# Distributed Multi-Robot Task Assignment via Consensus ADMM

Ola Shorinwa, Ravi Haksar, Patrick Washington, Mac Schwager

***Abstract*—We present a distributed algorithm to solve a class of multi-robot task assignment problems. We formulate task assignment as a mathematical optimization and solve for optimal solutions with a variant of the consensus Alternating Direction Method of Multipliers (C-ADMM). We provide C-ADMM-based algorithms for both the primal and dual problem formulations, and show the advantages of each form depending on the problem specifics. In our algorithm, each robot solves a series of local optimization problems and communicates the results to its local neighbors, ultimately converging to an optimal task assignment in problems with linear objective functions, and an optimal solution of the relaxed problem in convex problems. While many other distributed algorithms require a central station for their implementation, in our algorithm each robot only communicates with its one-hop neighbors. In linear task assignment problems, our algorithm converges to the optimal task assignment, unlike many other distributed algorithms for this problem, which yield sub-optimal solutions. We demonstrate our algorithms in task assignment problems over a variety of communication network topologies, where we show that our inexact dual algorithm is at least $60\%$ faster than other distributed algorithms which produce an optimal task assignment. In addition, our dual algorithm attains a $69\%$ speedup in computation time compared to a notable distributed variant of the Hungarian method [1]. We also apply our algorithm to a multi-UAV persistent surveillance problem, showing its suitability for problems involving periodic task assignments.**

***Index Terms*—Distributed Task Assignment, Distributed Mathematical Programming, Distributed Optimization, Multi-Robot Systems**

## I. Introduction

MULTI-ROBOT task assignment problems arise in a variety of applications, including disaster and rescue operations [2], persistent surveillance [3], [4], package delivery [5], [6], and transportation [7]–[9], where the deployment of multiple robots allows for the completion of several tasks concurrently. In many situations, the robots possess heterogeneous capabilities, with certain robots better suited for specific tasks. In realistic settings, each robot has access to only its local data and communication links to local neighbors, and as a result, computing the optimal task assignment requires collaboration among all robots. Centralized methods collate all the information available to each robot at a central station, which computes the optimal task assignment. This approach presents significant computational and communication overhead, especially in problems with a large number of robots, and introduces a brittle single point of failure in the central station. In this paper, we introduce distributed algorithms for the multi-robot task assignment problem, where each robot computes an optimal task assignment in problems with linear objective functions and more generally, an optimal solution of the relaxed task assignment problem locally, overcoming these challenges.

We formulate the multi-robot task assignment problem as a mathematical program, considering both its primal and dual forms, noting that these convex formulations yield the optimal task assignment [10] when the objective function consists of a sum of linear functions. This paper has evolved from our earlier work [11], in which we derived specialized primal and inexact dual algorithms for the multi-robot task assignment problem, demonstrating its application to task assignment problems with linear objective functions. In this paper, we extend [11] by presenting a distributed exact dual algorithm, in addition to providing a more thorough analysis of our distributed algorithms, which include a multi-robot primal algorithm for the task assignment problem (MUR-TAP), a multi-robot exact dual algorithm for the task assignment problem (MURD-TAP), and a multi-robot inexact dual algorithm for the task assignment problem (MURID-TAP). Collectively, we refer to our distributed algorithms as the *MUR Family* of algorithms for *task assignment* problems. Our distributed task assignment algorithms apply to task assignment problems with convex (linear or nonlinear) objective functions and affine constraints, a broader class of task assignment problems than those previously treated in the literature, which subsumes the classical task assignment problem with linear objective functions.

Our distributed algorithms are based on the alternating direction method of multipliers (ADMM), which builds upon the method of multipliers and the dual decomposition method, blending the benefits of both methods to provide improved convergence for a broader class of problems. While ADMM typically requires a central station for updating the dual variables [12], [13], we utilize a fully-distributed variant of ADMM, Consensus ADMM [14], which imposes consensus constraints on the local optimization variables of each robot, ensuring that all the robots arrive at the same optimal solution. Other fully-distributed variants of ADMM exist [15], [16];

Manuscript received April 19, 2021; revised August 16, 2021. This work was supported in part by DARPA YFA award D18AP00064 and NSF NRI awards 1830402 and 1925030. The NASA University Leadership initiative (grant #80NSSC20M0163) provided funds to assist the authors with their research, but this article solely reflects the opinions and conclusions of its authors and not any NASA entity. Toyota Research Institute provided funds to support this work. We are grateful for this support.

O. Shorinwa is with the Department of Mechanical Engineering, Stanford University, CA, USA, shorinwa@stanford.edu.

R. Haksar is with Nuro, Mountain View, CA, USA, rhaksar@nuro.ai.

P. Washington and M. Schwager are with the Department of Aeronautics and Astronautics Engineering, Stanford University, CA, USA, {phw, schwager}@stanford.edu.

however, these methods place stringent conditions on the problem.

In our algorithms, each robot solves a local optimization problem iteratively, while communicating with its neighbors, to compute an optimal solution of the mathematical program. We assume that each robot communicates with only its one-hop neighbors over a point-to-point communication network. Further, our algorithm does not require strategic data flow during the communication process. Although each robot has access to only its local problem data, all the robots compute the same optimal solution, without relying on a central station at any point. Consequently, our algorithms offer stronger protection against adversarial attacks, eliminating vulnerabilities associated with a single point of failure. Moreover, the iterates in our algorithms converge to the optimal solution of the relaxed task assignment problem, provided the objective function of the problem is proper and convex, and the Lagrangian of the problem has a saddle-point (standard technical conditions for solving convex optimization problems), unlike many existing distributed methods which produce sub-optimal solutions. As with other distributed methods, our algorithms provide better privacy by minimizing the amount of data shared by each robot, as each robot does not share its local objective and constraint functions data with other robots.

We demonstrate the performance of our task assignment algorithms in comparison to other distributed task assignment algorithms, showing the balanced trade-off provided by our algorithms with respect to communication and computation overhead. Particularly, MURID-TAP is about $60\%$ faster compared to the best competing distributed algorithm [17], among distributed algorithms which produce an optimal task assignment, in terms of the cumulative computation time required by each robot to compute an optimal task assignment. In comparison to a distributed variant of the widely used Hungarian method for task assignment [1], MURD-TAP provides about a $69\%$ speedup in computation time per robot at the expense of a greater number of communication rounds. Further, we demonstrate the application of our distributed algorithm to persistent surveillance problems, where our algorithm produces the optimal task assignment at each assignment episode. In addition, we show the amenability of our algorithms to problems with different communication network topology and further, highlight the versatility of our algorithms in multi-robot problems with constraints on the availability of different resources, including computational, communication, and data storage resources.

## II. Related Works

The task assignment problem, a canonical optimization scenario, with applications extending beyond the multi-robot systems domain to task scheduling and operations management, is widely solved using the Hungarian algorithm [18]—a centralized algorithm which requires knowledge of the costs of all possible robot and task pairs. The need for global access to the costs of all robot and task pairs renders the algorithm unsuitable for multi-robot problems, where each robot only has access to its local costs for each task. Dispersing information on the costs among all robots introduces computational and communication overhead, which degrades the efficiency of the algorithm. A few distributed variants of the Hungarian algorithm overcome these challenges [1], [19] and still provide the optimal task assignment. As with the centralized Hungarian algorithm, these distributed variants only apply to problems with linear objective cost functions, limiting the scope of multi-robot task assignment problems which can be solved using these methods. In contrast, our methods can be used for problems with general convex objectives.

Another popular approach, auction methods, involve a market construct where the robots negotiate with their neighbors to compute an assignment which minimizes their costs while ensuring that no task remains unassigned. Typically, a centralized auctioneer manages the auctions, receiving bids from the robots before assigning the task to the winning bidder [20]–[23]. Consequently, these methods require all robots to communicate with the auctioneer [24]. Other methods designate an auctioneer from the group of robots which also doubles as a bidder [25], [26], with the auction process only being executed when the robots are within range of their neighbors for spatially-distributed tasks [27], [28]. In other methods, each robot serves as an auctioneer, updating the assignment of all tasks iteratively upon receiving bids from its neighbors until all the tasks are assigned [29]–[34]. Despite the relative simplicity of auction methods, these methods do not generally provide an optimal assignment of all tasks [35], [36]. However, the auction method in [17], which involves a conflict resolution procedure for robots assigned to the same task, results in an optimal assignment. Unlike auction or market methods, our method produces the optimal task assignment without relying on any designated auctioneer.

