Distributed Spatial-Temporal Trajectory Optimization for Unmanned-Aerial-Vehicle Swarm

Xiaobo Zheng *, Defu Lin †
Beijing Institute of Technology, Beijing 100081, China

Pan Tang ‡
Beijing Institute of Aerospace Systems Engineering, Beijing 100076, China

Shaoming He §
Beijing Institute of Technology, Beijing 100081, China

Swarm trajectory optimization problems are a well-recognized class of multi-agent optimal control problems with strong nonlinearity. However, the heuristic nature of needing to set the final time for agents beforehand and the time-consuming limitation of the significant number of iterations prohibit the application of existing methods to large-scale swarm of Unmanned Aerial Vehicles (UAVs) in practice. In this paper, we propose a spatial-temporal trajectory optimization framework that accomplishes multi-UAV consensus based on the Alternating Direction Multiplier Method (ADMM) and uses Differential Dynamic Programming (DDP) for fast local planning of individual UAVs. The introduced framework is a two-level architecture that employs Parameterized DDP (PDDP) as the trajectory optimizer for each UAV, and ADMM to satisfy the local constraints and accomplish the spatial-temporal parameter consensus among all UAVs. This results in a fully distributed algorithm called Distributed Parameterized DDP (D-PDDP). In addition, an adaptive tuning criterion based on the spectral gradient method for the penalty parameter is proposed to reduce the number of algorithmic iterations. Several simulation examples are presented to verify the effectiveness of the proposed algorithm.

I. Introduction

Autonomous swarm of UAVs have received increasing attention in recent years due to its wide range of applications in both military and civilian missions, e.g., wild fire monitoring [1], search and rescue [2], situational awareness [3], cooperative interception [4]. It is well-recognized that trajectory optimization plays an essential part in enhancing the swarm's autonomy and the main objective is to accomplish mission with optimality while ensuring safety of the swarm. Mathematically, this problem can be formulated as a Multi-Agent Optimal Control (MAOC) problem, i.e., finding

*PhD Student, School of Aerospace Engineering, 5th Zhongguancun South Street
†Professor, School of Aerospace Engineering, 5th Zhongguancun South Street
‡Assidant Engineer, No.1 Beijing Dahongmen South Road
§Professor, School of Aerospace Engineering, 5th Zhongguancun South Street, Member AIAA, Email: shaoming.he@bit.edu.cn

the optimal control history of each agent to minimize the global common performance metrics (e.g., control effort consumption, mission completion time) while complying with multiple nonlinear constraints (e.g., dynamic constraint, control constraint, obstacle avoidance, and inter-UAV collision avoidance).

Trajectory optimization for a single agent has been widely studied and different types of numerical methods are used to solve this nonlinear problem, such as pseudo-spectral methods [5, 6], sequential quadratic programming [7, 8], sequential convex programming [9, 10], and DDP [11]. The authors in [12] provided a well-organized description of the understanding of different numerical optimization algorithms from the perspective highlighting their advantages and mathematical nature. Due to its rapid convergence features and well-developed problem formulation framework, DDP has emerged as a popular solution for resolving nonlinear optimal control problems [13–17]. DDP is derived from classical Dynamic Programming (DP) that is based on Bellman's optimality principle and leverages second-order Taylor expansion to approximate the value function. Through this method, the original problem is decomposed into smaller dimensional subproblems related to the number of discrete points, and then the optimal variation of the control effort is determined using the first-order optimality condition to continually create new trajectories until convergence. With second-order Taylor approximation in the domain of nominal trajectories, DDP overcomes the curse of dimensionality of traditional Dynamic Programming (DP) methods [18, 19] and has faster second-order convergence rates [20].

Unlike the single agent scenario, solving the MAOC problem requires a suitable architecture to share the information among agents. Compared to the centralized architecture, the distributed architecture provides improved robustness against single agent failure and scalability with respect to the number of agents in the swarm [21]. For this reason, the distributed trajectory optimization shows great potentials in application to UAV swarms since the onboard computational resource is usually very limited [21]. Three typical types of distributed trajectory optimization methods to solve MAOC reported in the literature are distributed gradient descent [22], distributed sequential convex programming [23, 24], and ADMM [25, 26]. Among them, the ADMM has attracted significant attention due to its ability to generate distributed structures, solve problems involving numerous optimization variables, and handle equality constraints [27, 28]. A detailed review of ADMM is given by Boyd and shows how this method can be applied to large-scale distributed optimization problems [25]. ADMM decomposes the original large-scale optimization problem into a number of simpler sub-problems, each of which optimizes an independent objective function and corresponding decision variables. These sub-problems are solved by alternately updating the primal and dual variables to minimize the augmented Lagrangian function separately by their respective optimization solvers, and then the global solution of the original problem is obtained by gathering the solutions of the sub-problems. Notice that the original ADMM method requires a central node to collect the information from all nodes for the dual variable updating. This assumption was eliminated in [26] by introducing the auxiliary primal variables in place of the communication protocol constraints in the original ADMM, resulting in a fully distributed consensus alternating directional multiplier method (C-ADMM).

Unfortunately, the strongly non-convex nature of the UAV swarm trajectory optimization problem, introduced by

dynamic constraints, collision constraints, communication distance maintenance, makes it difficult to apply the ADMM directly to practical missions. One possible way is to combine the local single agent numerical solvers with ADMM to deal with the dynamic constraints, coordination constraints, and consensus constraints in MAOC. The authors in [29, 30] proposed a multi-agent DDP approach based on ADMM, but the dual variable update step of the algorithm is still computed centrally and hence their practical scalability is limited. One notable work in [31] proposed a framework in which the distributed computation based on ADMM algorithm and the local planning based on DDP is performed. The fully distributed nature of the framework shows superior scalability against sequential convex programming and quadratic programming methods. However, the algorithm needs to set the terminal time for each agent beforehand and two additional issues are till need to be considered: time optimization and convergence acceleration.

In some practical UAV swarm application, pre-setting the terminal time is intractable in nature for some missions and time optimization is a critical metric from the mission perspective. For example, the mission duration should be minimized in some missions, e.g., search and rescue, courier delivery [32, 33]. The authors in [34] developed a fast time-optimal trajectory planning method for UAVs using the ADMM algorithm, where the ADMM is utilized as a trajectory optimization algorithm rather than a distributed architectural approach for multi-agent systems, i.e., the communication process and distributed computing is not considered during the algorithm design. A swarm UAV trajectory planner that considers spatial-temporal optimization to shape the trajectory and modify the time allocation simultaneously was developed in [35] for autonomous flight in complicated jungle environments. However, this work relies more on individual sensors and onboard planners of each UAV, with less information sharing between the UAVs, therefore showing limited mission-specific robustness, and challenges adapting to sudden mission changes. By dynamically optimizing the total time of the trajectory as well as the control effort, we reviewed the original DDP in our earlier work [36, 37] and developed flexible final time DDP (FFT-DDP). A method, i.e., PDDP, to simultaneously optimize the parameters and control inputs is proposed in [38] with rigorous mathematical proof showing that the algorithm enjoys the property of fast and stable convergence to the minimum cost. By setting the flight time as an unknown parameter, PDDP can be utilized to solve the optimal flight time problem. However, the extension of these algorithms to the multi-agent framework is not straightforward and requires a substantial adjustment. Also, in the mission of simultaneous attack or sequential attack of a high-value target, the terminal time interval between two vehicles should be specified. This will introduces the terminal time sequence constraints and has not been resolved before.

Another critical issue in applying ADMM is that the convergence performance as well as the reliability largely depends on the selection of its penalty parameters. Residual balancing (RB) is a well-established and available adaptive method for solving general problems. This approach is based on the idea of balancing the primal and dual residuals to be approximately equal while improves the convergence by increasing or decreasing the penalty parameter with iterations [25]. However, the performance of the RB method varies greatly with the size of the problem. The authors in [39] utilized the Nesterov acceleration method to speed up the convergence of the algorithm, but the method does not

work well for general problems. The Barzilai-Borwein 'spectral' method was leveraged in [40, 41] to estimate the local curvature of the objective function to establish a penalty parameter adaptive approach with theoretical convergence proof. The extension to distributed optimization, however, needs further adjustment.

Motivated by the above observations, this paper proposes a distributed spatial-temporal optimization method based on PDDP and ADMM. The proposed algorithm is built on a two-level architecture basis, in which the PDDP layer is employed to solve local trajectory optimization problem with control limits, while ADMM layer is utilized to satisfy the local path constraints as well as the spatial-temporal coordination among all UAVs. The first major contribution of this work lies in the development of a method that simultaneously optimizes both the trajectories and flight times of a UAV swarm by resolving the issue setting the final time for agents beforehand. This dual optimization of spatial and temporal aspects in a single framework is a significant step forward in multi-agent swarm control. To the best of the authors' knowledge, the distributed free-time DDP is the first fully decentralized joint spatial-temporal optimization method and is capable of being applied to large-scale UAV swarms. The second key contribution is the introduction of a parameter adaptive approach designed to accelerate the convergence of the ADMM layer. This approach dynamically adjusts the penalty parameter during the optimization process, significantly reducing the number of iterations required, which in turn leads to computational time savings. A variety of numerical simulations with flight time optimization and time coordination constraints are leveraged to evaluate the effectiveness of the proposed algorithm. Up to the best of our knowledge, no similar works have been reported in the literature. The results reveal that the proposed algorithm is capable of handling different mission scenarios, e.g., free terminal time, simultaneous arrival and specified time intervals, and is capable of being applied to large-scale UAV swarms.

The remainder of the paper is organized as follows. Section II introduces the general form of the multi-agent optimal control problem and briefly reviews the PDDP and ADMM methods. The proposed D-PDDP algorithm is described in details in Section III, followed by the adaptive acceleration in Section IV. A variety of numerical evaluations of the algorithm's performance are presented in Section V and some conclusions are offered in Section VI.

II. Backgrounds and Preliminaries

In this section, we first introduce the general MAOC problem and then provide some preliminaries of the PDDP and ADMM for the completeness of the paper.

A. Multi-Agent Optimal Control Problem

For a typical MAOC problem, there is a group of M agents represented as a set $\mathcal{M} = \{1, ..., M\}$. Before describing the mathematical problems, we first define the concepts related to neighbors [31].

Definition 1 *(Neighbor set): The 'neighbor' set is a set that contains all neighbor-agents $j \in \mathcal{M}$ (including agent i) of agent $i \in \mathcal{M}$, which can be denoted as $\mathcal{N}_i \in \mathcal{M}$.*

Definition 2 *(Deemed Neighbor Set): A "deemed" neighbor set, represented as $\mathcal{P}_i = \{j \in \mathcal{M} : i \in \mathcal{N}_j\}$, is a collection of agents $j \in \mathcal{M}$ that regard agent $i \in \mathcal{M}$ as their neighbor.*

In a UAV swarm, we let $j \in \mathcal{N}_i$ if UAV j is within the effective communication range of i. Note that there is no requirement for i and j to be neighbors of each other, i.e., there is no requirement for $\mathcal{N}_i = \mathcal{P}_i$. Furthermore, we require that the agents communicate with each other only locally by the following assumptions.

Assumption 1 *Each agent $i \in \mathcal{M}$ is limited to exchanging information with agents $j \in \mathcal{M}$ that are both i's neighbors and regard i to be a neighbor; that is, i can only interact with $j \in \mathcal{N}_i \cup \mathcal{P}_i$.*

We consider that each agent i of the swarm is governed by a discrete-time nonlinear dynamics as

$$\boldsymbol{x}_{i,k+1} = \boldsymbol{f}_i\left(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}\right) \tag{1}$$

where $\boldsymbol{x}_{i,k} \in \mathbb{R}^{p_i}$ and $\boldsymbol{u}_{i,k} \in \mathbb{R}^{q_i}$ are the state and control vectors of the i-th agent, respectively. The subscript $k = \{0, 1, \ldots, N\}$ represent the time instant in the discrete framework, and the value of a variable at time instant t_k is indicated by the variable with subscript k. $\boldsymbol{f}_i$ represents discrete system dynamics with nonlinear properties.

The problem's global cost function, which is attempted to minimize by each agent, can be given by

$$J = \sum_{i=1}^{M} J_i\left(\boldsymbol{X}_i, \boldsymbol{U}_i\right) \tag{2}$$

in which $\boldsymbol{X}_i$ and $\boldsymbol{U}_i$ represent the stack of state vectors and control inputs from all time instants of agent i. Each agent has their own local objective function that contains terminal constraint and running cost as

$$J_i\left(\boldsymbol{X}_i, \boldsymbol{U}_i\right) = \sum_{k=0}^{N-1} l_i(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}) + \phi_i(\boldsymbol{x}_{i,N}, t_{i,N}) \tag{3}$$

in which the running cost is denoted by scalar function $l_i(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k})$ while the terminal cost is represented by $\phi_i(\boldsymbol{x}_{i,N}, t_{i,N})$.

Due to the physical constraints, e.g., actuation limitations, the control effort of each agent is commonly constrained, i.e.,

$$\boldsymbol{b}_{i,k}\left(\boldsymbol{u}_{i,k}\right) \leq 0 \tag{4}$$

where $\boldsymbol{b}_{i,k}\left(\boldsymbol{u}_{i,k}\right)$ denotes the control effort constraint function.

In addition to the control effort constraints, each vehicle's trajectory will also be limited by nonlinear path constraints, such as obstacle avoidance:

$$\boldsymbol{a}_{i,k}\left(\boldsymbol{x}_{i,k}\right) \leq 0 \tag{5}$$

where $\boldsymbol{a}_{i,k}$ is a nonlinear function for describing path constraints.

In particular, in a spatial-temporal optimization problem, it is free for each vehicle w.r.t. its final time. Also, the terminal time should be within the boundaries of the vehicle's achievable range. Thus, we formulate the terminal time constraint as

$$c_{i,N}\left(t_{i,N}\right) \leq 0 \tag{6}$$

where $c_{i,N}\left(t_{i,N}\right)$ is a general nonlinear function.

With the definition of neighbors, the inter-agent state constraint between i and its neighbors can be expressed as

$$\boldsymbol{d}_{ij,k}\left(\boldsymbol{x}_{i,k}, \boldsymbol{x}_{j,k}\right) \leq 0 \tag{7}$$

and the inter-agent terminal time constraint between i and its neighbors can be formulated as:

$$\boldsymbol{e}_{ij,N}\left(t_{i,N}, t_{j,N}\right) = 0 \tag{8}$$

in which the above two inter-agent constraints do not contain agent i itself, i.e. $j \neq i$. And the expression of $\boldsymbol{d}_{ij,k}\left(\boldsymbol{x}_{i,k}, \boldsymbol{x}_{j,k}\right)$ represents the constraint of collision avoidance between UAV i and j, or the constraint that i and j maintain their connection within the communication distance. And the function $\boldsymbol{e}_{ij,N}\left(t_{i,N}, t_{j,N}\right)$ stands for the constraint that the all agents have the same final time or complete a mission in a certain time sequence.

In summary, the multi-agent optimal control problem, denoted as $MAOC_1$, considered in this paper has the following form.

$MAOC_1$: Find

$$\begin{aligned}
&\{\boldsymbol{U}_i^*, t_{i,N}^*\} = \min \sum_{i=1}^{M} J_i\left(\boldsymbol{X}_i, \boldsymbol{U}_i\right) \\
s.t. \quad & \boldsymbol{x}_{i,k+1} = \boldsymbol{f}_i\left(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}\right) \\
& \boldsymbol{a}_{i,k}\left(\boldsymbol{x}_{i,k}\right) \leq 0 \\
& \boldsymbol{b}_{i,k}\left(\boldsymbol{u}_{i,k}\right) \leq 0 \\
& c_{i,N}\left(t_{i,N}\right) \leq 0 \\
& \boldsymbol{d}_{ij,k}\left(\boldsymbol{x}_{i,k}, \boldsymbol{x}_{j,k}\right) \leq 0, \quad j \neq i \\
& \boldsymbol{e}_{ij,N}\left(t_{i,N}, t_{j,N}\right) = 0, \quad j \neq i
\end{aligned} \tag{9}$$

Problem $MAOC_1$ is assumed to have a solution. Because of the inherent nonlinearity, analytical problem solving is usually unfeasible. As a result, numerical procedures are typically employed to tackle this problem.

B. Parameterized Differential Dynamic Programming

The PDDP algorithm is a variant of DDP to optimize the control sequence and unknown parameter simultaneously. By abstracting the terminal time as an unknown parameter, PDDP is able to solve the problem of optimizing spatial-temporal trajectory without pre-setting terminal time for a single agent. For the completeness, we briefly review the key points of PDDP in this subsection and ignore the agent index for simplicity. PDDP is designed to solve the optimization problem as

$$\begin{aligned} \min_{\boldsymbol{U};\boldsymbol{\theta}} & J(\boldsymbol{U};\boldsymbol{\theta}) = \sum_{k=1}^{N-1} l\left(\boldsymbol{x}_k, \boldsymbol{u}_k; \boldsymbol{\theta}\right) + \phi\left(\boldsymbol{x}_N; \boldsymbol{\theta}\right) \\ s.t. \quad & \boldsymbol{x}_{k+1} = \boldsymbol{f}\left(\boldsymbol{x}_k, \boldsymbol{u}_k; \boldsymbol{\theta}\right) \end{aligned} \tag{10}$$

where the semicolon separating the parameter $\boldsymbol{\theta}$ from the state $\boldsymbol{x}$ and the control $\boldsymbol{u}$ is to show that $\boldsymbol{\theta}$ is a time-invariant parameter, independent of the time instant k.

The basic idea of the original PDDP is to transform the problem of optimizing the entire trajectory into the backward sequential optimization problem for each time instant using Bellman's optimality principle. The parameterized optimal cost-to-go, which can be also defined as the parameterized value function, is provided as

$$V\left(\boldsymbol{x}_k; \boldsymbol{\theta}\right) = \min[\underbrace{l\left(\boldsymbol{x}_k, \boldsymbol{u}_k; \boldsymbol{\theta}\right) + V\left(\boldsymbol{x}_{k+1}; \boldsymbol{\theta}\right)}_{Q(\boldsymbol{x}_k, \boldsymbol{u}_k; \boldsymbol{\theta})}] \tag{11}$$

and the action-value function is specified as $Q\left(\boldsymbol{x}_k, \boldsymbol{u}_k; \boldsymbol{\theta}\right)$.

Unlike the standard DDP that only considers quadratic approximations with respect to the state $\boldsymbol{x}_k$ and the control $\boldsymbol{u}_k$ along a nominal trajectory $\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right)$ at time t_k, the parameterized DDP also needs to consider quadratic approximations of the value function with respect to parameter $\boldsymbol{\theta}$, which gives

$$\begin{aligned} V\left(\boldsymbol{x}_k + \delta\boldsymbol{x}_k; \boldsymbol{\theta} + \delta\boldsymbol{\theta}\right) \approx & V\left(\boldsymbol{x}_k; \boldsymbol{\theta}\right) + V_{\boldsymbol{x},k}^{\mathrm{T}} \delta\boldsymbol{x}_k + V_{\theta,k}^{\mathrm{T}} \delta\boldsymbol{\theta} \\ & + \frac{1}{2} \begin{bmatrix} \delta\boldsymbol{x}_k \\ \delta\boldsymbol{\theta} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} V_{\boldsymbol{xx},k} & V_{\boldsymbol{x\theta},k} \\ V_{\boldsymbol{\theta x},k} & V_{\boldsymbol{\theta\theta},k} \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{x}_k \\ \delta\boldsymbol{\theta} \end{bmatrix} \end{aligned} \tag{12}$$

where the state, control and parameter deviations at t_k from the nominal trajectory is denoted as $\delta\boldsymbol{x}_k$, $\delta\boldsymbol{u}_k$ and $\delta\boldsymbol{\theta}$. $V_{\boldsymbol{x},k}$, $V_{\theta,k}$ are the gradient vectors and $V_{\boldsymbol{xx},k}$, $V_{\boldsymbol{x\theta},k}$, $V_{\boldsymbol{\theta x},k}$, $V_{\boldsymbol{\theta\theta},k}$ are the Hessian at time instant k of the value function $V\left(\boldsymbol{x}_k; \boldsymbol{\theta}\right)$.

