2014.
[70] S. Magnússon, P. C. Weeraddana, and C. Fischione, "A distributed approach for the optimal power-flow problem based on ADMM and sequential convex approximations," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 3, pp. 238–253, 2015.
[71] T. Erseghe, "Distributed optimal power flow using ADMM," *IEEE Transactions on Power Systems*, vol. 29, no. 5, pp. 2370–2380, 2014.
[72] Y. Shen, Z. Wen, and Y. Zhang, "Augmented lagrangian alternating direction method for matrix separation based on low-rank factorization," *Optimization Methods and Software*, vol. 29, no. 2, pp. 239–263, 2014.
[73] Z. Xu, S. De, M. Figueiredo, C. Studer, and T. Goldstein, "An empirical study of ADMM for nonconvex problems," *arXiv preprint arXiv:1612.03349*, 2016.
[74] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 1917–1922.
[75] T. Luukkonen, "Modelling and control of quadcopter," *Independent research project in applied mathematics, Espoo*, vol. 22, p. 22, 2011.
[76] Y. Pan, G. I. Boutselis, and E. A. Theodorou, "Efficient reinforcement learning via probabilistic trajectory optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 11, pp. 5459–5474, 2018.
[77] G. I. Boutselis, Y. Pan, and E. A. Theodorou, "Numerical trajectory optimization for stochastic mechanical systems," *SIAM Journal on Scientific Computing*, vol. 41, no. 4, pp. A2065–A2087, 2019.
[78] Y. Aoyama, A. D. Saravanos, and E. A. Theodorou, "Receding horizon differential dynamic programming under parametric uncertainty," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 3761–3767.
[79] W. Sun, Y. Pan, J. Lim, E. A. Theodorou, and P. Tsiotras, "Min-max differential dynamic programming: Continuous and discrete time formulations," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 12, pp. 2568–2580, 2018.
[80] O. So, Z. Wang, and E. A. Theodorou, "Maximum entropy differential dynamic programming," *arXiv preprint arXiv:2110.06451*, 2021.
[81] A. Oshin, M. Houghton, M. Acheson, I. Gregory, and E. Theodorou, "Parameterized Differential Dynamic Programming," in *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022.
[82] E. G. Birgin, R. A. Castillo, and J. M. Martínez, "Numerical comparison of augmented lagrangian algorithms for nonconvex problems," *Computational Optimization and Applications*, vol. 31, no. 1, pp. 31–55, 2005.

Augustinos D. Saravanos received his Diploma in Electrical and Computer Engineering from the University of Patras, Greece in 2019, and he is currently working toward the PhD in Machine Learning degree at Georgia Institute of Technology. He is also a recipient of the A. Onassis Foundation Scholarship since 2021. His research interests lie at the intersection of optimization, control theory and deep learning for achieving scalable and efficient algorithms for large-scale multi-agent systems in control and robotics.

Yuichiro Aoyama received his Bachelor of Engineering and Master of Information Science and Technology in Mechano-Informatics from the University of Tokyo in 2009 and 2011, respectively. After working as a mechanical design engineer at Komatsu Ltd., Tokyo, Japan, he is currently working toward a PhD at Georgia Institute of Technology. His research interests include trajectory optimization, and learning and controlling spatio-temporal systems.

Hongchang Zhu received the B.S. degree in mechanical engineering from the University of California, Berkely, Berkeley, CA, USA, in 2017, and the M.S. degree in mechanical engineering in 2020 from Georgia Institute of Technology, Atlanta, GA, USA, where he is currently working toward the PhD degree in robotics. His research interests include distributed control and planning for autonomous multi-robot systems, risk-aware motion planning, and obstacle avoidance.

Evangelos A. Theodorou is an Associate Professor with the Daniel Guggenheim School of Aerospace Engineering at Georgia Institute of Technology. He is also the director of the Autonomous Control and Decision Systems Laboratory, and he is affiliated with the Institute of Robotics and Intelligent Machines and the Center for Machine Learning Research at Georgia Institute of Technology. Dr. Theodorou holds a BS in Electrical Engineering, from the Technical University of Crete (TUC), Greece in 2001. He also holds three MSc degrees in Production Engineering from TUC in 2003, Computer Science and Engineering from University of Minnesota in 2007, and Electrical Engineering from the University of Southern California (USC) in 2010. In 2011, he graduated with his PhD in Computer Science from USC. From 2011 to 2013, he was a Postdoctoral Research Fellow with the department of Computer Science and Engineering, University of Washington. Dr. Theodorou is the recipient of the King-Sun Fu best paper award of the IEEE Transactions on Robotics in 2012 and recipient of several best paper awards and nominations in machine learning and robotics conferences. His theoretical research spans the areas of stochastic optimal control theory, machine learning, statistical physics and optimization.