The task assignment problem can be formulated as a mixed-integer optimization problem with binary assignment variables. As such, mixed-integer linear programming (MILP) methods apply to these problems with linear objective functions. The MILP methods in [37]–[40] solve the task assignment problem at a central station which collates all the relevant local information available to each robot. Some distributed MILP methods employ dual decomposition in solving the task assignment problem [41] without requiring each robot to communicate its local objective functions. However, these methods still require a central station for updating the dual variables after the local primal updates performed by each robot. Other methods utilize a primal decomposition approach to compute an approximate solution to the task assignment problem [42]. In [43], each robot computes its assignment from the dual formulation of the task assignment problem by solving MILP sub-problems iteratively using the branch and bound procedure. Other MILP methods utilize a distributed cutting-plane procedure to solve the task assignment problem but require the objective function to be integer-valued [44], introducing an additional limitation on the class of task assignment problems suitable for these methods.

Some distributed methods consider a convex relaxation of the task assignment problem to a linear program with real-valued variables over the unit simplex, producing the same optimal solution as the original mixed-integer problem. These

distributed methods only apply to task assignment problems with linear objective functions. In these methods, the robots compute the optimal assignment using a distributed simplex algorithm [10] or dual decomposition [45]. The surveys [46] and [47] provide an overview of methods for the multi-robot task assignment problem, noting the conditions required for implementation of these methods. We consider a convex relaxation of the task assignment problem. However, we do not limit the scope of our algorithms to problems with linear objective functions. Rather, we derive algorithms that equally apply to problems with nonlinear objective functions. In addition, our algorithms do not require any stringent condition on the nature of the objective function or the topology of the communication network.

## III. Contributions

We summarize our contributions below:

- We provide distributed algorithms for multi-robot task assignment problems (MUR-TAP and MURD-TAP), where each robot computes the optimal solution of the relaxed task assignment problem, which corresponds to the optimal task assignment in problems with linear objective functions, by solving a sequence of local optimization problems. In our algorithms, each robot communicates with other neighboring robots over a local network, without any stringent conditions on the topology of the communication network.
- In certain problems, multi-robot systems operate with limited access to computational resources. As a result, solving the optimization problems arising in MUR-TAP and MURD-TAP might be computationally demanding for these systems. We derive MURID-TAP for these specific situations. MURID-TAP enables each robot to compute an optimal solution of the relaxed task assignment problem via closed-form updates, without having to resort to a nested iterative optimization method to solve its local optimization problems.
- Compared to existing distributed algorithms for the task assignment problem, which produce an optimal task assignment [1], [17], MURID-TAP provides at least $60\%$ faster performance in computing an optimal task assignment, with respect to the cumulative computation time per robot.

### *Organization*

The paper is organized as follows: We present the classical task assignment problem with linear objective functions in Section IV before providing a more general formulation for problems with nonlinear objective functions and affine constraints, noting its distributed nature over a network of robots where each robot has access to only its local objective and constraint functions. We derive task assignment problems in Section VI. We demonstrate our distributed multi-robot task assignment algorithms in Section VII, examining their performance in comparison to other distributed task assignment methods. We provide concluding remarks in Section VIII.

## IV. Problem Formulation

We consider the classical multi-robot task assignment problem where we seek an optimal assignment of $N$ robots to $N$ tasks, described by the optimization problem

$$\begin{aligned} \underset{x}{\text{minimize}} \ & \sum_{i=1}^{N} c_i^\mathsf{T} x_i \\ \text{subject to} \ & \sum_{i=1}^{N} x_i = \mathbf{1}_N \\ & \mathbf{1}_N^\mathsf{T} x_i = 1 \quad \forall i \in \mathcal{V} \\ & x_{i,\tau} \in \{0,1\} \quad \forall \tau \in \mathcal{T},\ \forall i \in \mathcal{V} \end{aligned} \tag{1}$$

where $x_i \in \mathbb{R}^N$ denotes the optimization variable of robot $i$, with component $\tau$ of $x_i$ indicating if a robot is assigned to task $\tau$ and $x = [x_1^\mathsf{T}, \cdots, x_N^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{N^2}$. In addition, $c_i \in \mathbb{R}^N$ denotes the objective cost vector of robot $i$. We denote the set of all tasks by $\mathcal{T} = \{1, \cdots, N\}$. The optimization problem in (1) represents an integer optimization problem from the binary constraints on $x_i$. Solving this combinatorial optimization problem proves challenging, with typical algorithms often resorting to branch-and-bound methods, which fail to scale to problems with large numbers of robots. Generally, existing methods solve a relaxation of the task assignment problem which replaces the binary constraints with box constraints on $x_i,\ \forall i \in \mathcal{V}$. The relaxed problem is given by

$$\begin{aligned} \underset{x}{\text{minimize}} \ & \sum_{i=1}^{N} c_i^\mathsf{T} x_i \\ \text{subject to} \ & \sum_{i=1}^{N} x_i = \mathbf{1}_N \\ & \mathbf{1}_N^\mathsf{T} x_i = 1 \quad \forall i \in \mathcal{V} \\ & 0 \leq x_{i,\tau} \leq 1 \quad \forall \tau \in \mathcal{T},\ \forall i \in \mathcal{V} \end{aligned} \tag{2}$$

with $x_i$ constrained to lie between 0 and 1. Given a linear objective function, an optimal solution of the problem (2) always occurs at a vertex of the feasible set, since the relaxed optimization problem has a bounded feasible set, although the optimal solution might not be unique. As a result, an integer-valued solution can always be obtained from an optimal solution of (2). Moreover, this solution corresponds to an optimal solution of the integer optimization problem in (1).

In more complex problems, the costs incurred by each robot in performing a task is better captured by nonlinear objective functions, e.g., in problems with strictly convex cost functions. In addition, many task assignment problems often involve additional constraints arising from individual preferences and task priorities. As a result, we extend the problem formulation in (1) to consider a broader class of task assignment problems with $N$ robots, $m$ tasks, convex (linear/nonlinear) objective

functions, and affine constraints, given by

$$\begin{aligned} \underset{x}{\text{minimize}} \ & \sum_{i=1}^{N} f_i(x_i) \\ \text{subject to} \ & \sum_{i=1}^{N} x_i \geq \mathbf{1}_m \\ & \mathbf{1}_m^\mathsf{T} x_i = 1 \quad \forall i \in \mathcal{V} \\ & x_{i,\tau} \in \{0,1\} \quad \forall \tau \in \mathcal{T},\ \forall i \in \mathcal{V} \end{aligned} \tag{3}$$

where $x_i \in \mathbb{R}^m$ denotes the optimization variable of robot $i$, $x = [x_1^\mathsf{T}, \cdots, x_N^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{Nm}$, $f_i(x) \colon \mathbb{R}^m \to \mathbb{R}$ denotes the local convex objective function of robot $i$, and $\mathcal{T} = \{1, \cdots, m\}$. We note that solving the task assignment problem in (3) is even more challenging compared to solving (1), particularly via a distributed approach. Consequently, we consider a relaxation of (3), similar to (2). The relaxed problem is given by

$$\begin{aligned} \underset{x}{\text{minimize}} \ & \sum_{i=1}^{N} f_i(x_i) \\ \text{subject to} \ & \sum_{i=1}^{N} x_i \geq \mathbf{1}_m \\ & \mathbf{1}_m^\mathsf{T} x_i = 1 \quad \forall i \in \mathcal{V} \\ & 0 \leq x_{i,\tau} \leq 1 \quad \forall \tau \in \mathcal{T},\ \forall i \in \mathcal{V} \end{aligned} \tag{4}$$

where $x_i$ lies between 0 and 1. The formulation in (4) encompasses task assignment problems with an unequal number of robots and tasks, where $N \neq m$. We note that the optimal solution of (4) might not be integer-valued. In these cases, heuristics for obtaining a binary assignment can be employed [48], [49]. However, if an integer optimal solution is obtained for (4), then this solution is optimal for the original integer optimization problem.