Likewise, $Q\left(\boldsymbol{x}_k, \boldsymbol{u}_k; \boldsymbol{\theta}\right)$ defined in Eq. (11) has a quadratic approximation that is provided by

$$\begin{aligned} Q(\boldsymbol{x}_k + \delta\boldsymbol{x}_k, \boldsymbol{u}_k + \delta\boldsymbol{u}_k; \boldsymbol{\theta} + \delta\boldsymbol{\theta}) \approx & Q(\boldsymbol{x}_k, \boldsymbol{u}_k; \boldsymbol{\theta}) + Q_{\boldsymbol{x},k}^{\mathrm{T}}\delta\boldsymbol{x}_k + Q_{\boldsymbol{u},k}^{\mathrm{T}}\delta\boldsymbol{u}_k + Q_{\boldsymbol{\theta},k}^{\mathrm{T}}\delta\boldsymbol{\theta} \\ & + \frac{1}{2}\begin{bmatrix} \delta\boldsymbol{x}_k \\ \delta\boldsymbol{u}_k \\ \delta\boldsymbol{\theta} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} Q_{\boldsymbol{xx},k} & Q_{\boldsymbol{xu},k} & Q_{\boldsymbol{x\theta},k} \\ Q_{\boldsymbol{ux},k} & Q_{\boldsymbol{uu},k} & Q_{\boldsymbol{u\theta},k} \\ Q_{\boldsymbol{\theta x},k} & Q_{\boldsymbol{\theta u},k} & Q_{\boldsymbol{\theta\theta},k} \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{x}_k \\ \delta\boldsymbol{u}_k \\ \delta\boldsymbol{\theta} \end{bmatrix} \end{aligned} \tag{13}$$

Organizing the same order terms of $\delta\boldsymbol{x}$, $\delta\boldsymbol{u}$ and $\delta\boldsymbol{\theta}$ will provide the derivatives of $Q\left(\boldsymbol{x}_k, \boldsymbol{u}_k; \boldsymbol{\theta}\right)$ as

$$\begin{aligned} Q_{\boldsymbol{x},k} &= l_{\boldsymbol{x},k} + \boldsymbol{f}_{\boldsymbol{x},k}^{\mathrm{T}} V_{\boldsymbol{x},k+1} \\ Q_{\boldsymbol{u},k} &= l_{\boldsymbol{u},k} + \boldsymbol{f}_{\boldsymbol{u},k}^{\mathrm{T}} V_{\boldsymbol{x},k+1} \\ Q_{\boldsymbol{\theta},k} &= l_{\boldsymbol{\theta},k} + V_{\boldsymbol{\theta},k+1} + \boldsymbol{f}_{\boldsymbol{\theta},k}^{\mathrm{T}} V_{\boldsymbol{x},k+1} \\ Q_{\boldsymbol{xx},k} &= l_{\boldsymbol{xx},k} + \boldsymbol{f}_{\boldsymbol{x},k}^{\mathrm{T}} V_{\boldsymbol{xx},k+1} \boldsymbol{f}_{\boldsymbol{x},k} \\ Q_{\boldsymbol{uu},k} &= l_{\boldsymbol{uu},k} + \boldsymbol{f}_{\boldsymbol{u},k}^{\mathrm{T}} V_{\boldsymbol{xx},k+1} \boldsymbol{f}_{\boldsymbol{u},k} \\ Q_{\boldsymbol{\theta\theta},k} &= l_{\boldsymbol{\theta\theta},k} + V_{\boldsymbol{\theta\theta},k+1} + \boldsymbol{f}_{\boldsymbol{\theta},k}^{\mathrm{T}} V_{\boldsymbol{xx},k+1} \boldsymbol{f}_{\boldsymbol{\theta},k} + \boldsymbol{f}_{\boldsymbol{\theta},k}^{\mathrm{T}} V_{\boldsymbol{x\theta},k+1} + V_{\boldsymbol{\theta x},k+1} \boldsymbol{f}_{\boldsymbol{\theta},k} \\ Q_{\boldsymbol{xu},k} &= l_{\boldsymbol{xu},k} + \boldsymbol{f}_{\boldsymbol{x},k}^{\mathrm{T}} V_{\boldsymbol{xx},k+1} \boldsymbol{f}_{\boldsymbol{u},k} = Q_{\boldsymbol{ux},k}^{\mathrm{T}} \\ Q_{\boldsymbol{x\theta},k} &= l_{\boldsymbol{x\theta},k} + \boldsymbol{f}_{\boldsymbol{x},k}^{\mathrm{T}} V_{\boldsymbol{xx},k+1} \boldsymbol{f}_{\boldsymbol{\theta},k} + \boldsymbol{f}_{\boldsymbol{x},k}^{\mathrm{T}} V_{\boldsymbol{x\theta},k+1} = Q_{\boldsymbol{\theta x},k}^{\mathrm{T}} \\ Q_{\boldsymbol{u\theta},k} &= l_{\boldsymbol{u\theta},k} + \boldsymbol{f}_{\boldsymbol{u},k}^{\mathrm{T}} V_{\boldsymbol{xx},k+1} \boldsymbol{f}_{\boldsymbol{\theta},k} + \boldsymbol{f}_{\boldsymbol{u},k}^{\mathrm{T}} V_{\boldsymbol{u\theta},k+1} = Q_{\boldsymbol{\theta u},k}^{\mathrm{T}} \end{aligned} \tag{14}$$

The second order expansion of $Q(\boldsymbol{x}_k, \boldsymbol{u}_k; \boldsymbol{\theta})$ is a quadratic function w.r.t. control deviation $\delta\boldsymbol{u}_k$, then necessary optimality condition can be utilized for deriving $\delta\boldsymbol{u}_k^*$ as

$$\delta\boldsymbol{u}_k^* = \boldsymbol{k}_k + \boldsymbol{K}_k\delta\boldsymbol{x}_k + \boldsymbol{M}_k\delta\boldsymbol{\theta}_k \tag{15}$$

where

$$\begin{aligned} \boldsymbol{k}_k &= -Q_{\boldsymbol{uu},k}^{-1} Q_{\boldsymbol{u},k} \\ \boldsymbol{K}_k &= -Q_{\boldsymbol{uu},k}^{-1} Q_{\boldsymbol{ux},k} \\ \boldsymbol{M}_k &= -Q_{\boldsymbol{uu},k}^{-1} Q_{\boldsymbol{u\theta},k} \end{aligned} \tag{16}$$

where the feedforward term $\boldsymbol{k}_k$ and the feedback term $\boldsymbol{K}_k$ are the same as in the original DDP method, with the addition of an extra feedback term $\boldsymbol{M}_k$ for the optimization of $\boldsymbol{\theta}$ in PDDP.

Since parameter $\boldsymbol{\theta}$ is time-independent, i.e., it does not vary with time, it only needs to be updated at the time instant

$k = 1$. In other words, the expression for optimal parameter increment $\delta\boldsymbol{\theta}^*$ can be obtained with the condition that the deviation of $\delta\boldsymbol{x}_1 = 0$ for Eq. (12) as follows

$$\delta\boldsymbol{\theta}^* = -V_{\boldsymbol{\theta\theta},1}^{-1} V_{\boldsymbol{\theta},1} \tag{17}$$

To ensure the convergence of the PDDP algorithm, the feedforward terms $\boldsymbol{k}_k$ of $\delta\boldsymbol{u}_k^*$ and $\boldsymbol{M}_k$ of $\delta\boldsymbol{\theta}^*$ need to be scaled by the damping coefficients α_l, where the α_l can be determined by the line search method [38], and there is

$$\delta\boldsymbol{u}_k^* = \alpha_l \boldsymbol{k}_k + \boldsymbol{K}_k \delta\boldsymbol{x}_k + \boldsymbol{M}_k \delta\boldsymbol{\theta}_k \tag{18}$$

$$\delta\boldsymbol{\theta}^* = -\alpha_l V_{\boldsymbol{\theta\theta},1}^{-1} V_{\boldsymbol{\theta},1} \tag{19}$$

Substituting optimal control variation (15) into the Taylor expansion of $Q(\boldsymbol{x}_k, \boldsymbol{u}_k; \boldsymbol{\theta})$ and grouping the same order terms of $\delta\boldsymbol{x}_k$ and $\delta\boldsymbol{\theta}$ yields

$$\begin{aligned}
V\left(\boldsymbol{x}_k; \boldsymbol{\theta}\right) &= Q\left(\boldsymbol{x}_k, \boldsymbol{u}_k; \boldsymbol{\theta}\right) - \left(\frac{1}{2}\alpha_l^2 - \alpha_l\right) Q_{\boldsymbol{u},k}^{\mathrm{T}} Q_{\boldsymbol{uu},k}^{-1} Q_{\boldsymbol{u},k} \\
V_{\boldsymbol{x},k} &= Q_{\boldsymbol{x},k} - Q_{\boldsymbol{xu},k} Q_{\boldsymbol{uu},k}^{-1} Q_{\boldsymbol{u},k} \\
V_{\boldsymbol{\theta},k} &= Q_{\boldsymbol{\theta},k} - Q_{\boldsymbol{\theta u},k} Q_{\boldsymbol{uu},k}^{-1} Q_{\boldsymbol{u},k} \\
V_{\boldsymbol{xx},k} &= Q_{\boldsymbol{xx},k} - Q_{\boldsymbol{xu},k} Q_{\boldsymbol{uu},k}^{-1} Q_{\boldsymbol{ux},k} \\
V_{\boldsymbol{x\theta},k} &= Q_{\boldsymbol{x\theta},k} - Q_{\boldsymbol{xu},k} Q_{\boldsymbol{uu},k}^{-1} Q_{\boldsymbol{u\theta},k} = V_{\boldsymbol{\theta x},k}^{\mathrm{T}} \\
V_{\boldsymbol{\theta\theta},k} &= Q_{\boldsymbol{\theta\theta},k} - Q_{\boldsymbol{\theta u},k} Q_{\boldsymbol{uu},k}^{-1} Q_{\boldsymbol{u\theta},k}
\end{aligned} \tag{20}$$

Before running the PDDP algorithm, control effort initial guess should be given for generating the nominal trajectory. Afterwards, the terminal value function is initialized using the terminal objective function as $V\left(\boldsymbol{x}_N; \boldsymbol{\theta}\right) = \phi(\boldsymbol{x}_N; \boldsymbol{\theta})$, $V_{\boldsymbol{x},N} = \phi_{\boldsymbol{x},N}$, $V_{\boldsymbol{xx},N} = \phi_{\boldsymbol{xx},N}$, $V_{\boldsymbol{\theta},N} = \phi_{\boldsymbol{\theta},N}$, $V_{\boldsymbol{\theta\theta},N} = \phi_{\boldsymbol{\theta\theta},N}$, $V_{\boldsymbol{x\theta},N} = \phi_{\boldsymbol{x\theta},N} = V_{\boldsymbol{\theta x},N}^{\mathrm{T}}$. Then the optimal control correction $\delta\boldsymbol{u}_k$ is computed using Eq. (15) in backward order while the optimal parameter change $\delta\boldsymbol{\theta}$ is computed using Eq. (17) at time instant $k = 1$. In brief, the one-step control update and one-iteration parameter update for generating a new trajectory are given by

$$\begin{aligned}
\boldsymbol{u}_k &= \boldsymbol{u}_k + \delta\boldsymbol{u}_k^* = \boldsymbol{u}_k + \alpha_l \boldsymbol{k}_k + \boldsymbol{K}_k \delta\boldsymbol{x}_k + \boldsymbol{M}_k \delta\boldsymbol{\theta}_k \\
\boldsymbol{\theta} &= \boldsymbol{\theta} + \delta\boldsymbol{\theta}^* = \boldsymbol{\theta} - \alpha_l V_{\boldsymbol{\theta\theta},1}^{-1} V_{\boldsymbol{\theta},1}
\end{aligned} \tag{21}$$

The forward process is initiated to produce another nominal trajectory once the backward process has computed the optimal control updates and optimal parameter updates. The PDDP method performs the backward and forward

processes iteratively until the cost function's error between two iterations is smaller than a preset threshold, that is,

$$\left| J^{(n)} - J^{(n-1)} \right| < \epsilon \tag{22}$$

in which the constant $\epsilon > 0$ has a comparatively low value and $J^{(n)}$ indicates the n-th iteration's objective function. Once the termination condition is satisfied, the iteration process of the algorithm is terminated and the optimal state and controls are given. The pseudo-code of PDDP is presented in Algorithm 1 as follows:

Algorithm 1 PDDP algorithm workflow

Initialize $\boldsymbol{U}_0 = \{\boldsymbol{u}_0, \ldots, \boldsymbol{u}_k, \ldots, \boldsymbol{u}_{N-1}\}, \boldsymbol{\theta}_0$
for $k = 0, ..., N-1$ **do**
  $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k; \boldsymbol{\theta})$
end for
$J_0 \leftarrow$ Eq.(10)
while $|J_i - J_{i-1}| \geq \epsilon$ **do**
  $V(\boldsymbol{x}_N; \boldsymbol{\theta}) = \phi(\boldsymbol{x}_N; \boldsymbol{\theta})$, $V_{\boldsymbol{x},N} = \phi_{\boldsymbol{x},N}$, $V_{\boldsymbol{xx},N} = \phi_{\boldsymbol{xx},N}$
  $V_{\boldsymbol{\theta},N} = \phi_{\boldsymbol{\theta},N}$, $V_{\boldsymbol{\theta\theta},N} = \phi_{\boldsymbol{\theta\theta},N}$, $V_{\boldsymbol{x\theta},N} = \phi_{\boldsymbol{x\theta},N} = V^{\mathrm{T}}_{\boldsymbol{\theta x},N}$
  for $k = N-1, ..., 0$ **do**
    Using Eq.(14) to calculate the Jacobian and Hessian of $Q(\boldsymbol{x}_k, \boldsymbol{u}_k; \boldsymbol{\theta})$
    $\boldsymbol{k}_k, \boldsymbol{K}_k, \boldsymbol{M}_k \leftarrow$ Eq.(16)
    Using Eq.(20) to calculate the derivatives of $V(\boldsymbol{x}_k, \boldsymbol{u}_k; \boldsymbol{\theta})$
    if $k = 1$ **then**
      $\delta\boldsymbol{\theta}^* = -V^{-1}_{\boldsymbol{\theta\theta}}(V_{\boldsymbol{\theta}} + V^{\mathrm{T}}_{\boldsymbol{\theta x}}\delta\boldsymbol{x})$
    end if
  end for
  $\boldsymbol{\theta} \leftarrow$ Eq.(21)
  for $k = 0, ..., N-1$ **do**
    $\delta\boldsymbol{x}_k = \boldsymbol{x}^{\mathrm{new}}_k - \boldsymbol{x}_k$
    $\boldsymbol{u}_k = \min(\max(\boldsymbol{u}_k + \boldsymbol{k}_k + \boldsymbol{K}_k\delta\boldsymbol{x}_k + \boldsymbol{M}_k\delta\boldsymbol{\theta}, \boldsymbol{u}_{\min}), \boldsymbol{u}_{\max})$
    $\boldsymbol{x}^{\mathrm{new}}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k; \boldsymbol{\theta})$
  end for
  $\boldsymbol{X} = \boldsymbol{X}^{\mathrm{new}}$
end while

Remark 1 *Consider the terminal time t_N as an unknown parameter $\boldsymbol{\theta}$, the PDDP is able to find the optimal terminal time and the control sequence. However, it should be noted the forward pass of PDDP requires to integrate the system dynamics from the initial time instant until t_N. For reason, we artificially change the argument from t to a $\iota_k = \frac{t_k - t_{i,1}}{t_N} \in [0, 1]$. With this modification, the optimization problem can then be reformulated as*

$$\begin{aligned} \min_{\boldsymbol{U}; t_N} J(\boldsymbol{U}; t_N) &= \sum_{k=1}^{N-1} t_{i,N}\, l\left(\boldsymbol{x}_k(\iota_k), \boldsymbol{u}_k(\iota_k)\right) + \phi\left(\boldsymbol{x}_N(\iota_N)\right) \\ s.t. \quad \boldsymbol{x}_{k+1} &= t_N \boldsymbol{f}\left(\boldsymbol{x}_k(\iota_k), \boldsymbol{u}_k(\iota_k)\right) \end{aligned} \tag{23}$$

C. Alternating Direction Method of Multiplier

This subsection provides a brief description of the ADMM algorithm [25]. ADMM has integrated the decomposability of the dual ascent method and the excellent convergence properties of multiplier method. The basic idea of ADMM is to decompose the optimization of the objective function into a number of sub-problems optimized on different variables or substructures, and then through a coordination mechanism (through the introduction of Lagrangian multipliers and dual variables) to ensure that the solutions of each subproblem converge to the global optimum. The resulting method allows parallel optimization under more general conditions and the standard form is as

$$\begin{aligned} &\min_{\boldsymbol{\varphi},\boldsymbol{\psi}} H(\boldsymbol{\varphi}) + G(\boldsymbol{\psi}) \\ &\text{s.t. } \boldsymbol{\alpha}\boldsymbol{\varphi} + \boldsymbol{\beta}\boldsymbol{\psi} = \boldsymbol{\delta} \end{aligned} \tag{24}$$

In the standard ADMM form, the primal variables are $\boldsymbol{\varphi} \in \mathbb{R}^n$, $\boldsymbol{\psi} \in \mathbb{R}^m$, and the functions of $\boldsymbol{\varphi}$ and $\boldsymbol{\psi}$ are defined as $H : \mathbb{R}^n \to \mathbb{R}, G : \mathbb{R}^m \to \mathbb{R}$, and the coefficient matrix of $\boldsymbol{\varphi}$ and $\boldsymbol{\psi}$ in the constraint is $A \in \mathbb{R}^{r\times n}$, $B \in \mathbb{R}^{r\times m}$, and $\boldsymbol{\delta} \in \mathbb{R}^r$ is a constant vector.

The augmented Lagrange multipliers formation of problem (24) can be provided as

$$\begin{aligned} \mathcal{L}(\boldsymbol{\varphi}, \boldsymbol{\psi}, \boldsymbol{\lambda}) =& H(\boldsymbol{\varphi}) + G(\boldsymbol{\psi}) + \boldsymbol{\lambda}^{\mathrm{T}}(\boldsymbol{\alpha}\boldsymbol{\varphi} + \boldsymbol{\beta}\boldsymbol{\psi} - \boldsymbol{\delta}) \\ &+ \frac{\vartheta}{2}\|\boldsymbol{\alpha}\boldsymbol{\varphi} + \boldsymbol{\beta}\boldsymbol{\psi} - \boldsymbol{\delta}\|_2^2 \end{aligned} \tag{25}$$

in which the dual variable of the constraint is $\boldsymbol{\lambda} \in \mathbb{R}^r$ and the penalty parameter of it is $\vartheta > 0$.

The update step for iterative optimization of the algorithm is given by:

$$\begin{aligned} \boldsymbol{\varphi}^{n+1} &= \arg\min_{\boldsymbol{\varphi}} \mathcal{L}\left(\boldsymbol{\varphi}, \boldsymbol{\psi}^n, \boldsymbol{\lambda}^n\right) \\ \boldsymbol{\psi}^{n+1} &= \arg\min_{\boldsymbol{\psi}} \mathcal{L}\left(\boldsymbol{\varphi}^{n+1}, \boldsymbol{\psi}, \boldsymbol{\lambda}^n\right) \\ \boldsymbol{\lambda}^{n+1} &= \boldsymbol{\lambda}^n + \vartheta(\boldsymbol{\alpha}\boldsymbol{\varphi}^{n+1} + \boldsymbol{\beta}\boldsymbol{\psi}^{n+1} - \boldsymbol{\delta}) \end{aligned} \tag{26}$$

where n represents the number of iterations of the algorithm. The standard form of ADMM described above contains only two variables $\boldsymbol{\varphi}$ and $\boldsymbol{\psi}$, but a multiblock form containing more than three variables can be developed via the ADMM method [42].

III. Distributed Parameterized Differential Dynamic Programming

This section provides a comprehensive explanation of D-PDDP algorithm. We first reformulate the problem to make it suitable in using distributed optimization and then derive the algorithm using the concept of ADMM. In the framework of free-final-time spatial-temporal trajectory optimization, the final time $t_{i,N}$ of each agent is chosen as

the unknown parameter $\boldsymbol{\theta}$ in the PDDP. Therefore, $t_{i,N}$ is used as the symbolic representation of the spatial-temporal trajectory optimization in the following derivation and as the symbolic representation of unknown parameter in the PDDP algorithm in this section.

A. Problem Reformulation

The problem (9) is a general form of the multi-agent system problem, which is necessary to be transformed to optimize it using the distributed optimization method [25]. Since the local spatial-temporal constraints need to be solved by the ADMM method in the two-layer framework, the 'safe' copy variable $\tilde{\boldsymbol{x}}_{i,k} \in \mathbb{R}^{p_i}, \tilde{\boldsymbol{u}}_{i,k} \in \mathbb{R}^{q_i}, \tilde{t}_{i,N} \in \mathbb{R}$ of $\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}$ and $t_{i,N}$ are introduced first, which should consensus with the actual trajectory computed by the PDDP algorithm, as

$$\tilde{\boldsymbol{x}}_{i,k} = \boldsymbol{x}_{i,k}, \tilde{\boldsymbol{u}}_{i,k} = \boldsymbol{u}_{i,k}, \tilde{t}_{i,N} = t_{i,N} \tag{27}$$

and they should also satisfy the local constraints of a single agent, i.e., $\boldsymbol{a}_{i,k}(\tilde{x}_{i,k}) \leq 0$, $\boldsymbol{b}_{i,k}\left(\tilde{\boldsymbol{u}}_{i,k}\right) \leq 0$ and $c_{i,N}\left(\tilde{t}_{i,N}\right) \leq 0$.

Since the inter-agent constraints (7) and (8) introduce the inter-couplings among the agents, indicating that problem (9) is difficult to solve in a decentralized manner. For this reason, each agent within the proposed architecture is required to keep a 'safe' copy of its variables and those of its neighbors to achieve consensus. Specifically, the state variable and final time of j that under the viewpoint of i is defined as: $\tilde{\boldsymbol{x}}^i_{j,k} \in \mathbb{R}^{p_j}, \tilde{t}^i_{j,N} \in \mathbb{R}$, $j \in \mathcal{N}_i, i \in \mathcal{M}$, i.e., $\tilde{\boldsymbol{x}}^i_{j,k}$ is a variable that is relatively safe and $\tilde{t}^i_{j,N}$ is a proper final time of agent j from agent i's perspective. Thus, augmented state and time variable, which are stacked from all neighbors of agent i, can be defined as

$$\tilde{\boldsymbol{x}}^{\mathrm{a}}_{i,k} = \left[\left\{\tilde{\boldsymbol{x}}^i_{j,k}\right\}_{j\in\mathcal{N}_i}\right] \in \mathbb{R}^{\tilde{p}_i}, \tilde{p}_i = \sum_{j\in\mathcal{N}_i} p_j \tag{28}$$

$$\tilde{\boldsymbol{t}}^{\mathrm{a}}_{i,N} = \left[\left\{\tilde{t}^i_{j,N}\right\}_{j\in\mathcal{N}_i}\right] \in \mathbb{R}^{|\mathcal{N}_i|} \tag{29}$$

Constraints (7) and (8) should be reviewed from the viewpoint of every agent i with the incorporation of copy variables. The inter-agent state constant can be recast as $\boldsymbol{d}_{ij,k}(\tilde{\boldsymbol{x}}_{i,k}, \tilde{\boldsymbol{x}}^i_{j,k}) \leq 0$ and the inter-agent time constraint also could be rewritten as $\boldsymbol{e}_{ij,N}(\tilde{t}_{i,N}, \tilde{t}^i_{j,N}) \leq 0$, or represented using a compact form with

$$\begin{aligned} \boldsymbol{d}^{\mathrm{a}}_{i,k}\left(\tilde{\boldsymbol{x}}^{\mathrm{a}}_{i,k}\right) &\leq 0 \\ \boldsymbol{e}^{\mathrm{a}}_{i,N}\left(\tilde{\boldsymbol{t}}^{\mathrm{a}}_{i,N}\right) &\leq 0 \end{aligned} \tag{30}$$

in which $\boldsymbol{d}^{\mathrm{a}}_{i,k}\left(\tilde{\boldsymbol{x}}^{\mathrm{a}}_{i,k}\right) = \left[\left\{\boldsymbol{d}_{ij,k}\left(\tilde{\boldsymbol{x}}_{i,k}, \tilde{\boldsymbol{x}}^i_{j,k}\right)\right\}_{j\in\mathcal{N}_i\setminus\{i\}}\right]$, $\boldsymbol{e}^{\mathrm{a}}_{i,N}\left(\tilde{t}^{\mathrm{a}}_{i,k}\right) = \left[\left\{\boldsymbol{e}_{ij,N}\left(\tilde{t}_{i,N}, \tilde{t}^i_{j,N}\right)\right\}_{j\in\mathcal{N}_i\setminus\{i\}}\right]$.

Furthermore, since multiple neighbors of agent i may hold different perspectives on i's variables, a consensus between these perspectives needs to be achieved to avoid conflict. Thus, the state and time variables for consensus are

defined as $z_{i,k} \in \mathbb{R}^p$ and $s_{i,N} \in \mathbb{R}$, where $i \in \mathcal{M}$. Then, the consensus constraints can be enforced as

$$\tilde{\boldsymbol{x}}^i_{j,k} = \boldsymbol{z}_{j,k}, \quad t^i_{j,N} = s_{j,N}, \quad j \in \mathcal{N}_i \tag{31}$$

and the compact form considering all the neighbors of Eq. (31) is given by

$$\tilde{\boldsymbol{x}}^{\mathrm{a}}_{i,k} = \boldsymbol{z}^{\mathrm{a}}_{i,k}, \quad \tilde{\boldsymbol{t}}^{\mathrm{a}}_{i,N} = \boldsymbol{s}^{\mathrm{a}}_{i,N} \tag{32}$$

where $\boldsymbol{z}^{\mathrm{a}}_{i,k} = \left[\left\{\boldsymbol{z}_{j,k}\right\}_{j\in\mathcal{N}_i}\right] \in \mathbb{R}^{\tilde{p}_i}$, $\boldsymbol{s}^{\mathrm{a}}_{i,N} = \left[\left\{\boldsymbol{s}_{j,N}\right\}_{j\in\mathcal{N}_i}\right] \in \mathbb{R}^{M}$.

In summary, we can transform the original $MAOC_1$ problem into the following equivalent optimization problem.

$MAOC_2$: Find

$$\begin{aligned}
\{\boldsymbol{U}^*_i, t^*_{i,N}\} = \min & \sum_{i=1}^{M} J_i\left(\boldsymbol{X}_i, \boldsymbol{U}_i\right) \\
s.t. \quad & \boldsymbol{x}_{i,k+1} = \boldsymbol{f}_i\left(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}\right) \\
& \boldsymbol{a}_{i,k}\left(\tilde{\boldsymbol{x}}_{i,k}\right) \le 0 \\
& \boldsymbol{b}_{i,k}\left(\tilde{\boldsymbol{u}}_{i,k}\right) \le 0 \\
& c_{i,N}\left(\tilde{t}_{i,N}\right) \le 0 \\
& \boldsymbol{d}^{\mathrm{a}}_{i,k}\left(\tilde{\boldsymbol{x}}^{\mathrm{a}}_{i,k}\right) \le 0 \\
& \boldsymbol{e}^{\mathrm{a}}_{i,N}\left(\tilde{\boldsymbol{t}}^{\mathrm{a}}_{i,N}\right) = 0 \\
\boldsymbol{u}_{i,k} = \tilde{\boldsymbol{u}}_{i,k},\ & \boldsymbol{x}_{i,k} = \tilde{\boldsymbol{x}}_{i,k},\ t_{i,N} = \tilde{t}_{i,N},\ \tilde{\boldsymbol{x}}^{\mathrm{a}}_{i,k} = \boldsymbol{z}^{\mathrm{a}}_{i,k},\ \tilde{\boldsymbol{t}}^{\mathrm{a}}_{i,N} = \boldsymbol{s}^{\mathrm{a}}_{i,N}
\end{aligned} \tag{33}$$

The algorithm proposed in this subsection to solve $MAOC_2$ has a two-layer structure, where 'inter-agent' spatial-temporal constraints are handled by ADMM, while flexible-final-time trajectory optimization with dynamics and control limits for a single agent is solved using PDDP. The general idea of the proposed algorithm can be expressed in the Figure 1.

B. Problem Solution

To Problem (33) using ADMM, we first construct the standard ADMM form by fusing the constraints into the objective function using the corresponding indicator function

$$\begin{aligned}
\min & \sum_{i=1}^{M}\sum_{k=0}^{N} J_i\left(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}\right) + \mathcal{I}_{\boldsymbol{f}_i}\left(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}\right) + \mathcal{I}_{\boldsymbol{b}_{i,k}}\left(\tilde{\boldsymbol{u}}_{i,k}\right) + \mathcal{I}_{c_i}\left(\tilde{t}_{i,N}\right) \\
& + \mathcal{I}_{\boldsymbol{a}_{i,k}}\left(\tilde{\boldsymbol{x}}_{i,k}, \tilde{\boldsymbol{u}}_{i,k}\right) + \mathcal{I}_{\boldsymbol{d}^{\mathrm{a}}_{i,k}}\left(\tilde{\boldsymbol{x}}^{\mathrm{a}}_{i,k}\right) + \mathcal{I}_{\boldsymbol{e}^{\mathrm{a}}_{i,N}}\left(\tilde{\boldsymbol{t}}^{\mathrm{a}}_{i,N}\right) \\
\text{s.t. } & \boldsymbol{u}_{i,k} = \tilde{\boldsymbol{u}}_{i,k},\ \boldsymbol{x}_{i,k} = \tilde{\boldsymbol{x}}_{i,k},\ t_{i,N} = \tilde{t}_{i,N},\ \tilde{\boldsymbol{x}}^{\mathrm{a}}_{i,k} = \boldsymbol{z}^{\mathrm{a}}_{i,k},\ \tilde{\boldsymbol{t}}^{\mathrm{a}}_{i,N} = \boldsymbol{s}^{\mathrm{a}}_{i,N}
\end{aligned} \tag{34}$$

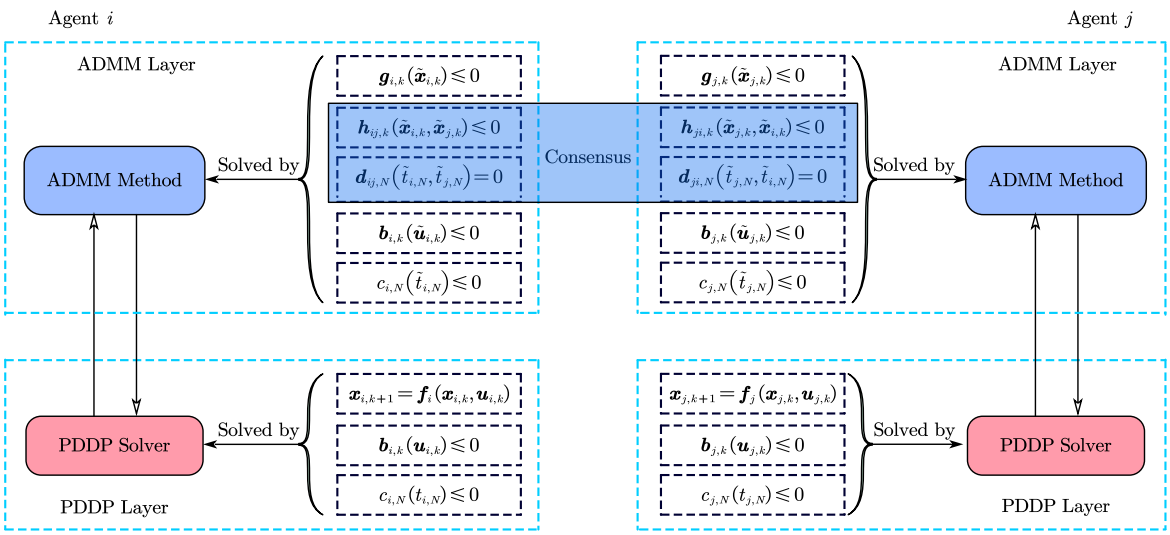


Fig. 1 Schematic diagram of a two-layer structure for distributed spatial-temporal optimization.

where $J_i\left(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}\right) = l_i(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k})$, if $k < N$ and $J_i\left(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}\right) = \phi_i(\boldsymbol{x}_{i,N}, t_{i,N})$ if $k = N$. The indicator function $\mathcal{I}_C(\boldsymbol{x})$ is defined to denote whether the vector $\boldsymbol{x}$ is on the set C. If the vector $\boldsymbol{x}$ is on the set C, then $\mathcal{I}_C(\boldsymbol{x}) = 0$; conversely $\boldsymbol{x}$ is not on C, then $\mathcal{I}_C(\boldsymbol{x}) = +\infty$.

As the derivation process of standard ADMM, the augmented Lagrange multipliers formation of problem (34) is given by

$$
\begin{aligned}
\mathcal{L} = & \sum_{i=1}^{M} \sum_{k=0}^{N} J_i\left(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}\right) + \mathcal{I}_{\boldsymbol{f}_i}\left(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}\right) + \mathcal{I}_{\boldsymbol{b}_{i,k}}\left(\tilde{\boldsymbol{u}}_{i,k}\right) + \mathcal{I}_{c_i}\left(\tilde{t}_{i,N}\right) \\
& + \mathcal{I}_{\boldsymbol{a}_{i,k}}\left(\tilde{\boldsymbol{x}}_{i,k}\right) + \mathcal{I}_{\boldsymbol{d}^{\mathrm{a}}_{i,k}}\left(\tilde{\boldsymbol{x}}^{\mathrm{a}}_{i,k}\right) + \mathcal{I}_{\boldsymbol{e}^{\mathrm{a}}_{i,k}}\left(\tilde{\boldsymbol{t}}^{\mathrm{a}}_{i,N}\right) \\
& + \boldsymbol{\zeta}^{\mathrm{T}}_{i,k}\left(\boldsymbol{u}_{i,k} - \tilde{\boldsymbol{u}}_{i,k}\right) + \boldsymbol{\lambda}^{\mathrm{T}}_{i,k}\left(\boldsymbol{x}_{i,k} - \tilde{\boldsymbol{x}}_{i,k}\right) + {\nu_{i,N}}^{\mathrm{T}}\left(t_{i,N} - \tilde{t}_{i,N}\right) \\
& + \boldsymbol{y}^{\mathrm{T}}_{i,k}\left(\tilde{\boldsymbol{x}}^{\mathrm{a}}_{i,k} - \boldsymbol{z}^{\mathrm{a}}_{i,k}\right) + \boldsymbol{\eta}^{\mathrm{T}}_{i,N}\left(\tilde{\boldsymbol{t}}^{\mathrm{a}}_{i,N} - \boldsymbol{s}^{\mathrm{a}}_{i,N}\right) \\
& + \frac{\tau}{2}\left\|\boldsymbol{u}_{i,k} - \tilde{\boldsymbol{u}}_{i,k}\right\|_2^2 + \frac{\rho}{2}\left\|\boldsymbol{x}_{i,k} - \tilde{\boldsymbol{x}}_{i,k}\right\|_2^2 + \frac{\sigma}{2}\left\|t_{i,N} - \tilde{t}_{i,N}\right\|_2^2 \\
& + \frac{\mu}{2}\left\|\tilde{\boldsymbol{x}}^{\mathrm{a}}_{i,k} - \boldsymbol{z}^{\mathrm{a}}_{i,k}\right\|_2^2 + \frac{\gamma}{2}\left\|\tilde{\boldsymbol{t}}^{\mathrm{a}}_{i,k} - \boldsymbol{s}^{\mathrm{a}}_{i,N}\right\|_2^2
\end{aligned}
\tag{35}
$$

where the dual variables of constraints are represented by $\boldsymbol{\zeta}_{i,k}$, $\boldsymbol{\lambda}_{i,k}$, $\nu_{i,N}$, $\boldsymbol{y}_{i,k}$, $\boldsymbol{\eta}_{i,N}$, and the penalty parameters of constraints are denoted using $\tau, \rho, \sigma, \mu, \gamma > 0$. The D-PDDP is divided into three optimization(computation) steps, which are derived as follows.

Optimization Step 1. Single-Agent PDDP Update: In this update step, PDDP algorithm is employed to minimize the augmented Lagrangian with respect to the local variables $\boldsymbol{x}_{i,k}$, $\boldsymbol{u}_{i,k}$, $t_{i,N}$ as a spatial-temporal trajectory optimizer for a single agent, i.e.,

$$
\left\{\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}, t_{i,N}\right\}^{n+1} = \arg\min \mathcal{L}\left(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}, t_{i,N}, \tilde{\boldsymbol{x}}^{\mathrm{a},n}_{i,k}, \tilde{\boldsymbol{u}}^{n}_{i,k}, \tilde{\boldsymbol{t}}^{\mathrm{a},n}_{i,N}, \boldsymbol{z}^{\mathrm{a},n}_{i,k}, s^{\mathrm{a},n}_{i,N}, \boldsymbol{\zeta}^{n}_{i,k}, \boldsymbol{\lambda}^{n}_{i,k}, \nu^{n}_{i,N}, \boldsymbol{y}^{n}_{i,k}, \boldsymbol{\eta}^{n}_{i,N}\right) \tag{36}
$$

where the number of iterations is denoted by n. This optimization step can be decomposed into subproblems that can be computed in a parallel way for each agent $i \in \mathcal{M}$ in the following form

$$\begin{aligned} \left\{\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}; t_{i,N}\right\}^{n+1} &= \arg\min \sum_{k=1}^{N-1} \left[\hat{l}_i\left(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}; t_{i,N}\right)\right] + \hat{\phi}_i\left(\boldsymbol{x}_{i,N}; t_{i,N}\right) \\ &\text{s.t. } \boldsymbol{x}_{i,k+1} = \boldsymbol{f}_i\left(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}; t_{i,N}\right) \end{aligned} \tag{37}$$

where the running cost $\hat{l}_i\left(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}; t_{i,N}\right)$ and the terminal cost $\hat{\phi}_i\left(\boldsymbol{x}_{i,N}; t_{i,N}\right)$ are revised as

$$\begin{aligned} \hat{l}_i\left(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}; t_{i,N}\right) &= l_i\left(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}; t_{i,N}\right) + \frac{\rho}{2}\left\|\boldsymbol{x}_{i,k} - \tilde{\boldsymbol{x}}_{i,k} + \frac{\boldsymbol{\lambda}_{i,k}}{\rho}\right\|_2^2 + \frac{\tau}{2}\left\|\boldsymbol{u}_{i,k} - \tilde{\boldsymbol{u}}_{i,k} + \frac{\boldsymbol{\zeta}_{i,k}}{\tau}\right\|_2^2 \\ \hat{\phi}_i\left(\boldsymbol{x}_{i,N}; t_{i,N}\right) &= \phi_i\left(\boldsymbol{x}_{i,N}; t_{i,N}\right) + \frac{\rho}{2}\left\|\boldsymbol{x}_{i,N} - \tilde{\boldsymbol{x}}_{i,N} + \frac{\boldsymbol{\lambda}_{i,N}}{\rho}\right\|_2^2 + \frac{\sigma}{2}\left\|t_{i,N} - \tilde{t}_{i,N} + \frac{\nu_{i,N}}{\sigma}\right\|_2^2 \end{aligned} \tag{38}$$

Every agent $i \in \mathcal{M}$ computes a single subproblem above in parallel using the PDDP method. The derivatives of the action value function $Q\left(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}; \boldsymbol{\theta}_i\right)$ are revised as

$$\begin{aligned} Q_{\boldsymbol{x}_i,k} &= l_{\boldsymbol{x}_i,k} + \boldsymbol{f}_{\boldsymbol{x}_i,k}^{\mathrm{T}} V_{\boldsymbol{x}_i,k+1} + \rho\left(\boldsymbol{x}_{i,k} - \tilde{\boldsymbol{x}}_{i,k}\right) + \boldsymbol{\lambda}_{i,k} \\ Q_{\boldsymbol{u}_i,k} &= l_{\boldsymbol{u}_i,k} + \boldsymbol{f}_{\boldsymbol{u}_i,k}^{\mathrm{T}} V_{\boldsymbol{x}_i,k+1} + \tau\left(\boldsymbol{u}_{i,k} - \tilde{\boldsymbol{u}}_{i,k}\right) + \boldsymbol{\zeta}_{i,k} \\ Q_{\boldsymbol{\theta}_i,k} &= l_{\boldsymbol{\theta}_i,k} + V_{\boldsymbol{\theta}_i,k+1} + \boldsymbol{f}_{\boldsymbol{\theta}_i,k}^{\mathrm{T}} V_{\boldsymbol{x}_i,k+1} \\ Q_{\boldsymbol{x}_i\boldsymbol{x}_i,k} &= l_{\boldsymbol{x}_i\boldsymbol{x}_i,k} + \boldsymbol{f}_{\boldsymbol{x}_i,k}^{\mathrm{T}} V_{\boldsymbol{x}_i\boldsymbol{x}_i,k+1} \boldsymbol{f}_{\boldsymbol{x}_i,k} + \rho \boldsymbol{I} \\ Q_{\boldsymbol{u}_i\boldsymbol{u}_i,k} &= l_{\boldsymbol{u}_i\boldsymbol{u}_i,k} + \boldsymbol{f}_{\boldsymbol{u}_i,k}^{\mathrm{T}} V_{\boldsymbol{x}_i\boldsymbol{x}_i,k+1} \boldsymbol{f}_{\boldsymbol{u}_i,k} + \tau \boldsymbol{I} \\ Q_{\boldsymbol{\theta}_i\boldsymbol{\theta}_i,k} &= l_{\boldsymbol{\theta}_i\boldsymbol{\theta}_i,k} + V_{\boldsymbol{\theta}_i\boldsymbol{\theta}_i,k+1} + \boldsymbol{f}_{\boldsymbol{\theta}_i,k}^{\mathrm{T}} V_{\boldsymbol{x}_i\boldsymbol{x}_i,k+1} \boldsymbol{f}_{\boldsymbol{\theta}_i,k} + \boldsymbol{f}_{\boldsymbol{\theta}_i,k}^{\mathrm{T}} V_{\boldsymbol{x}_i\boldsymbol{\theta}_i,k+1} + V_{\boldsymbol{\theta}_i\boldsymbol{x}_i,k+1} \boldsymbol{f}_{\boldsymbol{\theta}_i,k} \\ Q_{\boldsymbol{x}_i\boldsymbol{u}_i,k} &= l_{\boldsymbol{x}_i\boldsymbol{u}_i,k} + \boldsymbol{f}_{\boldsymbol{x}_i,k}^{\mathrm{T}} V_{\boldsymbol{x}_i\boldsymbol{x}_i,k+1} \boldsymbol{f}_{\boldsymbol{u}_i,k} = Q_{\boldsymbol{u}_i\boldsymbol{x}_i,k}^{\mathrm{T}} \\ Q_{\boldsymbol{x}_i\boldsymbol{\theta}_i,k} &= l_{\boldsymbol{x}_i\boldsymbol{\theta}_i,k} + \boldsymbol{f}_{\boldsymbol{x}_i,k}^{\mathrm{T}} V_{\boldsymbol{x}_i\boldsymbol{x}_i,k+1} \boldsymbol{f}_{\boldsymbol{\theta}_i,k} + \boldsymbol{f}_{\boldsymbol{x}_i,k}^{\mathrm{T}} V_{\boldsymbol{x}_i\boldsymbol{\theta}_i,k+1} = Q_{\boldsymbol{\theta}_i\boldsymbol{x}_i,k}^{\mathrm{T}} \\ Q_{\boldsymbol{u}_i\boldsymbol{\theta}_i,k} &= l_{\boldsymbol{u}_i\boldsymbol{\theta}_i,k} + \boldsymbol{f}_{\boldsymbol{u}_i,k}^{\mathrm{T}} V_{\boldsymbol{x}_i\boldsymbol{x}_i,k+1} \boldsymbol{f}_{\boldsymbol{\theta}_i,k} + \boldsymbol{f}_{\boldsymbol{u}_i,k}^{\mathrm{T}} V_{\boldsymbol{u}_i\boldsymbol{\theta}_i,k+1} = Q_{\boldsymbol{\theta}_i\boldsymbol{u}_i,k}^{\mathrm{T}} \end{aligned} \tag{39}$$