## V. Preliminaries

We represent the robots as nodes in an undirected communication graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a set of vertices $\mathcal{V} = \{1, \cdots, N\}$ and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. An edge $(i, j)$ exists in $\mathcal{E}$ if robots $i$ and $j$ share a communication link. To ensure all robots compute the same solution, we make the following assumptions on the communication graph between the robots.

**Assumption 1.** *The communication graph $\mathcal{G}$ is connected.*

This assumption indicates that a communication path exists between any pair of robots, possibly involving multiple hops along the edges in $\mathcal{E}$.

We denote the $n \times n$ identity matrix as $I_n$ and the vector of all ones as $\mathbf{1}_n$. We interpret box constraints on vector-valued variables element-wise. We provide the following definition of a proper, closed, and convex function, in addition to the definition of a coercive function, before introducing the next assumption.

**Definition 1.** *A function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is proper if it does not attain a value of $+\infty$ everywhere, i.e., there exists $x \in \mathbb{R}^n$ such that $f(x) \in \mathbb{R}$. Further, $f$ is closed if its epigraph $\text{epi}(f) = \{(x, t) \in \mathbb{R}^n \times \mathbb{R} \mid x \in \mathbb{R}^n, t \geq f(x)\}$ is closed.*

**Definition 2.** *A function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is convex if for any $x \in \mathbb{R}^n$, $y \in \mathbb{R}^n$, and $t \in \mathbb{R}$ with $t \in [0, 1]$*

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y). \tag{5}$$

**Definition 3.** *A continuous function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is coercive if*

$$\lim_{\|x\| \to \infty} f(x) = +\infty, \tag{6}$$

*i.e., for any constant $L > 0$, there exists a constant $R_L > 0$ such that $\|f(x)\| > L$ when $\|x\| > R_L$.*

We assume the Lagrangian of the task assignment problem in (4) has a saddle-point.

**Assumption 2.** *The objective function in (7) is proper, closed, and convex, and a saddle-point exists for the Lagrangian of the task assignment problem.*

The Lagrangian of the task assignment problem has a saddle-point if the task assignment problem is feasible. As such, Assumption 2 is not restrictive, as it is satisfied in almost all practical cases. This assumption enables us to compute a primal-dual solution of the problem in (4). Further, this assumption indicates that strong duality holds with equality between the optimal primal objective value and the optimal dual objective value. This assumption is standard in many papers in constrained optimization, although it is often not stated in many of these papers.

## VI. Distributed Algorithms for Multi-Robot Task Assignment Problems

In the following discussion, we derive distributed algorithms for the multi-robot task assignment problem (4), considering its primal and dual forms.

### A. Distributed Primal Algorithm

We derive a distributed algorithm for the task assignment problem in (4). To obtain a concise formulation, we group all the constraints in (4) into a single affine constraint, representing the optimization problem as

$$\begin{aligned} \mathcal{P}: \quad & \underset{x}{\text{minimize}} \ \sum_{i=1}^{N} f_i(x) \\ & \text{subject to} \ A_i x \leq b_i \quad \forall i \in \mathcal{V} \end{aligned} \tag{7}$$

where

$$A_i = \begin{bmatrix} -\check{P} \\ \hline \check{Q}_i \\ \hline -\check{Q}_i \\ \hline -I_{Nm} \\ \hline I_{Nm} \end{bmatrix}, \quad b_i = \begin{bmatrix} -\mathbf{1}_m \\ \hline 1 \\ \hline -1 \\ \hline \mathbf{0}_{Nm} \\ \hline \mathbf{1}_{Nm} \end{bmatrix}, \tag{8}$$

and $\check{P} \in \mathbb{R}^{m \times Nm}$ represents a horizontal block matrix consisting of $N$ identity matrices $(I_m)$ concatenated horizontally, $\check{Q}_i \in \mathbb{R}^{1 \times Nm}$ represents a horizontal block matrix of all zeros except the $i$-th horizontal block component which is set to $\mathbf{1}_m^\mathsf{T}$. Although the local objective function of robot $i$

only depends on $x_i$, which is a component of $x$, we express it in a more general form in (7), for simplicity.

We express the problem in (7) in its distributed form by introducing local copies of the optimization variables maintained by each robot. Robot $i$ maintains $\boldsymbol{x}_i \in \mathbb{R}^{Nm}$, representing a copy of $x$, with the resulting distributed form of the problem given by

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \ & \sum_{i=1}^{N} f_i(\boldsymbol{x}_i) \\ \text{subject to} \ & A_i \boldsymbol{x}_i \leq b_i \quad \forall i \in \mathcal{V} \\ & \boldsymbol{x}_i = \boldsymbol{x}_j \quad \forall j \in \mathcal{N}_i, \ \forall i \in \mathcal{V} \end{aligned} \tag{9}$$

where $\boldsymbol{x} = [\boldsymbol{x}_i^{\mathsf{T}}, \ \forall i \in \mathcal{V}]^{\mathsf{T}}$ denotes the vertical concatenation of the optimization variables of all robots. We introduce the equality constraint between the optimization variables of neighboring robots to ensure agreement between all robots.

**Proposition 1.** *The distributed problem in* (9) *is equivalent to the mathematical programming problem in* (7) *with the same optimal solution and optimal objective value.*

*Proof.* From the equality constraints in (9), all robots compute the same solution for their local optimization variables since the communication graph $\mathcal{G}$ is connected. Consequently, we can replace the local optimization variable of each robot with a common variable $\tilde{x}$. The resulting optimization problem has the same objective function and feasible set as the problem in (7). Hence, both problems have the same optimal solution and optimal objective value. □

We derive a distributed method for solving the mathematical program in (9) using Consensus ADMM [14]. We introduce the local slack variables $\alpha_{ij} \in \mathbb{R}^{Nm}$ and $\beta_{ij} \in \mathbb{R}^{Nm}$ into the equality constraints between robots $i$ and $j$ in (9), expressing the problem as

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{\sigma}}{\text{minimize}} \ & \sum_{i=1}^{N} f_i(\boldsymbol{x}_i) \\ \text{subject to} \ & A_i \boldsymbol{x}_i \leq b_i \quad \forall i \in \mathcal{V} \\ & \boldsymbol{x}_i = \alpha_{ij} \quad \forall j \in \mathcal{N}_i, \ \forall i \in \mathcal{V} \\ & \boldsymbol{x}_j = \beta_{ij} \quad \forall j \in \mathcal{N}_i, \ \forall i \in \mathcal{V} \\ & \alpha_{ij} = \beta_{ij} \quad \forall j \in \mathcal{N}_i, \ \forall i \in \mathcal{V} \end{aligned} \tag{10}$$

where robot $i$ optimizes over $\alpha_{ij}$, robot $j$ optimizes over $\beta_{ij}$, and $\boldsymbol{\sigma} \in \mathbb{R}^{2|\mathcal{E}|Nm}$ denotes the vertical concatenation of all the slack variables. With the slack variables, the augmented Lagrangian of (9) can be expressed as

$$\begin{aligned} \mathcal{L}_a^p(\boldsymbol{x}, \boldsymbol{\sigma}, u, v) = & \sum_{i=1}^{N} f_i(\boldsymbol{x}_i) \\ & + \sum_{(i,j)\in\mathcal{E}} \Big( u_{ij}^{\mathsf{T}}(\boldsymbol{x}_i - \alpha_{ij}) + v_{ij}^{\mathsf{T}}(\boldsymbol{x}_j - \beta_{ij}) \Big) \\ & + \frac{\rho}{2} \sum_{(i,j)\in\mathcal{E}} \Big( \|\boldsymbol{x}_i - \alpha_{ij}\|_2^2 + \|\boldsymbol{x}_j - \beta_{ij}\|_2^2 \Big) \end{aligned} \tag{11}$$

where $u_{ij} \in \mathbb{R}^{Nm}$ and $v_{ij} \in \mathbb{R}^{Nm}$ denote the Lagrange multipliers for the equality constraints between the optimization variables of robots $i$ and $j$ respectively. We have not relaxed the affine inequality constraint in (9) but rather enforce that the local primal variable of each robot satisfies its local inequality constraint. Likewise, we enforce the constraint $\alpha_{ij} = \beta_{ij}$. The augmented Lagrangian includes a penalty on the violation of the equality constraints, with the contribution of this violation determined by the parameter $\rho \in \mathbb{R}$. In ADMM, the primal variables are updated iteratively as the minimizers of the augmented Lagrangian using the Lagrange multipliers at the previous iteration before updating the Lagrange multipliers through gradient ascent on the augmented Lagrangian.

The update procedure for the slack variable consists of minimizing a strongly convex quadratic problem with a closed-form solution, given by

$$\alpha_{ij}^{k+1} = \beta_{ij}^{k+1} = \frac{1}{2}\big(\boldsymbol{x}_i^{k+1} + \boldsymbol{x}_j^{k+1}\big) \tag{12}$$

if the Lagrange multipliers are initialized with $u_{ij}^0 = v_{ij}^0 = 0$ (refer to [14] for additional details). Similarly, the update procedures of the Lagrange multipliers simplify to

$$\begin{aligned} u_{ij}^{k+1} &= u_{ij}^k + \frac{\rho}{2}\big(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_j^{k+1}\big) \\ v_{ij}^{k+1} &= v_{ij}^k + \frac{\rho}{2}\big(\boldsymbol{x}_j^{k+1} - \boldsymbol{x}_i^{k+1}\big) \end{aligned} \tag{13}$$

at iteration $k$. With the specified initialization of the Lagrange multipliers, $u_{ij}^k = -v_{ij}^k$ at each iteration $k$. To update its primal variable $\boldsymbol{x}_i$ at iteration $k$, robot $i$ solves the minimization problem

$$\begin{aligned} \underset{\boldsymbol{x}_i}{\text{minimize}} \ & \left\{ f_i(\boldsymbol{x}_i) + \sum_{j\in\mathcal{N}_i} \big(u_{ij}^k + v_{ji}^k\big)^{\mathsf{T}} \boldsymbol{x}_i \right. \\ & \left. + \rho \sum_{j\in\mathcal{N}_i} \left\| \boldsymbol{x}_i - \frac{\boldsymbol{x}_i^k + \boldsymbol{x}_j^k}{2} \right\|_2^2 \right\} \\ \text{subject to} \ & A_i \boldsymbol{x}_i \leq b_i \end{aligned} \tag{14}$$

which simplifies to

$$\begin{aligned} \underset{\boldsymbol{x}_i}{\text{minimize}} \ & \left\{ f_i(\boldsymbol{x}_i) + q_i^{k\mathsf{T}} \boldsymbol{x}_i + \rho \sum_{j\in\mathcal{N}_i} \left\| \boldsymbol{x}_i - \frac{\boldsymbol{x}_i^k + \boldsymbol{x}_j^k}{2} \right\|_2^2 \right\} \\ \text{subject to} \ & A_i \boldsymbol{x}_i \leq b_i \end{aligned} \tag{15}$$

where

$$q_i^k = \sum_{j\in\mathcal{N}_i} \big(u_{ij}^k + v_{ji}^k\big), \tag{16}$$

by combining the local Lagrange multipliers of robot $i$ into $q_i \in \mathbb{R}^n$. Subsequently, robot $i$ updates $q_i$ using

$$q_i^{k+1} = q_i^k + \rho \sum_{j\in\mathcal{N}_i} \big(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_j^{k+1}\big) \tag{17}$$

from the update procedures in (13).

Algorithm 1 outlines our distributed primal algorithm for the task assignment problem. Each robot does not compute the slack variables $\alpha$ and $\beta$ and the Lagrange multipliers $u$ and $v$.

As noted in Algorithm 1, robot $i$ computes its local primal variable $\boldsymbol{x}_i$ along with its Lagrange multiplier $q_i$. Our algorithm for (9), MUR-MP, does not require a feasible initialization of

**Algorithm 1:** Distributed Multi-Robot Algorithm for Task Assignment Problems (MUR-TAP)

Initialization:
$\boldsymbol{x}_i^0 \in \mathbb{R}^n \quad \forall i \in \mathcal{V}$
$q_i^0 \leftarrow 0 \quad \forall i \in \mathcal{V}$
$k \leftarrow 0$
**do in parallel** $\forall i \in \mathcal{V}$
  $\boldsymbol{x}_i^{k+1} \leftarrow$ Procedure (15)
  $q_i^{k+1} \leftarrow$ Procedure (17)
  $k \leftarrow k+1$
**while** *not converged or stopping criterion is not met*;

the local primal variables of each robot. Upon updating its local variables, each robot communicates its primal variables with its neighbors.

**Theorem 1** (Convergence of $\{\boldsymbol{x}^k\}$)**.** *The sequence of primal variables $\{\boldsymbol{x}_i^k\}$ of robot $i$ converges to the optimal solution $x^\star$ of the mathematical program in* (7)*, $\forall i \in \mathcal{V}$.*

*Proof.* We provide the proof in the Appendix. □

### *B. Distributed Dual Algorithm*

We note that the MUR-TAP algorithm requires robot $i$ to maintain a copy of the entire task assignment problem variable $x \in \mathbb{R}^{Nm}$, instead of only maintaining its local task assignment variable $x_i \in \mathbb{R}^n$, which can result in unnecessary computation overhead. As such, we derive a distributed algorithm for task assignment problems by considering the dual formulation of the problem, where robot $i$ maintains only its task assignment variable $x_i$, without maintaining the task assignment variables of other robots. First, we express the task assignment problem in (4) as

$$\begin{aligned} \mathcal{D}: \quad & \underset{x}{\text{minimize}} \ \sum_{i=1}^{N} f_i(x_i) \\ & \text{subject to} \ \sum_{i=1}^{N} \big(A_i x_i - b_i\big) \leq 0 \\ & \qquad x_i \geq 0 \quad \forall i \in \mathcal{V} \end{aligned} \tag{18}$$

where

$$A_i = \left[\begin{array}{c} -I_m \\ \hline G_i \\ \hline -G_i \\ \hline H_i \end{array}\right], \quad b_i = \left[\begin{array}{c} -\frac{1}{N}\mathbf{1}_m \\ \hline \frac{1}{N}\mathbf{1}_N \\ \hline -\frac{1}{N}\mathbf{1}_N \\ \hline \check{v}_i \end{array}\right], \tag{19}$$

$A_i \in \mathbb{R}^{e \times m}$, with $e = m + 2N + Nm$, and $G_i \in \mathbb{R}^{N \times m}$ has all its entries set to zero except the $i$-th row of $G_i$ which is set to $\mathbf{1}_m^\mathsf{T}$. Likewise, $H_i \in \mathbb{R}^{Nm \times Nm}$ represents a block diagonal matrix of all zeros with the $i$-th block set to $I_m$, while $\check{v}_i \in \mathbb{R}^{Nm}$ represents a block column vector of all zeros with the $i$-th block set to $\mathbf{1}_m$.

The Lagrangian of the mathematical program in (18) is given by

$$\mathcal{L}(x, y) = \sum_{i=1}^{N} f_i(x_i) + \tilde{y}^\mathsf{T} \sum_{i=1}^{N} \big(A_i x_i - b_i\big) \tag{20}$$

where $\tilde{y} \in \mathbb{R}^e$ denotes the dual variable for the separable coupling constraint in (18), with $\tilde{y} \geq 0$.