For the value functions $V(\boldsymbol{x}_{i,N}, t_{i,N})$ and its derivatives, the terminal conditions will be given by

$$\begin{aligned} V\left(\boldsymbol{x}_{i,N}; t_{i,N}\right) &= \phi_i\left(\boldsymbol{x}_{i,N}; t_{i,N}\right) + \frac{\rho}{2}\left\|\boldsymbol{x}_{i,N} - \tilde{\boldsymbol{x}}_{i,N} + \frac{\boldsymbol{\lambda}_{i,N}}{\rho}\right\|_2^2 + \frac{\sigma}{2}\left\|t_{i,N} - \tilde{t}_{i,N} + \frac{\nu_{i,N}}{\sigma}\right\|_2^2 \\ V_{\boldsymbol{x}_i,N} &= \phi_{\boldsymbol{x}_i,N} + \rho\left(\boldsymbol{x}_{i,N-1} - \tilde{\boldsymbol{x}}_{i,N-1}\right) + \boldsymbol{\lambda}_{i,N-1} \\ V_{\boldsymbol{\theta}_i,N} &= \phi_{\boldsymbol{\theta}_i,N} + \sigma\left(t_{i,N-1} - \tilde{t}_{i,N-1}\right) + \nu_{i,N-1} \\ V_{\boldsymbol{x}_i\boldsymbol{x}_i,N} &= \phi_{\boldsymbol{x}_i\boldsymbol{x}_i,N} + \rho\boldsymbol{I}_{p_i\times p_i} \\ V_{\boldsymbol{x}_i\boldsymbol{\theta}_i,N} &= \phi_{\boldsymbol{x}_i\boldsymbol{\theta}_i,N} = V_{\boldsymbol{\theta}_i\boldsymbol{x}_i,N}^{\mathrm{T}} \\ V_{\boldsymbol{\theta}_i\boldsymbol{\theta}_i,N} &= \phi_{\boldsymbol{\theta}_i\boldsymbol{\theta}_i,N} + \sigma\boldsymbol{I}_{1\times 1} \end{aligned} \tag{40}$$

With the above derivation, the dynamic constraint represented in Fig. 1 is handled. Moreover, the PDDP solver also needs to deal with the constraints $\boldsymbol{b}_{i,k}\left(\boldsymbol{u}_{i,k}\right) \leq 0$ shown in Fig. 1 to prevent the algorithm from diverging due to poor initial guesses. A new control effort is limited within a bounded region using the element-wise clamping, or projection operator, to guarantee numerical stability as

$$\boldsymbol{u}_{i,k} = \min(\max(\boldsymbol{u}_{i,k} + \delta\boldsymbol{u}_{i,k}^*, \boldsymbol{u}_{\min}), \boldsymbol{u}_{\max}) \tag{41}$$

in which the lower and higher boundaries of the control effort are denoted using $\boldsymbol{u}_{\min}$ and $\boldsymbol{u}_{\max}$, respectively.

Likewise, the final time update is also limited to ensure the constraint $c_{i,N}\left(\tilde{t}_{i,N}\right) \leq 0$ in Fig. 1 as

$$t_{i,N} = \min(\max(t_{i,N} + \delta t_{i,N}^*, t_{\min}), t_{\max}) \tag{42}$$

in which the final time's boundaries are denoted using $t_{\min}$ and $t_{\max}$.

Remark 2 *The new local problems* (36) *include a significant modification compared to the unconstrained single-agent problem, i.e., the new objective function added terms that enforce consensus to be reached between local and the copy variables, which will constrain the local constraints of agents to reach consensus between PDDP and ADMM.*

Optimization Step 2. Single-Agent Safe Update: In this update step, ADMM method is utilized to handle the rest local constraints as the following update rule

$$\left\{\tilde{\boldsymbol{x}}_{i,k}^{\mathrm{a}}, \tilde{\boldsymbol{u}}_{i,k}, \tilde{\boldsymbol{t}}_{i,N}^{\mathrm{a}}\right\}^{n+1} = \arg\min \mathcal{L}\left(\boldsymbol{x}_{i,k}^{n+1}, \boldsymbol{u}_{i,k}^{n+1}, t_{i,N}^{n+1}, \tilde{\boldsymbol{x}}_{i,k}^{\mathrm{a}}, \tilde{\boldsymbol{u}}_{i,k}, \tilde{\boldsymbol{t}}_{i,N}^{\mathrm{a}}, \boldsymbol{z}_{i,k}^{\mathrm{a},n}, s_{i,N}^{\mathrm{a},n}, \boldsymbol{\zeta}_{i,k}^{n}, \boldsymbol{\lambda}_{i,k}^{n}, \nu_{i,N}^{n}, \boldsymbol{y}_{i,k}^{n}, \boldsymbol{\eta}_{i,N}^{n}\right) \tag{43}$$

Since the state and control variables of each agent are decoupled, problem (43) can be decomposed to the following

$M \times (N+1)$ state subproblems for every agent i at each time instant k, which can be solved in parallel as

$$\begin{aligned} \tilde{\boldsymbol{x}}_{i,k}^{\mathrm{a},n+1} = \arg\min \frac{\rho}{2} \left\| \boldsymbol{x}_{i,k} - \tilde{\boldsymbol{x}}_{i,k} + \frac{\boldsymbol{\lambda}_{i,k}}{\rho} \right\|_2^2 + \frac{\mu}{2} \left\| \tilde{\boldsymbol{x}}_{i,k}^{\mathrm{a}} - \boldsymbol{z}_{i,k}^{\mathrm{a}} + \frac{\boldsymbol{y}_{i,k}}{\mu} \right\|_2^2 \\ \text{s.t. } \boldsymbol{a}_{i,k}\left(\tilde{\boldsymbol{x}}_{i,k}\right) \leq 0,\ \boldsymbol{d}_{i,k}^{\mathrm{a}}\left(\tilde{\boldsymbol{x}}_{i,k}^{\mathrm{a}}\right) \leq 0 \end{aligned} \tag{44}$$

Similarly, the control effort can be decomposed into $M \times N$ smaller problems for solving

$$\begin{aligned} \tilde{\boldsymbol{u}}_{i,k}^{n+1} = \arg\min \frac{\tau}{2} \left\| \boldsymbol{u}_{i,k} - \tilde{\boldsymbol{u}}_{i,k} + \frac{\boldsymbol{\zeta}_{i,k}}{\tau} \right\|_2^2 \\ \text{s.t.} \quad \boldsymbol{b}_{i,k}\left(\tilde{\boldsymbol{u}}_{i,k}\right) \leq 0 \end{aligned} \tag{45}$$

and M terminal time optimization subproblems

$$\begin{aligned} \tilde{\boldsymbol{t}}_{i,N}^{\mathrm{a},n+1} = \arg\min \frac{\sigma}{2} \left\| t_{i,N} - \tilde{t}_{i,N} + \frac{\nu_{i,N}}{\sigma} \right\|_2^2 + \frac{\gamma}{2} \left\| \tilde{\boldsymbol{t}}_{i,N}^{\mathrm{a}} - \boldsymbol{s}_{i,N}^{\mathrm{a}} + \frac{\boldsymbol{\eta}_{i,N}}{\gamma} \right\|_2^2 \\ \text{s.t. } c_{i,N}\left(\tilde{t}_{i,N}\right) \leq 0,\ \boldsymbol{e}_{i,N}^{\mathrm{a}}\left(\tilde{\boldsymbol{t}}_{i,N}^{\mathrm{a}}\right) = 0 \end{aligned} \tag{46}$$

where the control input constraints $\boldsymbol{b}_{i,k}\left(\tilde{\boldsymbol{u}}_{i,k}\right) \leq 0$ can be handled in a similar way as step 1 and the terminal time boundary constraints $c_{i,N}\left(\tilde{t}_{i,N}\right) \leq 0$ will be solved with the constraint $\boldsymbol{e}_{i,N}^{\mathrm{a}}\left(\tilde{\boldsymbol{t}}_{i,N}^{\mathrm{a}}\right) = 0$ by the standard solver.

Remark 3 *Optimization Step enables to decompose* (43) *into* $M(2N+2)$ *low-dimensional subproblems and hence the computation can performed in parallel for each agent as well as each time instant, which can greatly improve the computational efficiency of the algorithm. With reasonable convexification, the above subproblems* (44)*,* (45)*,* (46) *can be transformed into Quadratic Programming (QP) problems, and thus solved by well-established QP optimization toolbox.*

Optimization Step 3. Global Update: After the local optimization step, the global variables $\boldsymbol{z}_{i,k}$, $\boldsymbol{s}_{i,N}$ are updated, i.e.,

$$\left\{\boldsymbol{z}_{i,k}^{\mathrm{a}}, \boldsymbol{s}_{i,N}^{\mathrm{a}}\right\}^{n+1} = \arg\min \mathcal{L}\left(\boldsymbol{x}_{i,k}^{n+1}, \boldsymbol{u}_{i,k}^{n+1}, t_{i,N}^{n+1}, \tilde{\boldsymbol{x}}_{i,k}^{\mathrm{a},n+1}, \tilde{\boldsymbol{u}}_{i,k}^{n+1}, \tilde{\boldsymbol{t}}_{i,N}^{\mathrm{a},n+1}, \boldsymbol{z}_{i,k}^{\mathrm{a}}, \boldsymbol{s}_{i,N}^{\mathrm{a}}, \boldsymbol{\zeta}_{i,k}^{n}, \boldsymbol{\lambda}_{i,k}^{n}, \nu_{i,N}^{n}, \boldsymbol{y}_{i,k}^{n}, \boldsymbol{\eta}_{i,N}^{n}\right) \tag{47}$$

According to constraint (31), we have

$$\left\{\boldsymbol{z}_{i,k}, \boldsymbol{s}_{i,N}\right\}^{n+1} = \arg\min \sum_{j \in \mathcal{P}_i} \frac{\mu}{2} \left\| \tilde{\boldsymbol{x}}_{i,k}^{j} - \boldsymbol{z}_{i,k} + \frac{\boldsymbol{y}_{i,k}}{\mu} \right\|_2^2 + \frac{\gamma}{2} \left\| \tilde{\boldsymbol{t}}_{i,N}^{j} - \boldsymbol{s}_{i,N} + \frac{\boldsymbol{\eta}_{i,N}}{\gamma} \right\|_2^2 \tag{48}$$

which means that the global consensus variable at agent i is the combined evaluation of the state from the perspective of

i's neighbor $j \in \mathcal{N}_i$, and thus the update rules of the global variables is given by

$$\boldsymbol{z}_{i,k}^{n+1} = \frac{1}{|\mathcal{P}_i|} \sum_{j \in \mathcal{P}_i} \left(\tilde{\boldsymbol{x}}_{i,k}^{j,n+1} + \frac{1}{\mu} \boldsymbol{y}_{i,k}^{j,n} \right) \tag{49a}$$

$$\boldsymbol{s}_{i,N}^{n+1} = \frac{1}{|\mathcal{P}_i|} \sum_{j \in \mathcal{P}_i} \left(\tilde{t}_{i,N}^{j,n+1} + \frac{1}{\gamma} \boldsymbol{\eta}_{i,N}^{j,n} \right) \tag{49b}$$

where $\left\{ \boldsymbol{z}_{j,k} \right\}_{j \in \mathcal{N}_i \setminus \{i\}}$ of $\boldsymbol{z}_{i,k}^{\mathrm{a}}$ is obtained from neighbors of agent i through communication. Since the dual variables $\boldsymbol{y}_{i,k}$ and $\boldsymbol{\eta}_{i,N}$ will gradually become 0 as the iterations n increase, it can be seen from Eq. (49) that the global update is gradually a local average of all copy variables of agent i. Finally, the dual updates are given as

$$\boldsymbol{\zeta}_{i,k}^{n+1} = \boldsymbol{\zeta}_{i,k}^{n} + \tau \left(\boldsymbol{u}_{i,k}^{n+1} - \tilde{\boldsymbol{u}}_{i,k}^{n+1} \right) \tag{50a}$$

$$\boldsymbol{\lambda}_{i,k}^{n+1} = \boldsymbol{\lambda}_{i,k}^{n} + \rho \left(\boldsymbol{x}_{i,k}^{n+1} - \tilde{\boldsymbol{x}}_{i,k}^{n+1} \right) \tag{50b}$$

$$\boldsymbol{y}_{i,k}^{n+1} = \boldsymbol{y}_{i,k}^{n} + \mu \left(\tilde{\boldsymbol{x}}_{i,k}^{\mathrm{a},n+1} - \boldsymbol{z}_{i,k}^{\mathrm{a},n+1} \right) \tag{50c}$$

$$\nu_{i,N}^{n+1} = \nu_{i,N}^{n} + \sigma \left(t_{i,N}^{n+1} - \tilde{t}_{i,N}^{n+1} \right) \tag{50d}$$

$$\boldsymbol{\eta}_{i,N}^{n+1} = \boldsymbol{\eta}_{i,N}^{n} + \gamma \left(\tilde{\boldsymbol{t}}_{i,N}^{\mathrm{a},n+1} - \boldsymbol{s}_{i,N}^{\mathrm{a},n+1} \right) \tag{50e}$$

The stopping criterion for the update of the D-PDDP algorithm can be determined by checking whether the residuals of the primal and dual variables of the algorithm, named primal-residual and dual-residual, are below a certain threshold [25]. The primal and dual residuals of the D-PDDP algorithm at the n-th iteration are defined as:

$$\begin{aligned}
\boldsymbol{r}_{\boldsymbol{U}}^{\mathrm{p},n} &= \boldsymbol{U}^{n} - \tilde{\boldsymbol{U}}^{n} & \boldsymbol{r}_{\boldsymbol{U}}^{\mathrm{d},n} &= \tau(\tilde{\boldsymbol{U}}^{n} - \tilde{\boldsymbol{U}}^{n-1}) \\
\boldsymbol{r}_{\boldsymbol{X}}^{\mathrm{p},n} &= \boldsymbol{X}^{n} - \tilde{\boldsymbol{X}}^{n} & \boldsymbol{r}_{\boldsymbol{X}}^{\mathrm{d},n} &= \rho(\tilde{\boldsymbol{X}}^{n} - \tilde{\boldsymbol{X}}^{n-1}) \\
\boldsymbol{r}_{\tilde{\boldsymbol{X}}}^{\mathrm{p},n} &= \tilde{\boldsymbol{X}}^{\mathrm{a},n} - \boldsymbol{Z}^{\mathrm{a},n} & \boldsymbol{r}_{\tilde{\boldsymbol{X}}}^{\mathrm{d},n} &= \mu(\boldsymbol{Z}^{\mathrm{a},n} - \boldsymbol{Z}^{\mathrm{a},n-1}) \\
\boldsymbol{r}_{\boldsymbol{T}}^{\mathrm{p},n} &= \boldsymbol{T}^{n} - \tilde{\boldsymbol{T}}^{n} & \boldsymbol{r}_{\boldsymbol{T}}^{\mathrm{d},n} &= \sigma(\tilde{\boldsymbol{T}}^{n} - \tilde{\boldsymbol{T}}^{n-1}) \\
\boldsymbol{r}_{\tilde{\boldsymbol{T}}}^{\mathrm{p},n} &= \tilde{\boldsymbol{T}}^{\mathrm{a},n} - \boldsymbol{S}^{\mathrm{a},n} & \boldsymbol{r}_{\tilde{\boldsymbol{T}}}^{\mathrm{d},n} &= \gamma(\boldsymbol{S}^{n} - \boldsymbol{S}^{n-1})
\end{aligned} \tag{51}$$

where $\boldsymbol{r}_{(\cdot)}^{\mathrm{p},n}$ represents the primal residual and $\boldsymbol{r}_{(\cdot)}^{\mathrm{d},n}$ denotes the dual residual. $\boldsymbol{U} = \left[\boldsymbol{U}_1^{\mathrm{T}}, \ldots, \boldsymbol{U}_i^{\mathrm{T}}, \ldots, \boldsymbol{U}_M^{\mathrm{T}} \right]^{\mathrm{T}}$ is the slack of the control variables from all agents and the entire time history. Similarly, we can define other vectors $\boldsymbol{X}$, $\boldsymbol{Z}$, $\boldsymbol{T}$, $\boldsymbol{S}$. The residuals here are a synthesized evaluation of the stacked vectors of all UAVs at all time instants. The primal and dual residuals can be used to monitor convergence of the optimization algorithm, i.e., the primal residual $\boldsymbol{r}_{(\cdot)}^{\mathrm{p},n}$ tends to zero when the algorithm update satisfies the consensus constraints in Problem (34) and the dual residual $\boldsymbol{r}_{(\cdot)}^{\mathrm{d},n}$ tends to

zero when the algorithm is updated close to the target minimum [25]. Hence, the criterion for determining that the update satisfies the stopping condition is:

$$
\begin{aligned}
\left\|\boldsymbol{r}_{\boldsymbol{U}}^{\mathrm{p},n}\right\|_2 &\leq \sqrt{\mathcal{N}_{\boldsymbol{U}}}\epsilon_{\mathrm{abs}} + \epsilon^{\mathrm{rel}} \max\left\{\|\boldsymbol{U}^n\|_2, \left\|\tilde{\boldsymbol{U}}^n\right\|_2\right\} \qquad \left\|\boldsymbol{r}_{\boldsymbol{U}}^{\mathrm{d},n}\right\|_2 &\leq \sqrt{\mathcal{N}_{\boldsymbol{U}}}\epsilon_{\mathrm{abs}} + \epsilon^{\mathrm{rel}} \|\boldsymbol{\Xi}^n\|_2 \\
\left\|\boldsymbol{r}_{\boldsymbol{X}}^{\mathrm{p},n}\right\|_2 &\leq \sqrt{\mathcal{N}_{\boldsymbol{X}}}\epsilon_{\mathrm{abs}} + \epsilon^{\mathrm{rel}} \max\left\{\|\boldsymbol{X}^n\|_2, \left\|\tilde{\boldsymbol{X}}^n\right\|_2\right\} \qquad \left\|\boldsymbol{r}_{\boldsymbol{X}}^{\mathrm{d},n}\right\|_2 &\leq \sqrt{\mathcal{N}_{\boldsymbol{X}}}\epsilon_{\mathrm{abs}} + \epsilon^{\mathrm{rel}} \|\boldsymbol{\Lambda}^n\|_2 \\
\left\|\boldsymbol{r}_{\boldsymbol{T}}^{\mathrm{p},n}\right\|_2 &\leq \sqrt{\mathcal{N}_{\boldsymbol{T}}}\epsilon_{\mathrm{abs}} + \epsilon^{\mathrm{rel}} \max\left\{\|\boldsymbol{T}^n\|_2, \left\|\tilde{\boldsymbol{T}}^n\right\|_2\right\} \qquad \left\|\boldsymbol{r}_{\boldsymbol{T}}^{\mathrm{d},n}\right\|_2 &\leq \sqrt{\mathcal{N}_{\boldsymbol{T}}}\epsilon_{\mathrm{abs}} + \epsilon^{\mathrm{rel}} \|\boldsymbol{\Pi}^n\|_2 \\
\left\|\boldsymbol{r}_{\boldsymbol{Z}}^{\mathrm{p},n}\right\|_2 &\leq \sqrt{\mathcal{N}_{\tilde{\boldsymbol{X}}^{\mathrm{a}}}}\epsilon_{\mathrm{abs}} + \epsilon^{\mathrm{rel}} \max\left\{\left\|\tilde{\boldsymbol{X}}^{\mathrm{a},n}\right\|_2, \|\boldsymbol{Z}^{\mathrm{a},n}\|_2\right\} \qquad \left\|\boldsymbol{r}_{\boldsymbol{Z}}^{\mathrm{d},n}\right\|_2 &\leq \sqrt{\mathcal{N}_{\tilde{\boldsymbol{X}}^{\mathrm{a}}}}\epsilon_{\mathrm{abs}} + \epsilon^{\mathrm{rel}} \|\boldsymbol{Y}^n\|_2 \\
\left\|\boldsymbol{r}_{\boldsymbol{S}}^{\mathrm{p},n}\right\|_2 &\leq \sqrt{\mathcal{N}_{\tilde{\boldsymbol{T}}^{\mathrm{a}}}}\epsilon_{\mathrm{abs}} + \epsilon^{\mathrm{rel}} \max\left\{\left\|\tilde{\boldsymbol{T}}^{\mathrm{a},n}\right\|_2, \|\boldsymbol{S}^{\mathrm{a},n}\|_2\right\} \qquad \left\|\boldsymbol{r}_{\boldsymbol{S}}^{\mathrm{d},n}\right\|_2 &\leq \sqrt{\mathcal{N}_{\tilde{\boldsymbol{T}}^{\mathrm{a}}}}\epsilon_{\mathrm{abs}} + \epsilon^{\mathrm{rel}} \|\boldsymbol{V}^n\|_2
\end{aligned} \tag{52}
$$

where $\epsilon_{\mathrm{abs}} \geq 0$ is the absolute tolerance, $\epsilon_{\mathrm{rel}} \geq 0$ is the relative tolerance, and the coefficients $\mathcal{N}_{(\cdot)}$ denote the dimensions of the corresponding vector. The choice of the relative and absolute tolerances depends on the size of the typical variable values in different applications [25].