From the Lagrangian, we obtain the dual function associated with (18), given by

$$g(\tilde{y}) = \inf_{x} \left\{ \sum_{i=1}^{N} \big( f_i(x_i) + \tilde{y}^\mathsf{T} \big(A_i x_i - b_i\big)\big) \right\} \tag{21}$$

involving the minimization of the Lagrangian $\mathcal{L}(\cdot)$ over $x$. The dual problem associated with the mathematical program $\mathcal{D}$ in (18) consists of maximizing the dual function with respect to the dual variable $\tilde{y}$, resulting in the problem

$$\begin{aligned} & \underset{\tilde{y}}{\text{maximize}} \ g(\tilde{y}) \\ & \text{subject to} \ y \geq 0 \end{aligned} \tag{22}$$

which simplifies to

$$\begin{aligned} & \underset{\tilde{y}}{\text{maximize}} \ \sum_{i=1}^{N} \ \inf_{x_i} \big( f_i(x_i) + \tilde{y}^\mathsf{T} \big(A_i x_i - b_i\big)\big) \\ & \text{subject to} \ \tilde{y} \geq 0 \end{aligned} \tag{23}$$

from separability of the objective function in (21). Using Fenchel conjugates, the dual problem reduces to

$$\begin{aligned} & \underset{y}{\text{maximize}} \ \sum_{i=1}^{N} \ (-f_i^\star(-A_i^\mathsf{T}\tilde{y}) - \tilde{y}^\mathsf{T} b_i) \\ & \text{subject to} \ \tilde{y} \geq 0 \end{aligned} \tag{24}$$

where $f_i^\star(\tilde{y}) = \sup_{x_i} \big\{\tilde{y}^\mathsf{T} x_i - f_i(x_i)\big\}$ denotes the Fenchel conjugate of the local objective function of robot $i$, which highlights the separable structure of the dual optimization problem.

For a distributed approach to solving (24), we assign local dual variables to each robot, with an equality constraint ensuring that all robots compute the same dual variables. With this approach, robot $i$ maintains only its local dual variable $\tilde{y}_i$, a copy of $\tilde{y}$. We express (24) in its distributed form as

$$\begin{aligned} & \underset{\tilde{\boldsymbol{y}}}{\text{maximize}} \ \sum_{i=1}^{N} \ (-f_i^\star(-A_i^\mathsf{T}\tilde{y}_i) - \tilde{y}_i^\mathsf{T} b_i) \\ & \text{subject to} \ \tilde{y}_i \geq 0 \quad \forall i \in \mathcal{V} \\ & \qquad \tilde{y}_i = \tilde{y}_j \quad \forall j \in \mathcal{N}_i, \ \forall i \in \mathcal{V} \end{aligned} \tag{25}$$

with the equality constraints enforced between neighboring robots, as defined by the communication graph $\mathcal{G}$. We denote the vertical concatenation of the dual variables of all robots as $\tilde{\boldsymbol{y}} = [\tilde{y}_i^\mathsf{T}, \ \forall i \in \mathcal{V}]^\mathsf{T}$.

**Proposition 2.** *The dual optimization problem in* (25) *has the same optimal solution and optimal objective value as the problem in* (24)*.*

*Proof.* The proof of equivalence between both problems follows along the same lines as in Proposition (1). □

Following the same procedure utilized in the Section VI-A, we derive a method for solving (25) where each robot computes its dual variables locally. We introduce local slack variables to allow for local computations by each robot, with the augmented Lagrangian of (25) given by

$$\begin{aligned}\mathcal{L}_a^d(\tilde{\boldsymbol{y}},\gamma,\zeta,p,w) = &\sum_{i=1}^{N}(-f_i^\star(-A_i^\mathsf{T}\tilde{y}_i) - \tilde{y}_i^\mathsf{T} b_i) \\ &- \sum_{(i,j)\in\mathcal{E}}\Big(p_{ij}^\mathsf{T}(\tilde{y}_i-\gamma_{ij}) + w_{ij}^\mathsf{T}(\tilde{y}_j-\zeta_{ij})\Big) \\ &- \frac{\rho}{2}\sum_{(i,j)\in\mathcal{E}}\Big(\|\tilde{y}_i-\gamma_{ij}\|_2^2 + \|\tilde{y}_j-\zeta_{ij}\|_2^2\Big)\end{aligned} \tag{26}$$

with the Lagrange multipliers $p_{ij} \in \mathbb{R}^e$ and $w_{ij} \in \mathbb{R}^e$ and slack variables $\gamma_{ij} \in \mathbb{R}^e$ and $\zeta_{ij} \in \mathbb{R}^e$. Likewise, we enforce that $\tilde{y}_i$ satisfies the affine inequality constraint in (25) for non-negative dual variables and the constraint $\gamma_{ij} = \zeta_{ij}$.

Robot $i$ computes $\tilde{y}_i$ from the optimization problem

$$\begin{aligned}\underset{\tilde{y}_i}{\text{maximize}}\ &\left\{-f_i^\star(-A_i^\mathsf{T}\tilde{y}_i) - \tilde{y}_i^\mathsf{T} b_i - \sum_{j\in\mathcal{N}_i}\left(p_{ij}^k + w_{ji}^k\right)^\mathsf{T}\tilde{y}_i \right. \\ &\left. - \rho\sum_{j\in\mathcal{N}_i}\left\|\tilde{y}_i - \frac{\tilde{y}_i^k+\tilde{y}_j^k}{2}\right\|_2^2\right\} \\ &\text{subject to}\ \ \tilde{y}_i \geq 0\end{aligned} \tag{27}$$

which simplifies to

$$\begin{aligned}\underset{\tilde{y}_i}{\text{maximize}}\ &\left\{-f_i^\star(-A_i^\mathsf{T}\tilde{y}_i) - \tilde{y}_i^\mathsf{T} b_i - \tilde{r}_i^{k\mathsf{T}}\tilde{y}_i \right. \\ &\left. - \rho\sum_{j\in\mathcal{N}_i}\left\|\tilde{y}_i - \frac{\tilde{y}_i^k+\tilde{y}_j^k}{2}\right\|_2^2\right\} \\ &\text{subject to}\ \ \tilde{y}_i \geq 0\end{aligned} \tag{28}$$

with the update procedure for $\tilde{r}_i$ given by

$$\tilde{r}_i^{k+1} = \tilde{r}_i^k + \rho\sum_{j\in\mathcal{N}_i}\left(\tilde{y}_i^{k+1} - \tilde{y}_j^{k+1}\right) \tag{29}$$

at iteration $k$.

We can further simplify the update procedures for the local dual variable in (28) by recognizing that the equality constraint in (4) corresponds to an unrestricted dual variable. Consequently, the update procedure in (28) simplifies to

$$\begin{aligned}\underset{y_i,\lambda_i}{\text{maximize}}\ &\left\{-f_i^\star(y_i-\lambda_i\mathbf{1}_m) + \frac{1}{N}\mathbf{1}_m^\mathsf{T} y_i - \lambda_i - r_i^{k\mathsf{T}} y_i \right. \\ &\left. - \rho\sum_{j\in\mathcal{N}_i}\left\|y_i - \frac{y_i^k+y_j^k}{2}\right\|_2^2\right\} \\ &\text{subject to}\ \ y_i \geq 0\end{aligned} \tag{30}$$

where $y_i \in \mathbb{R}^m$ denotes robot $i$'s dual variable for the first inequality constraint in (4) and $\lambda_i \in \mathbb{R}$ denotes its dual variable for the equality constraint, with the associated update procedure for $r_i \in \mathbb{R}^m$ given by

$$r_i^{k+1} = r_i^k + \rho\sum_{j\in\mathcal{N}_i}\left(y_i^{k+1} - y_j^{k+1}\right) \tag{31}$$

at iteration $k$.

**Remark 1.** *In problems where the objective function $f_i(x_i)$ of robot $i$ in* (4) *consists of a linear function, the Fenchel conjugate $f_i^\star(y_i - \lambda_i\mathbf{1}_m)$ has a closed-form solution when*

$$c_i - y_i + \lambda_i\mathbf{1}_m \geq 0\ , \tag{32}$$

*with an optimal value of zero. In these problems, robot $i$ updates $y_i$ and $\lambda_i$ as the solution of the optimization problem*

$$\begin{aligned}\underset{y_i,\lambda_i}{\text{maximize}}\ &\left\{\frac{1}{N}\mathbf{1}_m^\mathsf{T} y_i - \lambda_i - r_i^{k\mathsf{T}} y_i \right. \\ &\left. - \rho\sum_{j\in\mathcal{N}_i}\left\|y_i - \frac{y_i^k+y_j^k}{2}\right\|_2^2\right\} \\ &\text{subject to}\ \ c_i - y_i + \lambda_i\mathbf{1}_m \geq \mathbf{0} \\ &\qquad\qquad y_i \geq 0\end{aligned} \tag{33}$$

*at iteration $k$.*

The dual optimization problem of robot $i$ in (30) does not involve the local constraints of other robots, noting that each robot only knows a subset of the problem constraints in (25). However, the equality constraints on the local dual variables in (25) ensures that the dual variable of each robot satisfies all the problem constraints.

**Theorem 2** (Convergence of $\{\boldsymbol{y}^k\}$)**.** *The sequence of local dual variables $\{y_i^k\}$ of robot $i$ converges to the optimal dual solution $y^\star,\ \ \forall i \in \mathcal{V}$.*

*Proof.* The proof follows along the same lines as the proof of Theorem 1. For completeness, we provide the proof in the Appendix. □

Upon convergence of the dual variables, each robot computes its optimal primal solution from

$$\begin{aligned}\underset{x_i}{\text{minimize}}\ \underset{y_i}{\text{maximize}}\ &\left\{f_i(x_i) + y_i^\mathsf{T}\left(-x_i + \frac{1}{N}\mathbf{1}_m\right) - r_i^{\star\mathsf{T}} y_i \right. \\ &\left. - \rho\sum_{j\in\mathcal{N}_i}\left\|y_i - \frac{\bar{y}_i+\bar{y}_j}{2}\right\|_2^2\right\} \\ &\text{subject to}\ \ y_i \geq 0 \\ &\qquad\qquad \mathbf{1}_m^\mathsf{T} x_i = 1 \\ &\qquad\qquad 0 \leq x_i \leq 1\end{aligned} \tag{34}$$

where $\bar{y}_i$ denotes the local dual variable of robot $i$ computed at the last iteration of our distributed algorithm. The problem in (34) simplifies to

$$\begin{aligned} \underset{x_i}{\text{minimize}} \ \underset{y_i}{\text{maximize}} \ & \Big\{ f_i(x_i) - \rho|\mathcal{N}_i| \, \|y_i - \nu_i^\star(x_i)\|_2^2 \\ & \quad + \rho|\mathcal{N}_i| \, \|\nu_i^\star(x_i)\| \\ & \quad - \frac{\rho}{4} \sum_{j \in \mathcal{N}_i} \|\bar{y}_i + \bar{y}_j\|_2^2 \Big\} \\ \text{subject to} \ & \ y_i \geq 0 \\ & \ \mathbf{1}_m^\mathsf{T} x_i = 1 \\ & \ 0 \leq x_i \leq 1 \end{aligned} \tag{35}$$

where

$$\nu_i^\star(x_i) = \frac{1}{2\rho|\mathcal{N}_i|} \left( -x_i + \frac{1}{N}\mathbf{1}_m - r_i^\star + \rho \sum_{j \in \mathcal{N}_i} (\bar{y}_i + \bar{y}_j) \right) \tag{36}$$

which results in a closed-form solution for $y_i$, given by

$$y_i^\star(x_i) = \max\left(0, \nu_i^\star(x_i)\right) \ , \tag{37}$$

with the resulting minimization problem

$$\begin{aligned} \underset{x_i}{\text{minimize}} \ & \Big\{ f_i(x_i) - \rho|\mathcal{N}_i| \, \|\min\left(0, \nu_i^\star(x_i)\right)\|_2^2 \\ & \quad + \rho|\mathcal{N}_i| \, \|\nu_i^\star(x_i)\|_2^2 \Big\} \\ \text{subject to} \ & \ \mathbf{1}_m^\mathsf{T} x_i = 1 \\ & \ 0 \leq x_i \leq 1 \end{aligned} \tag{38}$$

for $x_i$. After further simplifying the problem in (38), robot $i$ computes $x_i$ from the optimization problem

$$\begin{aligned} \underset{x_i}{\text{minimize}} \ & \Big\{ f_i(x_i) + \frac{\rho|\mathcal{N}_i|}{2} \, \|\nu_i^\star(x_i)\}\|_2^2 \\ & \quad + \frac{\rho|\mathcal{N}_i|}{2} |\nu_i^\star(x_i)|^\mathsf{T} \nu_i^\star(x_i) \Big\} \\ \text{subject to} \ & \ \mathbf{1}_m^\mathsf{T} x_i = 1 \\ & \ 0 \leq x_i \leq 1 \end{aligned} \tag{39}$$

after the last iteration of our distributed algorithm, as determined by a specified stopping criterion.

Algorithm 2 outlines our distributed dual algorithm for task assignment problems. Each robot does not maintain the slack variables $\gamma$ and $\zeta$ and the Lagrange multipliers $p$ and $w$. Further, the updates in (30) only require each robot to communicate its dual variables with its neighbors, without communicating its task assignment, minimizing the amount of potentially sensitive data exchanged by each robot. MURD-TAP does not require a feasible initialization of the local dual variables of each robot.

**Theorem 3.** *The primal variable of all robots $x^S = [x_1^\mathsf{T}, \cdots, x_N^\mathsf{T}]^\mathsf{T}$, computed in* (39)*, corresponds to the optimal solution $x^\star$ of* (18)*.*

*Proof.* We provide the proof in the Appendix. □

**Algorithm 2:** Distributed Multi-Robot Dual Algorithm for Task Assignment Problems (MURD-TAP)

Initialization:
$y_i^0 \in \mathbb{R}^m$, $\lambda_i^0 \in \mathbb{R} \quad \forall i \in \mathcal{V}$
$r_i^0 \leftarrow 0 \quad \forall i \in \mathcal{V}$
$k \leftarrow 0$
**do in parallel** $\forall i \in \mathcal{V}$
  $\left(y_i^{k+1}, \lambda_i^{k+1}\right) \leftarrow$ Procedure (30)
  $r_i^{k+1} \leftarrow$ Procedure (31)
  $k \leftarrow k + 1$
**while** *not converged or stopping criterion is not met*;
Recovery:
$x_i \leftarrow$ Procedure (39) $\quad \forall i \in \mathcal{V}$

### C. Distributed Inexact Dual Algorithm

In the MUR-TAP and MURD-TAP algorithms, each robot solves the optimization problem arising in its primal update procedure through a nested iterative method to update its local variables. In problems where each robot has limited access to computational resources, solving these optimization problems may prove challenging for each robot. Consequently, we derive a distributed algorithm with simpler update procedures for task assignment problems. We express the problem in (4) as

$$\begin{aligned} \underset{x}{\text{minimize}} \ & \sum_{i=1}^N f_i(x_i) \\ \text{subject to} \ & \sum_{i=1}^N \left( \frac{1}{N}\mathbf{1}_m - x_i \right) \leq 0 \\ & \sum_{i=1}^N \left( G_i x_i - \frac{1}{N}\mathbf{1}_N \right) = 0 \\ & 0 \leq x_i \leq 1 \quad \forall i \in \mathcal{V} \end{aligned} \tag{40}$$

where $G_i \in \mathbb{R}^{N \times m}$. We set the $i$-th row of $G_i$ to $\mathbf{1}_m^\mathsf{T}$ with all other entries set to zero.