By summarizing the previous derivation results, the pseudo-code in Algorithm 2 shows the detailed procedure of the proposed D-PDDP. It is worth noting that a single agent problem is computed once at the beginning of the algorithm using the original DDP (considering only the dynamics constraints (1)) to warm-start the subsequent algorithms (lines 1-3). During the iterations of the ADMM algorithm, Optimization Steps 1, 2 and 3 are strictly followed (lines 5-12) until the computational results satisfy specific convergence criteria. In particular, it should be noted that necessary information communication is required before updating the global and dual variables. All computational steps (lines 1, 5-8, 10, 12) of our proposed D-PDDP method could be determined by each agent i in a parallel way, while each agent i should share informations with their neighbors through communication (lines 2, 9, 11). Both computational and communication properties reflect the fully decentralized nature of D-PDDP.

Remark 4 *The stopping criterion of the ADMM algorithm can be determined by checking whether the residuals calculated from the primal and dual variables, named the primal residual and the dual residual, are below a certain threshold [25]. However, it is important to note that the computation of the residuals requires gathering information of all agents in the algorithm, which would destroy the distributed nature of the D-PDDP.*

Remark 5 *For distributed trajectory optimization problem of UAV swarm, the number of the swarm is large, and the nonlinear constraints are coupled with each other. For the practical computation, the performance of the algorithm is indeed sensitive to the initial guess as the convergence of the algorithm is related to both the initial flight time and the interference of the trajectory with obstacles and other UAV trajectories. For this reason, it is reasonable to use fast single agent trajectory optimization techniques, e.g., DDP algorithm, to compute an initial trajectory that satisfies the dynamics and flight time for each UAV trajectory independently.*

Remark 6 *Whether there is a minimum set of neighbors that must be satisfied to guarantee convergence of the algorithm and satisfaction of the constraints is a theoretical aspect that has not yet been derived and investigated in this paper. However, in practical algorithmic studies, the size of the neighbor set can theoretically be changed in the process of trajectory generation, but changing the size of the neighbor set in the process of optimization is likely to influence the convergence, e.g., convergence speed, of the algorithm. If the set of neighbors is too small, each UAV can communicate with fewer neighbors and obtain less information, which may lead to the swarm trajectories needing more iterations to converge.*

Algorithm 2 Distributed Parameterized Differential Dynamic Programming

Initial guesses $(\bar{\boldsymbol{x}}_{i,k}, \bar{\boldsymbol{u}}_{i,k}, \bar{t}_{i,N})$ generated by DDP for each agent $i \in \mathcal{M}$.
Each agent $i \in \mathcal{M}$ communicates with all its neighbors $j \in \mathcal{N}_i \backslash \{i\}$ and receives $(\bar{\boldsymbol{x}}_{j,k}, \bar{t}_{j,N})$.
Initialize: $\boldsymbol{x}_{i,k} \leftarrow \bar{\boldsymbol{x}}_{i,k}, \boldsymbol{u}_{i,k} \leftarrow \bar{\boldsymbol{u}}_{i,k}, t_{i,N} \leftarrow \bar{t}_{i,N}, \tilde{\boldsymbol{x}}^{\mathrm{a}}_{i,k} \leftarrow \left[\{\bar{\boldsymbol{x}}_j\}_{j\in\mathcal{N}_i}\right], \tilde{t}^{\mathrm{a}}_{i,N} \leftarrow \left[\{\bar{t}_j\}_{j\in\mathcal{N}_i}\right], \tilde{u}_{i,k} \leftarrow \bar{\boldsymbol{u}}_{i,k}, \boldsymbol{z}^{\mathrm{a}}_{i,k} \leftarrow \tilde{\boldsymbol{x}}^{\mathrm{a}}_{i,k}, \boldsymbol{s}^{\mathrm{a}}_{i,N} \leftarrow \tilde{t}^{\mathrm{a}}_{i,N}, \boldsymbol{\zeta}_{i,k} \leftarrow 0, \boldsymbol{\lambda}_{i,k} \leftarrow 0, \boldsymbol{y}_{i,k} \leftarrow 0, \nu_{i,N} \leftarrow 0, \boldsymbol{\eta}_{i,N} \leftarrow 0.$
while $n \le iter_{\max}$ **do**
  $(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}, t_{i,N}) \leftarrow$ Eq. (37) in parallel for each agent $i \in \mathcal{M}$.
  $\tilde{\boldsymbol{x}}^{\mathrm{a}}_{i,k} \leftarrow$ Eq. (44) in parallel for each agent $i \in \mathcal{M}$.
  $\tilde{\boldsymbol{u}}_{i,k} \leftarrow$ Eq. (45) in parallel for each agent $i \in \mathcal{M}$.
  $\tilde{t}^{\mathrm{a}}_{i,N} \leftarrow$ Eq. (46) in parallel for each agent $i \in \mathcal{M}$.
  Each agent $i \in \mathcal{M}$ communicates with $j \in \mathcal{P}_i \backslash \{i\}$ and receives $(\tilde{\boldsymbol{x}}^{j}_{i,k}, \tilde{t}^{j}_{i,N})$.
  $(\boldsymbol{z}_{i,k}, \boldsymbol{s}_{i,N}) \leftarrow$ in parallel for each agent $i \in \mathcal{M}$
  Each agent $i \in \mathcal{M}$ communicates with $j \in \mathcal{N}_i \backslash \{i\}$ and receives $(\boldsymbol{z}_{j,k}, s_{j,N})$.
  $\boldsymbol{\zeta}_{i,k}, \boldsymbol{\lambda}_{i,k}, \boldsymbol{y}_{i,k}, \nu_{i,N}, \boldsymbol{\eta}_{i,N} \leftarrow$ Update Eq. (50) in parallel $\forall i \in \mathcal{M}$.
  Calculate residuals using Eq. (51).
  if Eq. (52) **then**
    End the algorithm.
  end if
end while

IV. Adaptive Penalty Parameter Based Acceleration for D-PDDP

The setting of the penalty parameters in the original ADMM algorithm is important because it has a significant influence on the algorithm's convergence rate. As a result, a distributed adaptive acceleration scheme incorporated into D-PDDP updates to accelerate the convergence of the algorithm by introducing an 'agent-specific' penalty parameter tuning method for each agent is proposed in this section. It is known that the ADMM update is equivalent to performing a Douglas Rachford Splitting (DRS) on the dual of the original problem [31]. Based on this idea, we propose an adaptive penalty parameter updating rule, that leverage different penalty parameters for the state, control and time variables to take into account the constraint satisfaction of different agents, for the MAOC problem.

A. Derivition of Spectral Stepsize

The specific derivation of the adaptive penalty parameter rule will be presented below using the primal variable $\boldsymbol{x}_{i,k}$, its copy variable $\tilde{\boldsymbol{x}}_{i,k}$ and their dual variable $\boldsymbol{\lambda}_{i,k}$ as the example for the sake of brevity of the article. First, define

$$\begin{aligned} H\left(\boldsymbol{x}_{i,k}\right) &= J_i\left(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}\right) + \mathcal{I}_{\boldsymbol{f}_i}\left(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}\right) \\ G\left(\tilde{\boldsymbol{x}}_{i,k}\right) &= \mathcal{I}_{\boldsymbol{a}_{i,k}}\left(\tilde{\boldsymbol{x}}_{i,k}, \tilde{\boldsymbol{u}}_{i,k}\right) \end{aligned} \tag{53}$$

Based on the equivalence of the ADMM and the DRS on the dual problem of the original problem, problem (34) with respect to $\boldsymbol{x}_{i,k}$ and $\tilde{\boldsymbol{x}}_{i,k}$ can be reformulated as its dual form as

$$\begin{aligned} &\min_{\boldsymbol{\lambda}_{i,k}} H^*\left(\boldsymbol{\lambda}_{i,k}\right) + G^*\left(\boldsymbol{\lambda}_{i,k}\right) \\ &\text{s.t.} \quad \boldsymbol{x}_{i,k} - \tilde{\boldsymbol{x}}_{i,k} = 0 \end{aligned} \tag{54}$$

where H^*, G^* denotes the Fenchel conjugate of H, G, defined as $H^*(\boldsymbol{\lambda}_{i,k}) = \sup_{\boldsymbol{x}_{i,k}} \langle \boldsymbol{x}_{i,k}, \boldsymbol{\lambda} \rangle - H(\boldsymbol{x}_{i,k})$ and $G^*(\boldsymbol{\lambda}_{i,k}) = \sup_{-\tilde{\boldsymbol{x}}_{i,k}} \langle -\tilde{\boldsymbol{x}}_{i,k}, \boldsymbol{\lambda} \rangle - G(-\tilde{\boldsymbol{x}}_{i,k})$. An important property is that $\boldsymbol{\lambda}_{i,k} \in \partial H(\boldsymbol{x}_{i,k})$ if and only if $\boldsymbol{x}_{i,k} \in \partial H^*(\boldsymbol{\lambda}_{i,k})$, and $\boldsymbol{\lambda}_{i,k} \in \partial G(-\tilde{\boldsymbol{x}}_{i,k})$ if and only if $-\tilde{\boldsymbol{x}}_{i,k} \in \partial G^*(\boldsymbol{\lambda}_{i,k})$ [43].

According to the DRS, dual problem (54) can be decomposed into two steps for solving, i.e.,

$$0 \in \frac{\hat{\boldsymbol{\lambda}}_{i,k}^{n+1} - \boldsymbol{\lambda}_{i,k}^{n}}{\rho_i} + \partial H^*\left(\hat{\boldsymbol{\lambda}}_{i,k}^{n+1}\right) + \partial G^*\left(\boldsymbol{\lambda}_{i,k}^{n}\right) \tag{55a}$$

$$0 \in \frac{\boldsymbol{\lambda}_{i,k}^{n+1} - \boldsymbol{\lambda}_{i,k}^{n}}{\rho_i} + \partial H^*\left(\hat{\boldsymbol{\lambda}}_{i,k}^{n+1}\right) + \partial G^*\left(\boldsymbol{\lambda}_{i,k}^{n+1}\right) \tag{55b}$$

where $\partial H^*(\cdot)$ and $\partial G^*(\cdot)$ stands for the subgradient of $H^*(\cdot)$ and $G^*(\cdot)$. And ρ_i is no longer a fixed value, but a variable that changes according to the different agents and is equal at each time instant k. The intermediate variable $\hat{\boldsymbol{\lambda}}_{i,k+1}$ is defined as

$$\hat{\boldsymbol{\lambda}}_{i,k}^{n+1} = \boldsymbol{\lambda}_{i,k}^{n} + \rho_i^n\left(\boldsymbol{x}_{i,k}^{n+1} - \tilde{\boldsymbol{x}}_{i,k}^{n}\right) \tag{56}$$

where $\boldsymbol{x}_{i,k}^{n+1}$ and $\tilde{\boldsymbol{x}}_{i,k}^{n}$ are variables of different iterations. According to the important property of $H(\cdot)$ and $H^*(\cdot)$, we can readily obtain

$$\begin{aligned} \boldsymbol{x}_{i,k}^{n+1} &\in \partial H^*(\hat{\boldsymbol{\lambda}}_{i,k}^{n+1}) \\ -\tilde{\boldsymbol{x}}_{i,k}^{n+1} &\in \partial G^*(\boldsymbol{\lambda}_{i,k}^{n+1}) \end{aligned} \tag{57}$$

Due to the equivalence between the ADMM step and the DRS of the dual problem, we can obtain the following

update formula

$$\begin{aligned}\hat{\boldsymbol{\lambda}}_{i,k}^{n+1} &= \boldsymbol{\lambda}_{i,k}^{n} + \rho_i^n \left(\boldsymbol{x}_{i,k}^{n+1} - \tilde{\boldsymbol{x}}_{i,k}^{n}\right) \\ &\in \boldsymbol{\lambda}_{i,k}^{n} + \rho_i^n \left(\partial H^*(\hat{\lambda}_{i,k}^{n+1}) + \partial G^*(\lambda_{i,k}^{n})\right) \\ \boldsymbol{\lambda}_{i,k}^{n+1} &= \boldsymbol{\lambda}_{i,k}^{n} + \rho_i^n \left(\boldsymbol{x}_{i,k}^{n+1} - \tilde{\boldsymbol{x}}_{i,k}^{n+1}\right) \\ &\in \boldsymbol{\lambda}_{i,k}^{n} + \rho_i^n \left(\partial H^*(\hat{\lambda}_{i,k}^{n+1}) + \partial G^*(\lambda_{i,k}^{n+1})\right)\end{aligned} \tag{58}$$

Inspired by the fact that the spectral gradient method mimics the hessian of the original function to find a quasi-Newton step [40, 44], the gradients of the functions H^* and G^* can be approximated as linear functions as

$$\begin{aligned}\partial H^*(\hat{\boldsymbol{\lambda}}_{i,k}^{n}) &\approx \alpha_i^n \hat{\boldsymbol{\lambda}}_{i,k}^{n} + \boldsymbol{a}_i^n \\ \partial G^*(\boldsymbol{\lambda}_{i,k}^{n}) &\approx \beta_i^n \boldsymbol{\lambda}_{i,k}^{n} + \boldsymbol{b}_i^n\end{aligned} \tag{59}$$

where $\alpha_i^n > 0, \beta_i^n > 0$ are the first-order linear approximations of the dual functions $\partial H^*(\hat{\boldsymbol{\lambda}}_{i,k}^{n})$ and $\partial G^*(\boldsymbol{\lambda}_{i,k}^{n})$, and the constant vector $\boldsymbol{a}_i^n, \boldsymbol{b}_i^n \in \mathbb{R}^r$.

Substituting Eq. (59) into DRS step (55), we can obtain the relationship between $\boldsymbol{\lambda}_{i,k}^{n+1}$ and $\hat{\boldsymbol{\lambda}}_{i,k}^{n+1}$ with respect to $\boldsymbol{\lambda}_{i,k}^{n}$ as

$$\begin{aligned}\hat{\boldsymbol{\lambda}}_{i,k}^{n+1} &= \frac{1-\beta_i^n\rho_i^n}{1+\alpha_i^n\rho_i^n}\boldsymbol{\lambda}_{i,k}^{n} - \frac{\boldsymbol{a}_i^n\rho_i^n + \boldsymbol{b}_i^n\rho_i^n}{1+\alpha_i^n\rho_i^n} \\ \boldsymbol{\lambda}_{i,k}^{n+1} &= \frac{\left(1+\alpha_i^n\beta_i^n{\rho_i^n}^2\right)\boldsymbol{\lambda}_{i,k}^{n} - (\boldsymbol{a}_i^n + \boldsymbol{b}_i^n)\rho_i^n}{(1+\alpha_i^n\rho_i^n)(1+\beta_i^n\rho_i^n)}\end{aligned} \tag{60}$$

The residual at $\boldsymbol{\lambda}_{i,k}^{n+1}$ can be expressed as the norm of the subgradient $\partial H^*(\boldsymbol{\lambda}_{i,k})$ and $\partial G^*(\boldsymbol{\lambda}_{i,k})$, i.e.,

$$\begin{aligned}\left\|\boldsymbol{r}_{\boldsymbol{X}}^{\mathrm{d,n}}\right\|_2 &= \left\|(\alpha_i^n + \beta_i^n)\boldsymbol{\Lambda}^{n+1} + (\boldsymbol{a}_i^n + \boldsymbol{b}_i^n)\right\|_2 \\ &= \frac{1+\alpha_i^n\beta_i^n(\rho_i^n)^2}{(1+\alpha_i^n\rho_i^n)(1+\beta_i^n\rho_i^n)}\left\|(\alpha_i^n + \beta_i^n)\boldsymbol{\Lambda}^{n} + (\boldsymbol{a}_i^n + \boldsymbol{b}_i^n)\right\|_2\end{aligned} \tag{61}$$

where $\boldsymbol{\Lambda}$ is the slack of $\boldsymbol{\lambda}_{i,k}$ from all agents and all time instants.

The optimal penalty parameter for the n-th iteration is obtained by minimizing the residual $\left\|\boldsymbol{r}_{\boldsymbol{X}}^{\mathrm{d,n}}\right\|_2$ as

$$\begin{aligned}\rho_i^n &= \arg\min_{\rho_i}\left\|\boldsymbol{r}_{\boldsymbol{X}}^{\mathrm{d,n}}\right\|_2 \\ &= \arg\max_{\rho_i}\frac{(1+\alpha_i^n\rho_i)(1+\beta_i^n\rho_i)}{1+\alpha_i^n\beta_i^n\rho_i^2} \\ &= \arg\max_{\rho_i}\frac{(\alpha_i^n+\beta_i^n)\rho_i}{1+\alpha_i^n\beta_i^n\rho_i^2} = 1/\sqrt{\alpha_i^n\beta_i^n}\end{aligned} \tag{62}$$

B. Local Curvature Estimation

After the expression of the optimal step size is derived, the curvature information α_i^n, β_i^n needs to be determined. With (57) and (59), it can be obtained that

$$\begin{aligned} \boldsymbol{x}_{i,k}^n &\approx \alpha_i^n \hat{\boldsymbol{\lambda}}_{i,k}^n + \boldsymbol{a}_i^n \\ -\tilde{\boldsymbol{x}}_{i,k}^n &\approx \beta_i^n \boldsymbol{\lambda}_{i,k}^n + \boldsymbol{b}_i^n \end{aligned} \tag{63}$$

The estimates for these curvature parameters are derived from the outcomes of iteration n as well as previous iterations $n_l < n$ (α_i^n, β_i^n can be computed between a few iterations rather than at each iteration). Taking the estimation of α_i^n as an example, the following definitions are given as

$$\begin{aligned} \Delta \boldsymbol{x}_{i,k}^n &= \boldsymbol{x}_{i,k}^n - \boldsymbol{x}_{i,k}^{n_l} \\ \Delta \hat{\boldsymbol{\lambda}}_{i,k}^n &\coloneqq \hat{\boldsymbol{\lambda}}_{i,k}^n - \hat{\boldsymbol{\lambda}}_{i,k}^{n_l} \\ \Delta H_{i,k}^* &\coloneqq \partial H^*(\hat{\boldsymbol{\lambda}}_{i,k}^n) - \partial H^*(\hat{\boldsymbol{\lambda}}_{i,k}^{n_l}) \end{aligned} \tag{64}$$

Considering the property of $\partial H^*(\hat{\boldsymbol{\lambda}}_{i,k}^n)$ in Eq. (57) and the linear approximating for $\partial H^*(\hat{\boldsymbol{\lambda}}_{i,k}^n)$ in Eq. (59), we can get $\Delta \boldsymbol{x}_{i,k}^n \in \Delta H_{i,k}^*$ and $\Delta H_{i,k}^* \approx \alpha_i^n \Delta \hat{\boldsymbol{\lambda}}_{i,k}^n + \boldsymbol{a}_i^n$. Then the value of curvature α_i^n can be estimated by the least square problem as [45]

$$\min_{\alpha_i^n} \left\| \Delta \boldsymbol{x}_{i,k}^n - \alpha_i^n \Delta \hat{\boldsymbol{\lambda}}_{i,k}^n \right\|_2^2 \text{ or } \min_{\alpha_i^n} \left\| (\alpha_i^n)^{-1} \Delta \boldsymbol{x}_{i,k}^n - \Delta \hat{\boldsymbol{\lambda}}_{i,k}^n \right\|_2^2 \tag{65}$$

Defining $\hat{\alpha}_i^n = 1/\alpha_i^n$ for notational convenience, the closed-form solutions of the curvature estimates $\hat{\alpha}_i^n$ corresponding to the preceding two least square problems are

$$\hat{\alpha}_i^{n,\mathrm{SD}} = \frac{\left\langle \Delta \hat{\boldsymbol{\lambda}}_{i,k}^n, \Delta \hat{\boldsymbol{\lambda}}_{i,k}^n \right\rangle}{\left\langle \Delta \boldsymbol{x}_{i,k}^n, \Delta \hat{\boldsymbol{\lambda}}_{i,k}^n \right\rangle} \text{ and } \hat{\alpha}_i^{n,\mathrm{MG}} = \frac{\left\langle \Delta \boldsymbol{x}_{i,k}^n, \Delta \hat{\boldsymbol{\lambda}}_{i,k}^n \right\rangle}{\left\langle \Delta \boldsymbol{x}_{i,k}^n, \Delta \boldsymbol{x}_{i,k}^n \right\rangle} \tag{66}$$

in which SD denotes the steepest descent while MG denotes the minimum gradient, and $\hat{\alpha}_i^{n,\mathrm{SD}} \geq \hat{\alpha}_i^{n,\mathrm{MG}}$ resulting from Cauchy-Schwarz inequality [45]. A hybrid stepsize calculation rule is utilized for better convergence as

$$\hat{\alpha}_i^n = \begin{cases} \hat{\alpha}_i^{n,\mathrm{MG}} & \text{if } 2\hat{\alpha}_i^{n,\mathrm{MG}} > \hat{\alpha}_i^{n,\mathrm{SD}} \\ \hat{\alpha}_i^{n,\mathrm{SD}} - \hat{\alpha}_i^{n,\mathrm{MG}}/2 & \text{otherwise.} \end{cases} \tag{67}$$

Similarly, the dual functions $G_i^*(\boldsymbol{\lambda}_i^n)$ has the same property, thus we can get the spectral stepsize $\hat{\beta}^n = 1/\beta^n$ with the same derivation process as

$$\hat{\beta}_i^n = \begin{cases} \hat{\beta}_i^{n,\mathrm{MG}} & \text{if } 2\hat{\beta}_i^{n,\mathrm{MG}} > \hat{\beta}_i^{n,\mathrm{SD}} \\ \hat{\beta}_i^{n,\mathrm{SD}} - \hat{\beta}_i^{n,\mathrm{MG}}/2 & \text{otherwise.} \end{cases} \tag{68}$$

where $\hat{\beta}_i^{n,\mathrm{SD}} = \left\langle \Delta\boldsymbol{\lambda}_{i,k}^n, \Delta\boldsymbol{\lambda}_{i,k}^n \right\rangle / \left\langle \Delta\tilde{\boldsymbol{x}}_{i,k}^n, \Delta\boldsymbol{\lambda}_{i,k}^n \right\rangle$, $\hat{\beta}_i^{n,\mathrm{MG}} = \left\langle \Delta\tilde{\boldsymbol{x}}_{i,k}^n, \Delta\boldsymbol{\lambda}_{i,k}^n \right\rangle / \left\langle \Delta\tilde{\boldsymbol{x}}_{i,k}^n, \Delta\tilde{\boldsymbol{x}}_{i,k}^n \right\rangle$, with $\Delta G_i^* = \Delta\tilde{\boldsymbol{x}}_{i,k}^n = -\tilde{\boldsymbol{x}}_{i,k}^n + \tilde{\boldsymbol{x}}_i^{n_l}$, and $\Delta\boldsymbol{\lambda}_{i,k}^n = \boldsymbol{\lambda}_{i,k}^n - \boldsymbol{\lambda}_{i,k}^{n_l}$. It is worthy noting that $\hat{\alpha}_i^n$ and $\hat{\beta}_i^n$ are computed from intermediate quantities of different agents in different D-PDDP iterations, indicating that this update rule is especially useful for different agents in a distributed problem.