We express the Lagrangian of (40) as

$$\begin{aligned} \mathcal{L}_p(x, y, \lambda) = & \sum_{i=1}^N f_i(x_i) + y^\mathsf{T} \sum_{i=1}^N \left( \frac{1}{N}\mathbf{1}_m - x_i \right) \\ & + \lambda^\mathsf{T} \sum_{i=1}^N \left( G_i x_i - \frac{1}{N}\mathbf{1}_N \right) \end{aligned} \tag{41}$$

with dual variables $y \in \mathbb{R}^m$ and $\lambda \in \mathbb{R}^N$ where $y \geq 0$, without relaxing the box constraints on $x$. From (41), we obtain the dual problem

$$\begin{aligned} \underset{y, \lambda}{\text{maximize}} \ & \sum_{i=1}^N \phi_i(y, \lambda) \\ \text{subject to} \ & y \geq 0 \end{aligned} \tag{42}$$

where

$$\begin{aligned} \phi_i(y, \lambda) = \inf_{x_i} \Big\{ & f_i(x_i) + y^\mathsf{T} \left( \frac{1}{N}\mathbf{1}_m - x_i \right) \\ & + \lambda^\mathsf{T} \left( G_i x_i - \frac{1}{N}\mathbf{1}_N \right) \Big\} \end{aligned} \tag{43}$$

with $0 \leq x_i \leq 1$. To derive a distributed method for solving (42), we assign local copies of the dual variables $y$ and $\lambda$ to each robot, with the resulting optimization problem given by

$$\begin{aligned} \underset{\boldsymbol{y},\boldsymbol{\lambda}}{\text{maximize}} \ & \sum_{i=1}^{N} \ \phi_i(y_i, \lambda_i) \\ \text{subject to } & y_i \geq 0 \quad \forall i \in \mathcal{V} \\ & y_i = y_j, \ \ \lambda_i = \lambda_j \quad \forall j \in \mathcal{N}_i, \ \forall i \in \mathcal{V} \end{aligned} \tag{44}$$

where $\boldsymbol{y} = [y_i^\mathsf{T}, \ \forall i \in \mathcal{V}]^\mathsf{T}$ and $\boldsymbol{\lambda} = [\lambda_i^\mathsf{T}, \ \forall i \in \mathcal{V}]^\mathsf{T}$. Following a similar proof to that of Proposition 2, the optimization problems in (42) and (44) have the same optimal solution and optimal objective value.

Taking the same approach employed in Section VI-B, we derive a distributed algorithm for solving the optimization problem in (44). With this approach, robot $i$ computes its local variables $y_i$ and $\lambda_i$ from

$$\begin{aligned} \underset{y_i,\lambda_i}{\text{maximize}} \ & \Bigg\{ \phi_i(y_i, \lambda_i) - \eta_i^{k\mathsf{T}} y_i - \psi_i^{k\mathsf{T}} \lambda_i \\ & \quad - \rho \sum_{j \in \mathcal{N}_i} \left\| y_i - \frac{y_i^k + y_j^k}{2} \right\|_2^2 \\ & \quad - \rho \sum_{j \in \mathcal{N}_i} \left\| \lambda_i - \frac{\lambda_i^k + \lambda_j^k}{2} \right\|_2^2 \Bigg\} \\ \text{subject to } & \ y_i \geq 0 \end{aligned} \tag{45}$$

where $\eta_i \in \mathbb{R}^m$ and $\psi_i \in \mathbb{R}^N$ denote Lagrange multipliers for the equality constraints in (44). The optimization problem in (45) represents a min-max optimization problem, which in general can be difficult to solve. However, we leverage the existence of a saddle-point for the Lagrangian of (40) to compute a solution for (45). We express the optimization problem in (45) as

$$\begin{aligned} \underset{y_i,\lambda_i}{\text{maximize}} \ \underset{x_i}{\text{minimize}} \ & \Bigg\{ f_i(x_i) + y_i^\mathsf{T} \left( \frac{1}{N} \mathbf{1}_m - x_i \right) \\ & \quad + \lambda_i^\mathsf{T} \left( G_i x_i - \frac{1}{N} \mathbf{1}_N \right) \\ & \quad - \eta_i^{k\mathsf{T}} y_i - \psi_i^{k\mathsf{T}} \lambda_i \\ & \quad - \rho \sum_{j \in \mathcal{N}_i} \left\| y_i - \frac{y_i^k + y_j^k}{2} \right\|_2^2 \\ & \quad - \rho \sum_{j \in \mathcal{N}_i} \left\| \lambda_i - \frac{\lambda_i^k + \lambda_j^k}{2} \right\|_2^2 \Bigg\} \\ \text{subject to } & \ 0 \leq x_i \leq 1 \\ & \ y_i \geq 0 \end{aligned} \tag{46}$$

which simplifies to

$$\begin{aligned} \underset{y_i,\lambda_i}{\text{maximize}} \ \underset{x_i}{\text{minimize}} \ & \Big\{ f_i(x_i) - \rho |\mathcal{N}_i| \left\| y_i - \nu_i^k(x_i) \right\|_2^2 \\ & \quad + \rho |\mathcal{N}_i| \left\| \nu_i^k(x_i) \right\|_2^2 \\ & \quad - \rho |\mathcal{N}_i| \left\| \lambda_i - \ell_i^k(x_i) \right\|_2^2 \\ & \quad + \rho |\mathcal{N}_i| \left\| \ell_i^k(x_i) \right\|_2^2 + \theta_i^k \Big\} \\ \text{subject to } & \ 0 \leq x_i \leq 1 \\ & \ y_i \geq 0 \end{aligned} \tag{47}$$

where

$$\begin{aligned} \nu_i^k(x_i) &= \frac{1}{2\rho|\mathcal{N}_i|} \left( \frac{1}{N} \mathbf{1}_m - x_i - \eta_i^k + \rho \sum_{j \in \mathcal{N}_i} \left( y_i^k + y_j^k \right) \right), \\ \ell_i^k(x_i) &= \frac{1}{2\rho|\mathcal{N}_i|} \left( G_i x_i - \frac{1}{N} \mathbf{1}_N - \psi_i^k + \rho \sum_{j \in \mathcal{N}_i} \left( \lambda_i^k + \lambda_j^k \right) \right), \end{aligned} \tag{48}$$

and $\theta_i^k$ are independent of the dual variable $y_i$. The existence of a saddle-point enables us to swap the order of the optimization problems to solve (47), yielding a closed-form solution for $y_i$ with

$$\begin{aligned} y_i^{k+1} &= \max\left(0, \nu_i^k\left(x_i^{k+1}\right)\right) \\ \lambda_i^{k+1} &= \ell_i^k\left(x_i^{k+1}\right) \end{aligned} \tag{49}$$

where the max operator works element-wise [50]. By swapping the order of the optimization problems, the minimization problem for $x_i$ simplifies to

$$\begin{aligned} \underset{x_i}{\text{minimize}} \ & \Big\{ f_i(x_i) - \rho |\mathcal{N}_i| \left\| \min\left(0, \nu_i^k(x_i)\right) \right\|_2^2 \\ & \quad + \rho |\mathcal{N}_i| \left\| \nu_i^k(x_i) \right\|_2^2 + \rho |\mathcal{N}_i| \left\| \ell_i^k(x_i) \right\|_2^2 \Big\} \\ \text{subject to } & \ 0 \leq x_i \leq 1 \end{aligned} \tag{50}$$

which represents a nonlinear optimization problem. We note that the optimization problem in (50) might be challenging to solve in some situations. Hence, we utilize a proximal gradient update scheme for solving the problem in (50). We take a first-order approximation of the objective function of (50) at $x_i^k$ and solve the resulting optimization problem

$$\begin{aligned} \underset{x_i}{\text{minimize}} \ & \Big\{ \nabla f_i\left(x_i^k\right)^\mathsf{T} x_i - \kappa_i\left(x_i^k\right)^\mathsf{T} x_i - \nu_i^k\left(x_i^k\right)^\mathsf{T} x_i \\ & \quad + \ell_i^k\left(x_i^k\right)^\mathsf{T} G_i x_i + \frac{1}{2\beta_i} \left\| x_i - x_i^k \right\|_2^2 \Big\} \end{aligned} \tag{51}$$

where

$$\kappa_i\left(x_i^k\right) = -\min\left(0, \nu_i^k(x_i^k)\right) \ , \tag{52}$$