C. Adaptive Penalty Parameter Update

It should be pointed out that linear approximation of the gradients ∂H_i^*, ∂G_i^* might be inaccurate, leading to invalid step sizes. Thus, it is necessary to evaluate the accuracy of the curvature estimations in order to guarantee the safety of the updated step size. To test the quality, the cosine of the angle between the variation of the dual sub-gradient as well as the variation of the dual variables are computed as

$$\begin{aligned} \alpha_i^{n,\mathrm{cor}} &= \frac{\left\langle \Delta\boldsymbol{x}_{i,k}^n, \Delta\hat{\boldsymbol{\lambda}}_{i,k}^n \right\rangle}{\left\| \Delta\boldsymbol{x}_{i,k}^n \right\| \left\| \Delta\hat{\boldsymbol{\lambda}}_{i,k}^n \right\|} \\ \beta_i^{n,\mathrm{cor}} &= \frac{\left\langle \Delta\tilde{\boldsymbol{x}}_{i,k}^n, \Delta\boldsymbol{\lambda}_{i,k}^n \right\rangle}{\left\| \Delta\tilde{\boldsymbol{x}}_{i,k}^n \right\| \left\| \Delta\boldsymbol{\lambda}_{i,k}^n \right\|} \end{aligned} \tag{69}$$

The spectral stepsizes are only allowed to be updated if the above assessment is lower than a pre-defined threshold. The specific update rule can be expressed as

$$\rho_i^{n+1} = \begin{cases} \sqrt{\hat{\alpha}_i^n \hat{\beta}_i^n} & \text{if } \alpha_i^{n,\mathrm{cor}} > \epsilon^{\mathrm{cor}} \text{ and } \beta_i^{n,\mathrm{cor}} > \epsilon^{\mathrm{cor}} \\ \hat{\alpha}_i^n & \text{if } \alpha_i^{n,\mathrm{cor}} > \epsilon^{\mathrm{cor}} \text{ and } \beta_i^{n,\mathrm{cor}} \leq \epsilon^{\mathrm{cor}} \\ \hat{\beta}_i^n & \text{if } \alpha_i^{n,\mathrm{cor}} \leq \epsilon^{\mathrm{cor}} \text{ and } \beta_i^{n,\mathrm{cor}} > \epsilon^{\mathrm{cor}} \\ \rho_i^n & \text{otherwise,} \end{cases} \tag{70}$$

in which ϵ^{cor} is curvature estimations' accuracy threshold. Note that ρ_i^{n+1} remains equal to ρ_i^n from the previous computation when both curvature estimations are determined to be invalid. In addition, it is necessary to limit the above parameter update to ensure the numerical stability of the acceleration algorithm, where the specific limiting rules are as follows [46]:

$$\rho_i^{n+1} = \max\left\{ \min\left\{ \rho_i^{n+1}, \left(1 + \frac{C_{\mathrm{cg}}}{n^2}\right) \rho_i^n \right\}, \frac{\rho_i^n}{1 + C_{\mathrm{cg}}/n^2} \right\} \tag{71}$$

where C_{cg} is a large convergence constant.

We summarize the pseudocode of the necessary steps for adaptive penalty parameters in Algorithm 3. Although only the derivation with respect to $(\boldsymbol{x}_{i,k}^n, \tilde{\boldsymbol{x}}_{i,k}^n, \boldsymbol{\lambda}_{i,k}^n)$ is presented above, there are similar derivation procedures for $(\boldsymbol{u}_{i,k}^n, \tilde{\boldsymbol{u}}_{i,k}^n, \boldsymbol{\zeta}_{i,k}^n)$, $(\tilde{\boldsymbol{x}}_{i,k}^{\mathrm{a,n}}, \boldsymbol{z}_{i,k}^n, \boldsymbol{y}_{i,k}^n)$, $(t_{i,N}^n, \tilde{t}_{i,N}^n, \nu_{i,N}^n)$, $(\tilde{\boldsymbol{t}}_{i,N}^{\mathrm{a,n}}, \boldsymbol{s}_{i,N}^n, \boldsymbol{\eta}_{i,N}^n)$, of which the procedure for calculating the

adaptive penalty parameters are also listed in Algorithm 3. This algorithm adds the content of the adaptive penalty parameter calculation without affecting Algorithm 2. Algorithm 3 shows that the results obtained in the intermediate steps of the D-PDDP algorithm can be used to obtain the adaptive penalty parameters through the spectral gradient method, which helps to reduce the number of iterations of the algorithm and accelerates the algorithm's efficiency. In order to increase the reliability of the algorithm, it is recommended to execute Algorithm 3 every C_{Freq} iteration steps instead of executing it at every step.

Algorithm 3 Adaptive Penalty Parameter Scheme for D-PDDP

1: Initialize: $\tau_i^0, \rho_i^0, \gamma_i^0, \mu_i^0, \sigma_i^0, n_l = 0$.
2: **while** Not Converge **do**
3: Update D-PDDP using lines 5-11 in Algorithm 2.
4: Update dual variables using lines 12 in Algorithm 2 with adaptive penalty parameters $\tau_i^n, \rho_i^n, \mu_i^n, \gamma_i^n, \sigma_i^n$.
5: **if** mod $(n, C_{Freq}) == 1$ **then**
6: Locally update intermediate variable $\hat{\boldsymbol{\zeta}}_{i,k}^{n+1} = \boldsymbol{\zeta}_{i,k}^n + \tau_i^n \left(\boldsymbol{u}_{i,k}^{n+1} - \tilde{\boldsymbol{u}}_{i,k}^n\right)$
7: Locally update intermediate variable $\hat{\boldsymbol{\lambda}}_{i,k}^{n+1} = \boldsymbol{\lambda}_{i,k}^n + \rho_i^n \left(\boldsymbol{x}_{i,k}^{n+1} - \tilde{\boldsymbol{x}}_{i,k}^n\right)$
8: Locally update intermediate variable $\hat{\boldsymbol{y}}_{i,k}^{n+1} = \boldsymbol{y}_{i,k}^n + \mu_i^n \left(\boldsymbol{x}_{i,k}^{\mathrm{a},n+1} - \boldsymbol{z}_{i,k}^{\mathrm{a},n}\right)$
9: Locally update intermediate variable $\hat{\nu}_{i,N}^{n+1} = \nu_{i,N}^n + \sigma_i^n \left(t_{i,N}^{n+1} - \tilde{t}_{i,N}^n\right)$
10: Locally update intermediate variable $\hat{\boldsymbol{\eta}}_{i,N}^{n+1} = \boldsymbol{\eta}_{i,N}^n + \gamma_i^n \left(t_{i,N}^{\mathrm{a},n+1} - s_{i,N}^{\mathrm{a},n}\right)$
11: Locally compute spectral stepsizes and safely update ρ_i^n with (67)(68)(69)(70)
12: Locally compute spectral stepsizes and safely update $\tau_i^n, \mu_i^n, \gamma_i^n, \sigma_i^n$ with the same calculation steps as (67)(68)(69)(70)
13: **else**
14: $\tau_i^{n+1} = \tau_i^n$
15: $\rho_i^{n+1} = \rho_i^n$
16: $\mu_i^{n+1} = \mu_i^n$
17: $\gamma_i^{n+1} = \gamma_i^n$
18: $\sigma_i^{n+1} = \sigma_i^n$
19: **end if**
20: **end while**

V. Application Example: UAV Swarm Distributed Optimization

In this section, the effectiveness of the proposed methodology is validated through several numerical simulation studies using a fleet of UAV swarm. We first verify the effectiveness of the D-PDDP algorithm for distributed spatio-temporal trajectory optimization non-convex dynamics, obstacle constraints, inter-aircraft constraints, and terminal time constraints in four different scenarios. Then, we demonstrate the efficiency of the proposed adaptive penalty parameter updating approach by comparing it with other adaptive penalty rules.

A. Mission Scenario Setting

It is assumed that each UAV in the swarm is moving with a constant speed in a two-dimensional plan. The dynamics of the UAV is an ideal mass model, without considering its attitude dynamics. Under this assumption, the kinematics

model of each UAV can be described by

$$\begin{aligned} \dot{x}_i &= V\cos\theta_i \\ \dot{y}_i &= V\sin\theta_i \\ \dot{\theta}_i &= \omega_i \end{aligned} \tag{72}$$

where $[x_i, y_i, \theta_i]^T$ is the state variable of a single UAV, $[x_i, y_i]^T$ denotes the i-th UAV's coordinates and θ_i is the flight heading angle. The flight velocity of the UAV in the horizontal plane is set to a fixed value V. The UAV is controlled by the turning angular rate that adjusts the heading angle. Since the turning angular rate of the UAV is limited by physical constraints, we set the range of the turning rate ω_i to be $|\omega_i| \leq \omega_{\max}$, where $\omega_{\max}$ is a positive constant. Additionally, we establish $t_{\min} \leq t_N \leq t_{\max}$ as the limitations for each UAV's flying time.

We consider the mission that the swarm should approach the desired terminal states while respecting different constraints. To this end, a quadratic cost function $J_i(\boldsymbol{X}_i, \boldsymbol{U}_i)$ for all UAVs in the swarm is utilized as

$$\begin{aligned} J_i(\boldsymbol{X}_i, \boldsymbol{U}_i) = &\frac{1}{2}\left(\boldsymbol{x}_{i,N} - \boldsymbol{x}_{i,N}^{\mathrm{d}}\right)^{\mathrm{T}} \mathbf{W}_i^N \left(\boldsymbol{x}_{i,N} - \boldsymbol{x}_{i,N}^{\mathrm{d}}\right) \\ &+ \sum_{k=0}^{N-1}\left[\frac{1}{2}\boldsymbol{u}_{i,k}^{\mathrm{T}}\mathbf{R}_i\boldsymbol{u}_{i,k} + \frac{1}{2}(\boldsymbol{x}_{i,k} - \boldsymbol{x}_i^d)^{\mathrm{T}}\mathbf{W}_i^s(\boldsymbol{x}_{i,k} - \boldsymbol{x}_i^d)\right] \end{aligned} \tag{73}$$

where $\boldsymbol{x}_{i,N}^d = \left[x_i^d, y_i^d, \theta_i^d\right]^T$ denotes the desired terminal state vector of the i-th UAV . $\mathbf{W}_i^N$, $\mathbf{R}_i$ and $\mathbf{W}_i^s$ are weighting diagonal matrices of the cost function.

(1) *Path constraints*: The obstacles present in the environment during flight are considered as a circular no-fly zone, i.e.,

$$(x_{i,k} - x_o)^2 + (y_{i,k} - y_o)^2 \geq (r_o + d_o)^2 \tag{74}$$

where $[x_o, y_o]^T$ denotes the center coordinates of the obstacle region, r_o is the radius, the smallest safe distance that every agent needs to keep with the obstacle $o \in O$ is denoted as d_o, and the collection of obstacles in the environment are represented as O.

(2) *Inter-agent constraints*: Each UAV within the swarm needs to maintain a safe distance from its neighbors to avoid collisions, and it also should keep within a certain distance from its neighbors to maintain the functionality of the communication link due to physical limitations of the communication devices. Hence, these constraints can be described as

$$(x_{i,k} - x_{j,k})^2 + (y_{i,k} - y_{j,k})^2 \geq d_{\mathrm{col}}^2 \tag{75a}$$

$$(x_{i,k} - x_{j,k})^2 + (y_{i,k} - y_{j,k})^2 \leq d_{\mathrm{con}}^2 \tag{75b}$$

where d_{col} denotes the safe distance and d_{con} denotes maximum effective communication distance between two UAVs.

(3) *Terminal time sequence constraints*: We consider the UAVs arrive at the targets sequentially with a specified time interval. This constraint can be mathematically formulated as

$$\mathbf{A}_i \tilde{\boldsymbol{t}}_i^{\mathbf{a}} = \boldsymbol{t}_{\delta_i} \tag{76}$$

where $\boldsymbol{t}_{\delta_i}$ is the time interval vector and the matrix $\mathbf{A}_i$ represents the time series relationship between agent i and its neighbors as

$$\mathbf{A}_i = \begin{bmatrix} 1 & -1 & \cdots & 0 \\ 1 & \vdots & \ddots & \vdots \\ \vdots & 0 & \cdots & -1 \\ 1 & 0 & \cdots & 0 \end{bmatrix} \tag{77}$$

where the j-th row of $\mathbf{A}_i$ represents the temporal relationship between UAV i and its neighbor j, corresponding to the j-th row of $\boldsymbol{t}_{\delta_i}$ denoting the pre-defined time interval between i and j. Notice that when $\boldsymbol{t}_{\delta_i}$ is a zero vector, it means that the time interval is zero and all UAVs arrive at the targets simultaneously. When $\boldsymbol{t}_{\delta_i}$ is a non-zero vector, each UAV then follows the time interval of the elements in $\boldsymbol{t}_{\delta_i}$. In the implementation of the D-PDDP algorithm, we use the relaxation form of constraint (76) for better convergence as

$$|\mathbf{A}_i \tilde{\boldsymbol{t}}_{i,N}^{\mathbf{a}} - \boldsymbol{t}_{\delta_i}| \leq \boldsymbol{\epsilon}_{t_\delta} \tag{78}$$

where $\boldsymbol{\epsilon}_{t_\delta}$ is a constant relaxation parameter vector.

The conditions on the parameters of the algorithmic cost function are summarized in Table 1.

Table 1 Initial conditions and aerodynamics.

Parameters	Value
Terminal weighting matrix $\mathbf{W}_i^N$	$\text{diag}(25\mathbf{I}_3)$
Control weighting matrix $\mathbf{R}_i$	1
State weighting matrix $\mathbf{W}_i^s$	$\mathbf{0}$
Penalty τ^0 of $\boldsymbol{U}_i$	0.2
Penalty ρ^0 of $\boldsymbol{X}_i$	2
Penalty μ^0 of $\tilde{\boldsymbol{X}}_i^{\mathrm{a}}$	1
Penalty σ^0 of $\boldsymbol{T}_i$	2
Penalty γ^0 of $\tilde{\boldsymbol{T}}_i^{\mathrm{a}}$	1
Damping coefficient of PDDP α_l	0.4
Stopping threshold ϵ_{rel}, ϵ_{abs}	6×10^{-2}, 10^{-3}

Remark 7 *Nonlinear state constraints* (74) *and* (75) *can be mathematically expressed in a vector form* $\left\|\boldsymbol{p}_{i,k} - \boldsymbol{p}_o\right\|_2 \leq r_o$,

where $\boldsymbol{p}_{i,k} = \left[x_i^d, y_i^d\right]^T$ *represents the position of the ith UAV and* $\boldsymbol{p}_o (o \in O)$ *denotes the center coordinate of the region. This constraint is non-convex and needs to be solved with the necessary convexification before using the optimization toolbox. The constraints in this paper are convexified using a linear approximation around the nominal trajectory for the sake of simplicity*

$$\left\|\bar{\boldsymbol{p}}_{i,k} - \boldsymbol{p}_o\right\|_2 + \vartheta^{\mathrm{T}}\left[\left(\boldsymbol{p}_{i,k} - \boldsymbol{p}_o\right) - \left(\bar{\boldsymbol{p}}_{i,k} - \boldsymbol{p}_o\right)\right] \geq r_o \tag{79}$$

where

$$\vartheta = \frac{\bar{\boldsymbol{p}}_{i,k} - \boldsymbol{p}_o}{\left\|\bar{\boldsymbol{p}}_{i,k} - \boldsymbol{p}_o\right\|_2} \tag{80}$$

where $\bar{\boldsymbol{p}}_{i,k}$ *is the position of vehicle i corresponding to the nominal trajectory.*

B. Characteristics of Proposed Algorithm

In this subsection, we evaluate the distributed spatial-temporal optimization capability of the proposed D-PDDP algorithm for UAV swarms in four different scenarios through numerical simulations. In all simulations, we use the normal version of D-PDDP, which has a constant penalty parameter for all agents without any parameter adaptation. The initial guess of the D-PDDP algorithm uses the trajectory optimized by the DDP algorithm, which not considering obstacles, neighbors and free terminal time, and only considering each UAV itself. The constraints considered in the four scenarios include path constraints with obstacles, and inter-intelligence constraints with collision avoidance and communication maintenance, as well as terminal time-series constraints that satisfy certain time intervals. In order to quickly distinguish difference between the scenarios, the constraints in the 4 different scenarios are summarized in Table 2. The Scenario 2 and 3 mainly aim to illustrate that the proposed D-PDDP algorithm is able to compute the UAV swarm trajectory optimization problem that considering temporal constraints with time interval through different scenario settings. The Scenario 4 are mainly used to illustrate that the proposed D-PDDP algorithm is able to solve trajectory optimization problems in larger scale (tens of them) UAV swarm problems.

Table 2 The constraints for the 4 scenarios.

Constraints	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Obstacles	√	√	√	√
Collision avoidance	√	√	√	√
Communication maintenance	√	√	√	√
Terminal time-series	×	√	√	×

Scenario 1: We consider four UAVs in Scenario 1, where each two of them flies towards the other two UAVs. There is a circular obstacle in the environment from which all UAVs need to maintain a safe distance from it. In addition, the flight times of all UAVs are free and not subject to constraint (76), which means that each UAV can optimize its own

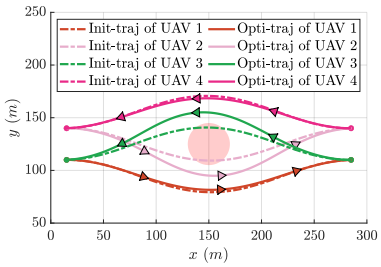
flight time without satisfying specific time sequence requirements with its neighbors (although each UAV still needs to share time variable with its neighbors and reach consensus). The four swarm UAVs in the scenario are neighbors of each other, which means that the constraint (75a) is naturally satisfied in this scenario, i.e., each UAV needs to communicate with all other UAVs. Due to the need for communication, the UAV need to stay within communication distance from the other UAVs. The communication distance is set slightly larger than the distance between the UAV and its farthest neighbor. Table 3 summarizes the specific state of each UAV, the positions and radius of the obstacle, the inter-agent distance, and the initialized values and bounds of the flight time. Notice that flight time here is the total flight duration of the vehicle, and its value changes constantly with iterations of trajectory optimization. In contrast, the initial guess of flight time is the empirical flight time value before starting trajectory optimization, which is used only once.

Table 3 Initial conditions of Scenario 1.

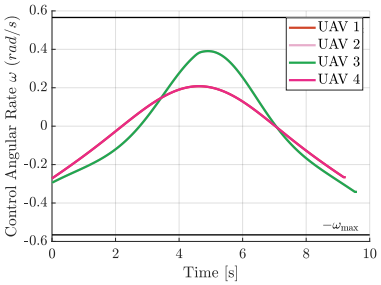
Parameters	Value
Initial state of UAV 1 $\left[x_{1_0}, y_{1_0}, \theta_{1_0}\right]^{\mathrm{T}}$	$[15.0m, 110.0m, 0°]^{\mathrm{T}}$
Initial state of UAV 2 $\left[x_{2_0}, y_{2_0}, \theta_{2_0}\right]^{\mathrm{T}}$	$[15.0m, 140.0m, 0°]^{\mathrm{T}}$
Initial state of UAV 3 $\left[x_{3_0}, y_{3_0}, \theta_{3_0}\right]^{\mathrm{T}}$	$[285.0m, 110.0m, 180°]^{\mathrm{T}}$
Initial state of UAV 4 $\left[x_{4_0}, y_{4_0}, \theta_{4_0}\right]^{\mathrm{T}}$	$[285.0m, 140.0m, 180°]^{\mathrm{T}}$
Target state of UAV 1 $\left[x_{1_f}, y_{1_f}, \theta_{1_f}\right]^{\mathrm{T}}$	$[285.0m, 110.0m, 0°]^{\mathrm{T}}$
Target state of UAV 2 $\left[x_{2_f}, y_{2_f}, \theta_{2_f}\right]^{\mathrm{T}}$	$[285.0m, 140.0m, 0°]^{\mathrm{T}}$
Target state of UAV 3 $\left[x_{3_f}, y_{3_f}, \theta_{3_f}\right]^{\mathrm{T}}$	$[15.0m, 110.0m, 180°]^{\mathrm{T}}$
Target state of UAV 4 $\left[x_{4_f}, y_{4_f}, \theta_{4_f}\right]^{\mathrm{T}}$	$[15.0m, 140.0m, 180°]^{\mathrm{T}}$
Velocity of each UAV V_i	$30m/s$
UAV turning maneuverability $\omega_{\max}$	$0.5768rad/s$
Initial guess of flight time of each UAV $t_{i,N}$	$9.3s$
Lower and upper bound of flight time $[t_{\min}, t_{\max}]$	$[0.1s, 20s]$
Safe distance to obstacle d_o and neighbors d_{col}	$10m, 10m$
Communication maintenance distance with neighbors d_{col}	$300m$
Center and Radius of circular obstacle p_o, r_o	$[150.0m, 125.0m]^{\mathrm{T}}, 20.0m$

The optimization results of the four UAVs in Scenario 1 are shown in Fig. 2. In Fig. 2(a), the dashed lines are the initial guesses of the flight trajectories and the solid lines are the optimized results, from which we can see that the proposed D-PDDP method can optimize from the initialized trajectory that violates the obstacle constraints to obstacle avoidance. The trajectory results show that the optimized trajectories are able to avoid obstacles in the environment beyond the safe distance while the trajectory of each UAV maintains a safe distance from its neighbors. The control history of the four UAVs is shown in Fig. 2(b). Since the constraint (76) is not required to be satisfied in this scenario, the four UAVs optimize their own flight times from $9.3s$ to $9.24s$ and $9.60s$, as can be seen from the change in trajectories from the initial guess to the optimized results in Fig. 2(a) and the convergence of the UAV flight times with the iterations of the algorithm in Fig. 2(c). Fig. 2(d) - Fig. 2(f) show the snapshots of the UAV flying according to the optimized

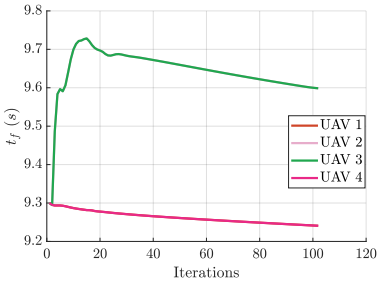
trajectory after the distributed optimization. The circles above and below the obstacle in the figure that match the color of the trajectory represent the safe distance range of the UAVs, where Fig. 2(e) is drawn at the closest point ($13.77m$) of the flight paths of the two UAVs. The dashed circle indicates the range of the communication distance of the UAV. Since the four UAVs are able to communicate with each other, all four UAVs are within each other's maximum communication distance circle, as shown in the figures. In summary, the results demonstrate that the proposed D-PDDP provides the ability to optimize the flight trajectories of multiple UAVs to avoid obstacles and satisfy inter-aircraft avoidance constraints while optimizing their own flight time.



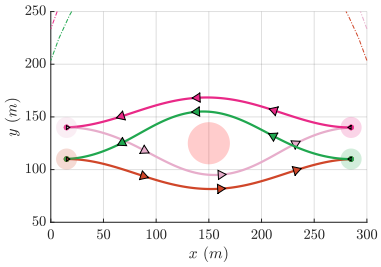
(a) The initial guess trajectory (transparent line) and optimized flight trajectory (solid line) of each UAV.



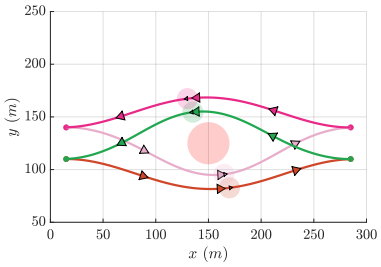
(b) Control angular velocity variation curve of each UAV.



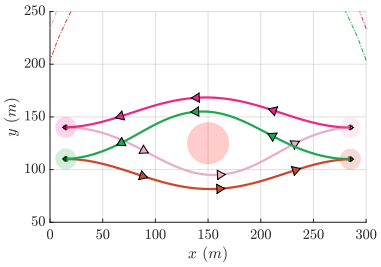
(c) The variation curve of the flight time for each UAV with the number of iterations.



(d) The snapshot of the scene at the initial moment of flight t_0.



(e) The snapshot of the scene at the moment of minimum distance between the vehicles.



(f) The snapshot of the scene at the terminal moment of flight t_f.

Fig. 2 Distributed spatial-temporal joint optimization results for Scenario 1.

Scenario 2: A five-UAV consecutive interception scenario is considered to validate the optimization capability of the D-PDDP algorithm for time-series tasks. The UAV swarm intercepts different targets in the same region from different orientations with the requirement of position and flight path angle constraints. The 5 swarm UAVs need to communicate with their neighbors and maintain a safe distance while flying to the target point. The neighborhood size of each UAV is set to $|\mathcal{N}_i| = 3$. In addition, the UAVs need to intercept the targets at certain time intervals, i.e., they satisfy the time-series constraints (78). The initial time for all the UAVs is set to $9.0s$, the impact times of all UAVs satisfies equal interval $\boldsymbol{t}_{\delta_i} = 0.1s$ constraints, and the relaxation variable in Eq. (78) is set to $0.01s$ for all UAV. To ensure higher accuracy of the time sequence constraints, we tune the stopping condition as $\epsilon_{\text{abs}} = 5 \times 10^{-4}$ in this scenario. In addition, the communication distance is set based on the initial position of the UAV using the closest distance criterion. Table 4

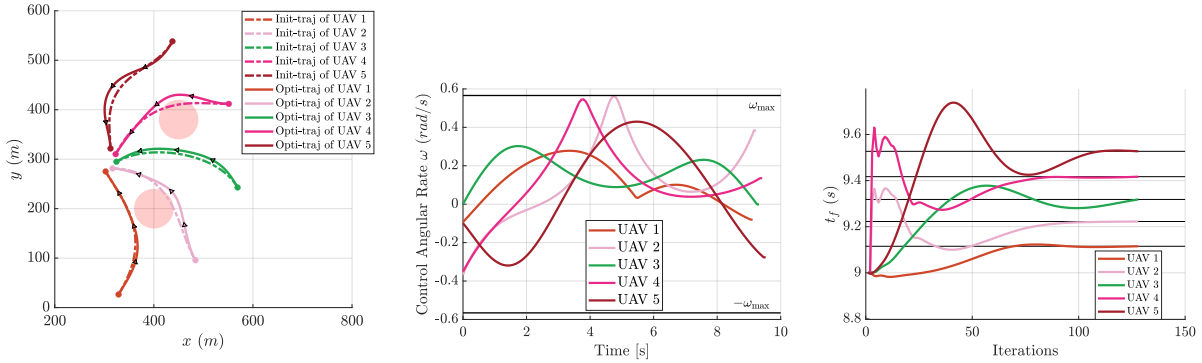
summarizes the relevant settings for Scenario 2, where the same parameters as in Scenario 1 are not repeated.

Table 4 Initial conditions of Scenario 2.

Parameters	Value
Initial state of UAV 1 $\left[x_{1_0}, y_{1_0}, \theta_{1_0}\right]^{\mathrm{T}}$	$[328.75m, 26.5m, 60°]^{\mathrm{T}}$
Initial state of UAV 2 $\left[x_{2_0}, y_{2_0}, \theta_{2_0}\right]^{\mathrm{T}}$	$[484.0m, 95.6m, 120°]^{\mathrm{T}}$
Initial state of UAV 3 $\left[x_{3_0}, y_{3_0}, \theta_{3_0}\right]^{\mathrm{T}}$	$[587.0m, 242.8m, 120°]^{\mathrm{T}}$
Initial state of UAV 4 $\left[x_{4_0}, y_{4_0}, \theta_{4_0}\right]^{\mathrm{T}}$	$[551.2m, 411.85m, 180°]^{\mathrm{T}}$
Initial state of UAV 5 $\left[x_{4_0}, y_{4_0}, \theta_{4_0}\right]^{\mathrm{T}}$	$[437.5m, 538.16m, 240°]^{\mathrm{T}}$
Target state of UAV 1 $\left[x_{1_f}, y_{1_f}, \theta_{1_f}\right]^{\mathrm{T}}$	$[302.6m, 275.14m, 120°]^{\mathrm{T}}$
Target state of UAV 2 $\left[x_{2_f}, y_{2_f}, \theta_{2_f}\right]^{\mathrm{T}}$	$[324.45m, 281.42m, 180°]^{\mathrm{T}}$
Target state of UAV 3 $\left[x_{3_f}, y_{3_f}, \theta_{3_f}\right]^{\mathrm{T}}$	$[342.45m, 294.8m, 210°]^{\mathrm{T}}$
Target state of UAV 4 $\left[x_{4_f}, y_{4_f}, \theta_{4_f}\right]^{\mathrm{T}}$	$[322.84m, 310.17m, 240°]^{\mathrm{T}}$
Target state of UAV 5 $\left[x_{4_f}, y_{4_f}, \theta_{4_f}\right]^{\mathrm{T}}$	$[312.5m, 321.65m, 270°]^{\mathrm{T}}$
Initial guess of flight time of each UAV $t_{i,N}$	$9.0s$
Communication maintenance distance with neighbors d_{col}	$380m$
Center of circular obstacle $\boldsymbol{p}_{o_1}$	$[400.0m, 200.0m]^{\mathrm{T}}$
Center of circular obstacle $\boldsymbol{p}_{o_2}$	$[450.0m, 380.0m]^{\mathrm{T}}$
Radius of circular obstacle r_o	$40.0m$

The optimization results of the UAV swarm in Scenario 2 are shown in Fig. 3, which contains the optimized trajectories, control history, time changes, and snapshots of the flights of the UAV swarm. From Fig. 3(a), it can be observed that the trajectories of UAV 2 and UAV 4 change from constraint-violating trajectories to obstacle-avoiding trajectories, and the trajectory of UAV 5 is more curved compared to the initial guess to satisfy the time sequence constraints. Similarly, it can be seen in Fig. 3(d) through Fig. 3(f) that all four UAVs are within each other's maximum communication distance circle and collision free, which means the UAVs satisfy the collision avoidance constraints and the communication maintenance constraints. On the other hand, it can be seen from Fig. 3(c) that the flight times of the five UAVs are optimized from the same initial guess $9.0s$ to $9.12s$, $9.22s$, $9.32s$, $9.42s$, $9.53s$, respectively, which satisfy the consecutive interception constraint with time interval being $0.1s$ between two UAVs. The results illustrate that the D-PDDP algorithm is capable of accomplishing the UAV swarm trajectory optimization task with time sequence constraints through distributed spatial-temporal joint optimization.

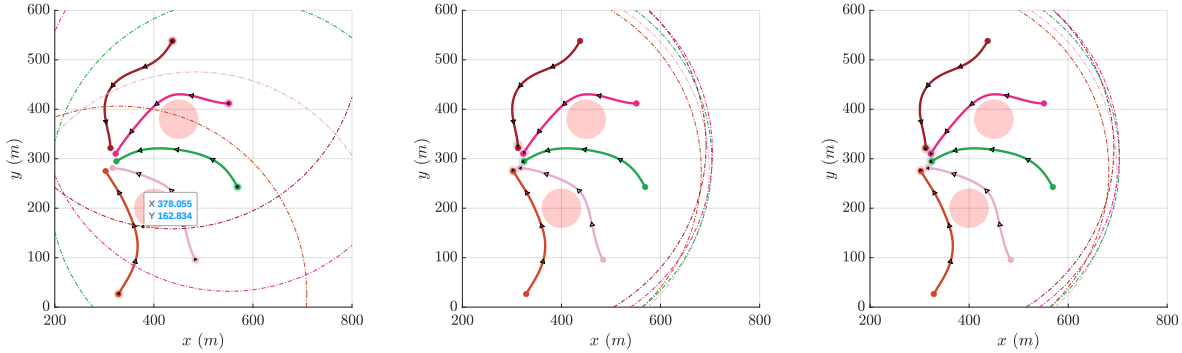
Scenario 3: We consider 16 UAVs evenly distributed in a circular formation to cross an obstacle at the center of the formation while flying to another point symmetric to the center of the circle in this scenario. The 16 swarm UAVs need to communicate with their neighbors and maintain a safe distance while flying to the target point. The neighborhood size of each UAV is set to $|\mathcal{N}_i| = 5$. In addition, all UAVs are required to complete flight missions simultaneously, i.e., they need to satisfy the constraint (76), where the time interval vector is $\mathbf{0}$. This means that each UAV needs to optimize its own flight time while keeping the same time as its neighbors. Table 6 summarizes the relevant settings for Scenario 3.



(a) The initial guess trajectory (transparent line) and optimized flight trajectory (solid line) of each UAV.

(b) Control angular velocity variation curve of each UAV.

(c) The variation curve of the flight time for each UAV with the number of iterations.



(d) The snapshot of the scene at the initial moment of flight t_0.

(e) The snapshot of the scene at the moment of minimum distance between the vehicles.

(f) The snapshot of the scene at the terminal moment of flight t_f.

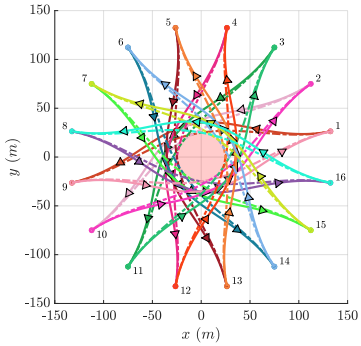
Fig. 3 Distributed spatial-temporal joint optimization results for Scenario 2.

The optimization results of the UAV swarm in Scenario 3 are shown in Fig. 4, which contains the optimized trajectories, control histories, time changes, and snapshots of the flights of the swarm UAVs. The control history and flight time optimization of the swarm UAVs have the same result due to the axisymmetric nature of each UAV. As shown in Fig. 4, the swarm UAVs can reach their respective target locations safely while maintaining communication and not conflicting with their neighbors as well as obstacles. Specifically, the closest distance between the UAVs are $13.8m$, and the communication maintenance distance $120m$ is not violated in this scenario. The flight time of the swarm UAVs is optimized from the initial $9.2s$ to $9.36s$, showing that the algorithm can meet the distributed time optimization

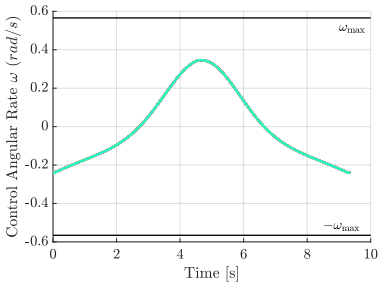
Table 5 Initial conditions of Scenario 3.

Parameters	Value
Center and radius of circular formation $\boldsymbol{p}_{\text{form}}, r_{\text{form}}$	$[0.0m, 0.0m]^{\text{T}}, 270.0m$
Initial guess of flight time of each UAV $t_{i,N}$	$9.2s$
Communication maintenance distance with neighbors d_{con}	$120m$
Center and radius of circular obstacle $\boldsymbol{p}_o, r_o$	$[0.0m, 0.0m]^{\text{T}}, 20.0m$

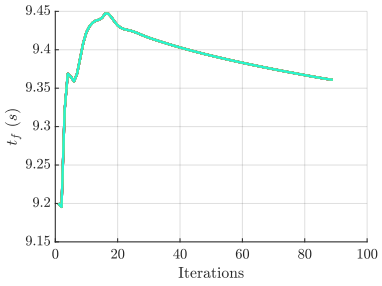
requirements of the scenario. Notice that all UAVs in this scenario are placed evenly with distributed in a circular formation to cross an obstacle at the center of the formation while flying to another point symmetric to the center of the circle, the relative geometry between each UAV and its destination is the same across all UAVs. This, thus, generates the same command and mission time history, as shown in Figs. 4(b) and 4(c).



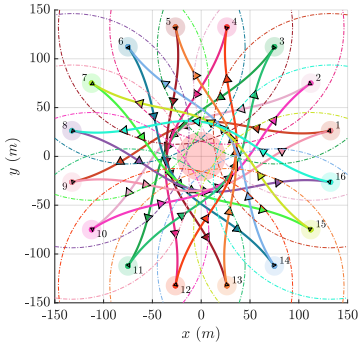
(a) The initial guess trajectory (transparent line) and optimized flight trajectory (solid line) of each UAV.



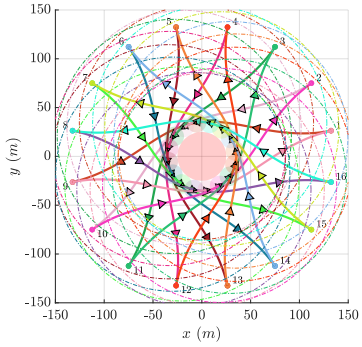
(b) Control angular velocity variation curve of each UAV.



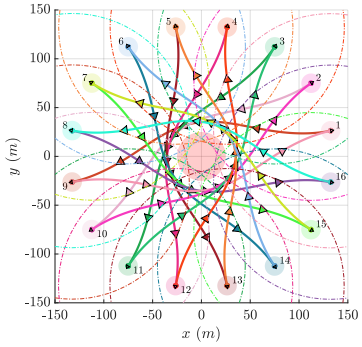
(c) The variation curve of the flight time for each UAV with the number of iterations.



(d) The snapshot of the scene at the initial moment of flight t_0.



(e) The snapshot of the scene at the moment of minimum distance between the vehicles.



(f) The snapshot of the scene at the terminal moment of flight t_f.

Fig. 4 Distributed spatial-temporal joint optimization results for Scenario 3.

Scenario 4: We consider 20 UAVs sequentially and uniformly distributed, flying through several random obstacles to the other side of the scene. During the flight to the target point, each swarm UAV needs to communicate with the UAVs with a neighborhood size of $|\mathcal{N}_i| = 5$ and maintain a safe distance. The length of each UAV's flight path may be different due to the complex environment, and hence each UAV optimizes its own flight time individually without requiring to satisfy the constraint (78). Table 6 summarizes the specific state of each UAV, the positions and radius of the obstacle, the inter-agent distance, and the initialized values and bounds of the flight time.

The optimization results of the UAV swarm in Scenario 4 are shown in Fig. 5, which contains the optimized trajectories, control histories, time changes, and snapshots of the flights of the UAV swarm. As shown in Fig. 5(d) to Fig. 5(f), we can see that all 20 UAVs can reach a consensus with their neighbors and complete the trajectory

Table 6 Initial conditions of Scenario 4.

Parameters	Value
Initial state of UAV i $\left[x_{i_0}, y_{i_0}, \theta_{i_0}\right]^{\mathrm{T}}$	$\left[(20+10\times(-1)^i)m, (100+(i-1)\times 30)m, 0^\circ\right]^{\mathrm{T}}$
Target state of UAV i $\left[x_{i_f}, y_{i_f}, \theta_{i_f}\right]^{\mathrm{T}}$	$\left[(290+10\times(-1)^i)m, (100+(i-1)\times 30)m, 0^\circ\right]^{\mathrm{T}}$
Initial guess of flight time of each UAV $t_{i,N}$	$9.2s$
Center and Radius of circular obstacle $\boldsymbol{p}_{o_1}, r_{o_1}$	$[200.0m, 200.0m]^{\mathrm{T}}, 15.0m$
Center and Radius of circular obstacle $\boldsymbol{p}_{o_1}, r_{o_2}$	$[150.0m, 120.0m]^{\mathrm{T}}, 20.0m$
Center and Radius of circular obstacle $\boldsymbol{p}_{o_1}, r_{o_3}$	$[180.0m, 300.0m]^{\mathrm{T}}, 20.0m$
Center and Radius of circular obstacle $\boldsymbol{p}_{o_1}, r_{o_4}$	$[150.0m, 390.0m]^{\mathrm{T}}, 20.0m$
Center and Radius of circular obstacle $\boldsymbol{p}_{o_1}, r_{o_5}$	$[210.0m, 500.0m]^{\mathrm{T}}, 20.0m$
Center and Radius of circular obstacle $\boldsymbol{p}_{o_1}, r_{o_6}$	$[150.0m, 580.0m]^{\mathrm{T}}, 15.0m$
Center and Radius of circular obstacle $\boldsymbol{p}_{o_1}, r_{o_7}$	$[180.0m, 680.0m]^{\mathrm{T}}, 20.0m$
Communication maintenance distance with neighbors d_{con}	$170m$
Safe distance to obstacle d_o and neighbors d_{col}	$10m, 10m$

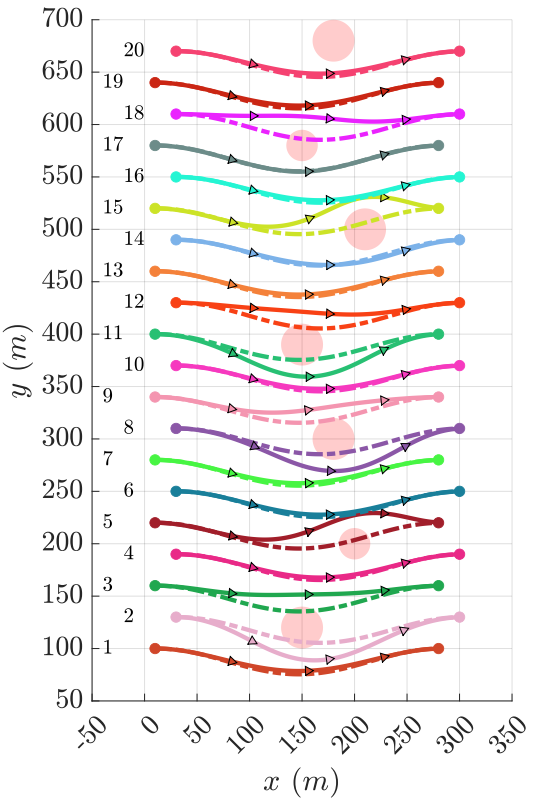
optimization of their own trajectories against obstacles and other UAVs' avoidance, as well as keeps each other within the communication distance. From Fig. 3(c) we can also see that each UAV is able to adjust its flight time according to its flight path length.

C. Comparison with Differential Acceleration Schemes

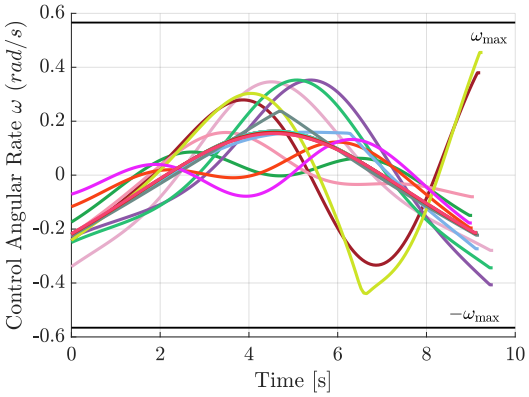
In this section, the fixed penalty parameters in the previous section will be regarded as tuning parameters and different acceleration schemes are compared in order to verify the superior acceleration performance of the proposed spectral gradient adaptive parameter (AP) scheme for the D-PDDP algorithm. The proposed acceleration scheme is compared with the original fixed penalty parameter scheme, the residual balancing (RB) scheme [31] and the Nesterov acceleration (NA) scheme [39] in the four aforementioned scenarios to demonstrate superior performance in terms of acceleration. To ensure that the comparisons are fair, all scenarios and simulation-related settings are the same as in Sec. V.B, except that the penalty parameter is shifted from fixed to varying. In addition, the implementation of the RB scheme is summarized in Eq. (81), same as in [31]. The penalty parameter is dynamically adjusted based on the residuals ratio to enhance convergence efficiency.

$$\rho_i^{n+1} = \begin{cases} \rho_i^n/2 & \text{if } \left\|\boldsymbol{r}_{(\cdot)}^{\mathrm{p},n}\right\|_2 \geq 10\left\|\boldsymbol{r}_{(\cdot)}^{\mathrm{d},n}\right\|_2 \\ 2\rho_i^n & \text{if } \left\|\boldsymbol{r}_{(\cdot)}^{\mathrm{d},n}\right\|_2 \geq 10\left\|\boldsymbol{r}_{(\cdot)}^{\mathrm{p},n}\right\|_2 \end{cases} \tag{81}$$

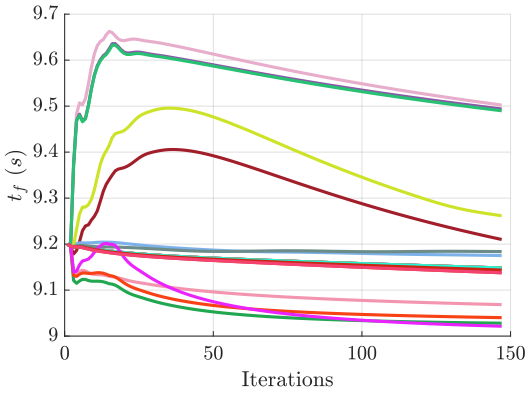
The NA parameter α_{Nes_1} is set to 1 and the tuning parameter η_{Nes_1} is set to 0.9. The relevant parameters for the adaptive parameter penalty are chosen as the safeguarding threshold $\epsilon_{\mathrm{cor}} = 0.5$ and the convergence constant $C_{\mathrm{cg}} = 5\times 10^2$. The penalty parameter is updated every 10 iteration steps in all acceleration schemes, i.e., C_{Freq} in



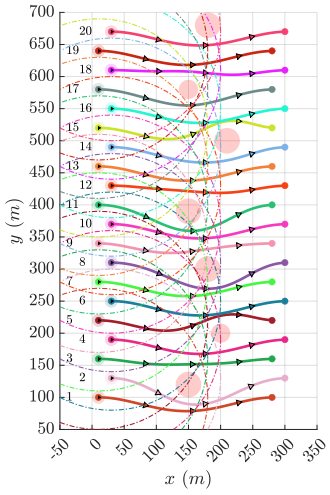
(a) The initial guess trajectory (transparent line) and optimized flight trajectory (solid line) of each UAV.



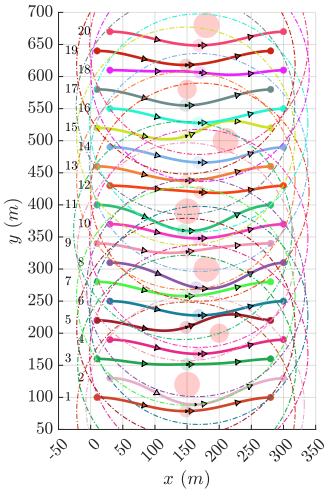
(b) Control angular velocity variation curve of each UAV.



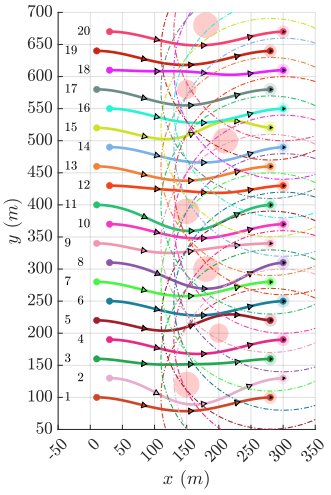
(c) The variation curve of the flight time for each UAV with the number of iterations.



(d) The snapshot of the scene at the initial moment of flight t_0.



(e) The snapshot of the scene at the moment of minimum distance between the vehicles.



(f) The snapshot of the scene at the terminal moment of flight t_f.

Fig. 5 Distributed spatial-temporal joint optimization results for Scenario 4.

Algorithm 3 is set to 10. The comparison data of the three different schemes in the four scenarios are summarized in Table 7.

The different schemes are all able to generate spatial-temporal optimized trajectories that satisfy the individual constraints as well as the inter-UAV constraints in all four scenarios, for which the specific results are not repeated in this section. As can be seen from Table 7, the proposed AP and the RB schemes reduce the number of iterations and the computational time of the original D-PDDP algorithm in all four scenarios, while the NA scheme increases the iterations in Scenario 2 instead. In all simulations, the proposed approach reduces the number of iterations more substantially than the other two schemes. And in Scenario 1, 2 and 4, the proposed method significantly reduces the computational time over the other two schemes. In summary, the proposed acceleration scheme proves to be an effective acceleration scheme.

Table 7 Iterations of different acceleration schemes.

Schemes	Vanilla	Time	RB	Iter	Time	NA	Iter	Time	AP	Iter	Time
1	102	90.45s	64	37.25%	54.56s	70	31.37%	66.74s	48	**52.94%**	**40.39s**
2	128	132.51s	100	21.875%	103.59s	129	−0.78%	137.89s	95	**25.78%**	**100.22s**
3	89	219.60s	73	17.98%	179.69s	69	22.47%	**173.93s**	68	**23.60%**	175.50s
4	147	406.20s	71	51.70%	196.77s	130	11.56%	369.47s	68	**53.74%**	**189.27s**

VI. Conclusions

In this paper, a fully distributed spatial-temporal trajectory optimization method is proposed for large-scale UAV swarm missions. Besides, an adaptive penalty parameter based on the spectral gradient method is proposed to efficiently reduce the number of iterations of the proposed algorithm. Our algorithm is capable of generating safe (both inter and outer) trajectories that satisfy different terminal time constraints (including time sequential constraints, simultaneous arrival and optimal free terminal time). The results with different scenarios demonstrate the flexibility, convergence and optimization capability of the proposed approach.

Due to the distributed computing nature of the D-PDDP algorithm, when migrating the algorithm to a real UAV swarm, it is only necessary to deploy the algorithm executed by each UAV in this paper to run independently on each real UAV, while guaranteeing the optimality of the global task. However, since this paper does not consider the communication delays of UAV swarm in real world, the algorithms need to be further investigated before deploying them to real swarm.

References

[1] Julian, K. D., and Kochenderfer, M. J., "Distributed wildfire surveillance with autonomous aircraft using deep reinforcement learning," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 8, 2019, pp. 1768–1778. https://doi.org/10.2514/1.G004106.

[2] Kolar, P., "Coupling Consensus Based Tasks with Subsumption Architecture for UAS Swarm Based Intelligence Surveillance and Reconnaissance Operations," *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, 2020, pp. 1–11. https://doi.org/10.1109/DASC50938.2020.9256816.

[3] Scharre, P., "How swarming will change warfare," *Bulletin of the Atomic Scientists*, Vol. 74, No. 6, 2018, pp. 385–389. https://doi.org/10.1080/00963402.2018.1533209.

[4] Balhance, N., Weiss, M., and Shima, T., "Cooperative guidance law for intrasalvo tracking," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 6, 2017, pp. 1441–1456. https://doi.org/10.2514/1.G002250.

[5] Chai, R., Savvaris, A., and Tsourdos, A., "Violation Learning Differential Evolution-Based hp-Adaptive Pseudospectral Method for Trajectory Optimization of Space Maneuver Vehicle," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 53, No. 4, 2017, pp. 2031–2044. https://doi.org/10.1109/TAES.2017.2680698.

[6] Laad, D., Elango, P., and Mohan, R., "Fourier Pseudospectral Method for Trajectory Optimization with Stability Requirements," *Journal of Guidance, Control, and Dynamics*, Vol. 43, No. 11, 2020, pp. 2073–2090. https://doi.org/10.2514/1.G005038.

[7] Hong, H., Maity, A., and Holzapfel, F., "Free Final-Time Constrained Sequential Quadratic Programming–Based Flight Vehicle Guidance," *Journal of Guidance, Control, and Dynamics*, Vol. 44, No. 1, 2021, pp. 181–189. https://doi.org/10.2514/1.G004874.

[8] Li, J., Li, C., and Zhang, Y., "Entry Trajectory Optimization With Virtual Motion Camouflage Principle," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 56, No. 4, 2020, pp. 2527–2536. https://doi.org/10.1109/TAES.2019.2949897.

[9] Morgan, D., Chung, S.-J., and Hadaegh, F. Y., "Model Predictive Control of Swarms of Spacecraft Using Sequential Convex Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 6, 2014, pp. 1725–1740. https://doi.org/10.2514/1.G000218.

[10] Deligiannis, A., Amin, M., Lambotharan, S., and Fabrizio, G., "Optimum Sparse Subarray Design for Multitask Receivers," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 55, No. 2, 2019, pp. 939–950. https://doi.org/10.1109/TAES.2018.2867258.

[11] Sun, W., Pan, Y., Lim, J., Theodorou, E. A., and Tsiotras, P., "Min-Max Differential Dynamic Programming: Continuous and Discrete Time Formulations," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 12, 2018, pp. 2568–2580. https://doi.org/10.2514/1.G003516.

[12] Chai, R., Savvaris, A., Tsourdos, A., Chai, S., and Xia, Y., "A review of optimization techniques in spacecraft flight trajectory design," *Progress in aerospace sciences*, Vol. 109, 2019, p. 100543. https://doi.org/10.1016/j.paerosci.2019.05.003.

[13] Li, W., and Todorov, E., "Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems," *Proceedings of the First International Conference on Informatics in Control, Automation and Robotics - Volume 1: ICINCO,*, INSTICC, SciTePress, 2004, pp. 222–229. https://doi.org/10.5220/0001143902220229.

[14] Aziz, J. D., Scheeres, D. J., and Lantoine, G., "Hybrid differential dynamic programming in the circular restricted three-body problem," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 5, 2019, pp. 963–975. https://doi.org/10.2514/1.G003617.

[15] Ozaki, N., Campagnola, S., Funase, R., and Yam, C. H., "Stochastic differential dynamic programming with unscented transform for low-thrust trajectory design," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 2, 2018, pp. 377–387. https://doi.org/10.2514/1.G002367.

[16] He, S., Shin, H.-S., and Tsourdos, A., "Computational guidance using sparse Gauss-Hermite quadrature differential dynamic programming," *IFAC-PapersOnLine*, Vol. 52, No. 12, 2019, pp. 13–18. https://doi.org/10.1016/j.ifacol.2019.11.062.

[17] Zhang, G., Wen, C., Han, H., and Qiao, D., "Aerocapture Trajectory Planning Using Hierarchical Differential Dynamic Programming," *Journal of Spacecraft and Rockets*, Vol. 59, No. 5, 2022, pp. 1647–1659. https://doi.org/10.2514/1.A35264.

[18] MAYNE, D., "A Second-order Gradient Method for Determining Optimal Trajectories of Non-linear Discrete-time Systems," *International Journal of Control*, Vol. 3, No. 1, 1966, pp. 85–95. https://doi.org/10.1080/00207176608921369.

[19] Pavlov, A., Shames, I., and Manzie, C., "Interior point differential dynamic programming," *IEEE Transactions on Control Systems Technology*, Vol. 29, No. 6, 2021, pp. 2720–2727. https://doi.org/10.1109/TCST.2021.3049416.

[20] Manchester, Z., and Kuindersma, S., "Derivative-free trajectory optimization with unscented dynamic programming," *2016 IEEE 55th Conference on Decision and Control (CDC)*, IEEE, 2016, pp. 3642–3647. https://doi.org/10.1109/CDC.2016.7798817.

[21] Shorinwa, O., Halsted, T., Yu, J., and Schwager, M., "Distributed Optimization Methods for Multi-Robot Systems: Part I–A Tutorial," *arXiv preprint*, 2023. https://doi.org/10.48550/arXiv.2301.11313.

[22] Nedić, A., and Olshevsky, A., "Distributed Optimization Over Time-Varying Directed Graphs," *IEEE Transactions on Automatic Control*, Vol. 60, No. 3, 2015, pp. 601–615. https://doi.org/10.1109/TAC.2014.2364096.

[23] Mokhtari, A., Ling, Q., and Ribeiro, A., "Network Newton Distributed Optimization Methods," *IEEE Transactions on Signal Processing*, Vol. 65, No. 1, 2017, pp. 146–161. https://doi.org/10.1109/TSP.2016.2617829.

[24] Lorenzo, P. D., and Scutari, G., "NEXT: In-Network Nonconvex Optimization," *IEEE Transactions on Signal and Information Processing over Networks*, Vol. 2, No. 2, 2016, pp. 120–136. https://doi.org/10.1109/TSIPN.2016.2524588.

[25] Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J., "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends® in Machine Learning*, Vol. 3, No. 1, 2011, pp. 1–122. https://doi.org/10.1561/2200000016.

[26] Shi, W., Ling, Q., Yuan, K., Wu, G., and Yin, W., "On the Linear Convergence of the ADMM in Decentralized Consensus Optimization," *IEEE Transactions on Signal Processing*, Vol. 62, No. 7, 2014, pp. 1750–1761. https://doi.org/10.1109/TSP.2014.2304432.

[27] Saravanos, A. D., Tsolovikos, A., Bakolas, E., and Theodorou, E. A., "Distributed Covariance Steering with Consensus ADMM for Stochastic Multi-Agent Systems," *Robotics: Science and Systems XVII*, 2021. URL https://api.semanticscholar.org/CorpusID:235651825.

[28] Shorinwa, O., Halsted, T., and Schwager, M., "Scalable Distributed Optimization with Separable Variables in Multi-Agent Networks," *2020 American Control Conference (ACC)*, 2020, pp. 3619–3626. https://doi.org/10.23919/ACC45564.2020.9147590.

[29] Zhang, X., Cheng, Z., Ma, J., Huang, S., Lewis, F. L., and Lee, T. H., "Semi-Definite Relaxation-Based ADMM for Cooperative Planning and Control of Connected Autonomous Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 23, No. 7, 2022, pp. 9240–9251. https://doi.org/10.1109/TITS.2021.3094215.

[30] Zhang, X., Cheng, Z., Ma, J., Zhao, L., Xiang, C., and Lee, T. H., "Parallel Collaborative Motion Planning with Alternating Direction Method of Multipliers," *IECON 2021 – 47th Annual Conference of the IEEE Industrial Electronics Society*, 2021, pp. 1–6. https://doi.org/10.1109/IECON48115.2021.9589675.

[31] Saravanos, A. D., Aoyama, Y., Zhu, H., and Theodorou, E. A., "Distributed Differential Dynamic Programming Architectures for Large-Scale Multiagent Control," *IEEE Transactions on Robotics*, 2023, pp. 1–21. https://doi.org/10.1109/TRO.2023.3319894.

[32] Horyna, J., Baca, T., Walter, V., Albani, D., Hert, D., Ferrante, E., and Saska, M., "Decentralized swarms of unmanned aerial vehicles for search and rescue operations without explicit communication," *Autonomous Robots*, Vol. 47, No. 1, 2023, pp. 77–93. https://doi.org/10.1007/s10514-022-10066-5.

[33] Li, A., Hansen, M., and Zou, B., "Traffic management and resource allocation for UAV-based parcel delivery in low-altitude urban space," *Transportation Research Part C: Emerging Technologies*, Vol. 143, 2022, p. 103808. https://doi.org/10.1016/j.trc.2022.103808.

[34] Ni, R., Pan, Z., and Gao, X., "Robust Multi-Robot Trajectory Optimization Using Alternating Direction Method of Multiplier," *IEEE Robotics and Automation Letters*, Vol. 7, No. 3, 2022, pp. 5950–5957. https://doi.org/10.1109/LRA.2022.3159848.

[35] Zhou, X., Wen, X., Wang, Z., Gao, Y., Li, H., Wang, Q., Yang, T., Lu, H., Cao, Y., Xu, C., and Gao, F., "Swarm of Micro Flying Robots in the Wild," *Science Robotics*, Vol. 7, No. 66, 2022, p. eabm5954. https://doi.org/10.1126/scirobotics.abm5954.

[36] Zheng, X., He, S., and Lin, D., "Constrained Trajectory Optimization With Flexible Final Time for Autonomous Vehicles," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 58, No. 3, 2022, pp. 1818–1829. https://doi.org/10.1109/TAES.2021.3121668.

[37] Zheng, X., Guan, H., Lin, D., He, S., and Shin, H.-S., "Game-Theoretic Flexible-Final-Time Differential Dynamic Programming Using Gaussian Quadrature," *Journal of Guidance, Control, and Dynamics*, Vol. 46, No. 4, 2023, pp. 742–751. https://doi.org/10.2514/1.G007113.

[38] Oshin, A., Houghton, M. D., Acheson, M. J., Gregory, I. M., and Theodorou, E. A., "Parameterized Differential Dynamic Programming," *arXiv preprint*, 2022. https://doi.org/10.48550/arXiv.2204.03727.

[39] Goldstein, T., O'Donoghue, B., Setzer, S., and Baraniuk, R., "Fast Alternating Direction Optimization Methods," *SIAM Journal on Imaging Sciences*, Vol. 7, No. 3, 2014, pp. 1588–1623. https://doi.org/10.1137/120896219.

[40] Xu, Z., Figueiredo, M., and Goldstein, T., "Adaptive ADMM with Spectral Penalty Parameter Selection," *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, Vol. 54, edited by A. Singh and J. Zhu, PMLR, 2017, pp. 718–727. URL https://proceedings.mlr.press/v54/xu17a.html.

[41] Xu, Z., Figueiredo, M. A. T., Yuan, X., Studer, C., and Goldstein, T., "Adaptive Relaxed ADMM: Convergence Theory and Practical Implementation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[42] Tang, W., and Daoutidis, P., "Distributed nonlinear model predictive control through accelerated parallel ADMM," *2019 American Control Conference (ACC)*, 2019, pp. 1406–1411. https://doi.org/10.23919/ACC.2019.8814732.

[43] Rockafellar, R. T., *Convex Analysis in the Calculus of Variations*, Springer US, Boston, MA, 2001, pp. 135–151. https://doi.org/10.1007/978-1-4613-0279-7_7.

[44] BARZILAI, J., and BORWEIN, J. M., "Two-Point Step Size Gradient Methods," *IMA Journal of Numerical Analysis*, Vol. 8, No. 1, 1988, pp. 141–148. https://doi.org/10.1093/imanum/8.1.141.

[45] Zhou, B., Gao, L., and Dai, Y.-H., "Gradient Methods with Adaptive Step-Sizes," *Computational Optimization and Applications*, Vol. 35, No. 1, 2006, pp. 69–86. https://doi.org/10.1007/s10589-006-6446-0.

[46] Xu, Z., Taylor, G., Li, H., Figueiredo, M. A. T., Yuan, X., and Goldstein, T., "Adaptive Consensus ADMM for Distributed Optimization," *Proceedings of the 34th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 70, edited by D. Precup and Y. W. Teh, PMLR, 2017, pp. 3841–3850. URL https://proceedings.mlr.press/v70/xu17c.html.