which yields the closed-form solution

$$\hat{x}_i^k = x_i^k - \beta_i \left( \nabla f_i\left(x_i^k\right) - \kappa_i\left(x_i^k\right) - \nu_i^k\left(x_i^k\right) + G_i^\mathsf{T} \ell_i^k\left(x_i^k\right) \right), \tag{53}$$

equivalent to the gradient descent update of $x_i$ with step-size $\beta_i$ at iteration $k$. Subsequently, we take a proximal projection of the solution in (53) using the proximal operator $\rho_i\left(\hat{x}_i^k\right)$ defined as the solution to the optimization problem

$$\begin{aligned} \underset{x_i}{\text{minimize}} \ & \left\| x_i - \hat{x}_i^k \right\|_2^2 \\ \text{subject to } & \ 0 \leq x_i \leq 1 \end{aligned} \tag{54}$$

which admits a closed-form solution. Consequently, robot $i$ updates $x_i$ at iteration $k$ using the closed-form procedure

$$x_i^{k+1} = \min\big(1, \max\big(0, \hat{x}_i^k\big)\big) \tag{55}$$

before computing $y_i$ and $\lambda_i$ using (49). In addition, robot $i$ updates $\eta_i$ and $\psi_i$ using

$$\begin{aligned} \eta_i^{k+1} &= \eta_i^k + \rho \sum_{j \in \mathcal{N}_i} \big(y_i^{k+1} - y_j^{k+1}\big) \\ \psi_i^{k+1} &= \psi_i^k + \rho \sum_{j \in \mathcal{N}_i} \big(\lambda_i^{k+1} - \lambda_j^{k+1}\big) \end{aligned} \tag{56}$$

at iteration $k$.

We outline our distributed inexact dual algorithm for the multi-robot task assignment problem in Algorithm 3. Note that our distributed algorithm does not require each robot to utilize a nested iterative method to update their local variables. Rather, all the robots update their local variables using closed-form solutions, which involve arithmetic operations.

**Algorithm 3:** Distributed Multi-Robot Inexact Dual Algorithm for Task Assignment Problems (MURID-TAP)

Initialization:
$x_i^0 \in \mathbb{R}^m$, $y_i^0 \in \mathbb{R}^m$, $\lambda_i^0 \in \mathbb{R}^N \quad \forall i \in \mathcal{V}$
$\eta_i^0 \leftarrow 0$, $\psi_i^0 \leftarrow 0 \quad \forall i \in \mathcal{V}$
$k \leftarrow 0$
**do in parallel** $\forall i \in \mathcal{V}$
  $x_i^{k+1} \leftarrow$ Procedure (55)
  $\big(y_i^{k+1}, \lambda_i^{k+1}\big) \leftarrow$ Procedure (49)
  $\big(\eta_i^{k+1}, \psi_i^{k+1}\big) \leftarrow$ Procedure (56)
  $k \leftarrow k + 1$
**while** *not converged or stopping criterion is not met*;

**Theorem 4** (Convergence of $(x^k, \boldsymbol{y}^k, \boldsymbol{\lambda}^k)$)**.** *The iterates $(x_i^k, y_i^k, \lambda_i^k)$ of robot $i$ converge to an optimal primal-dual solution pair $(x_i^\star, y_i^\star, \lambda_i^\star)$ of* (40)*, $\forall i \in \mathcal{V}$.*

*Proof.* Refer to [51] for the proof. □

### *D. Algorithm Selection*

Our algorithms apply to a variety of task assignment problems, given by (4), including the classical task assignment problem with linear objective functions. Selection of the most efficient algorithm for a given multi-robot task assignment problem depends on the relative availability of computation and communication resources at each robot, and the privacy requirements in the given problem. To guide selection of an efficient algorithm, we examine the computational, storage, and communication requirements of each algorithm.

*1) Computational Complexity:* The MUR-TAP algorithm requires each robot to solve the constrained optimization problem in (15) to compute its task assignment. The optimization problem can be solved using an interior-point method, which involves factorizing the matrix associated with the Karush-Kuhn-Tucker (KKT) necessary conditions for optimality — the KKT matrix. Assuming the KKT matrix is positive definite, factorizing the matrix can be performed through Cholesky decomposition at a cost of $O((Nm)^3)$ floating-point operations (FLOPS), where we have retained the dominant terms in quantifying the number of floating-point operations required. In addition, robot $i$ updates its local Lagrange multiplier $q_i$ in the MUR-TAP algorithm using $O(Nm)$ FLOPS. Hence, the MUR-TAP algorithm has a net computational complexity of $O((Nm)^3)$ FLOPS at each iteration.

In our distributed exact dual algorithm, MURD-TAP, each robot updates its dual variables by solving the optimization problem in (30). The optimization problem can be solved using an interior-point method, which requires $O(m^3)$ FLOPS to factorize the associated KKT matrix. Further, each robot updates its Lagrange multiplier using (31) with $O(m)$ FLOPS. Consequently, the MURD-TAP algorithm has a net computational complexity of $O(m^3)$ FLOPS at each iteration.

MURID-TAP does not require a nested iterative method for any optimization problem. Each robot updates its local variables using the closed-form procedures in (55), (49), and (56). As a result, robot $i$ requires $O(m + N)$ FLOPS to update its local variables at each iteration.

*2) Communication Complexity:* In the MUR-TAP algorithm, each robot shares its local primal variable with its neighbors at each iteration. As such, each robot transmits $O(Nm)$ bits of information to its neighbors and, likewise, receives $O(Nm)$ bits of information. The MURD-TAP algorithm requires each robot to communicate its local dual variables to its neighbors. Hence, the MURD-TAP algorithm has a communication complexity of $O(m)$ bits. Similarly, the MURID-TAP algorithm requires each robot to share its local dual variables with its neighbors, resulting in a communication complexity of $O(m + N)$ bits.

*3) Data Storage Complexity:* Each robot maintains a local primal variable and a local Lagrange multiplier for the equality constraints between its primal solution and that of its neighbors in the MUR-TAP algorithm. We assume that each robot represents its optimization variables using $\mathcal{Q}$ bits. When these variables are represented as double precision floating-point numbers, $\mathcal{Q} = 64$ bits. With this assumption, each robot requires $2Nm\mathcal{Q}$ bits for storing its local variables in the MUR-TAP algorithm. In the MURD-TAP algorithm, each robot maintains a local dual variable and a Lagrange multiplier, which require $(2m + 1)\mathcal{Q}$ bits for data storage. The primal solution can be recovered upon termination of the MURD-TAP algorithm. In contrast, the MURID-TAP algorithm requires each robot to maintain a local primal and dual variable and Lagrange multipliers for the equality constraints between its dual variable and that of its neighbors. As a result, in the MURID-TAP algorithm, robot $i$ requires $(3m + 2N)\mathcal{Q}$ bits for storing its local variables.

*4) Selection Guide:* We summarize the results of the complexity analysis in Table I. The computational complexity analysis reveals that the per-iteration computational complexity of the MUR-TAP algorithm scales cubically the number of robots $N$ and the number of tasks $m$ in the task assignment problem in (4), while the per-iteration computational complexity of the MURD-TAP algorithm scales

TABLE I
COMPLEXITY OF THE MULTI-ROBOT TASK ASSIGNMENT ALGORITHMS PER ITERATION. COMPUTATIONAL AND COMMUNICATION COMPLEXITY IS MEASURED IN FLOPS, WHILE DATA STORAGE COMPLEXITY IS MEASURED IN BITS. WE DENOTE THE NUMBER OF BITS REQUIRED TO REPRESENT EACH COMPONENT OF THE OPTIMIZATION VARIABLE AS $\mathcal{Q}$.

| Algorithm | Computation | Communication | Data Storage |
|---|---|---|---|
| MUR-TAP | $O((Nm)^3)$ | $O(Nm)$ | $2(Nm)\mathcal{Q}$ |
| MURD-TAP | $O(m^3)$ | $O(m)$ | $(2m+1)\mathcal{Q}$ |
| MURID-TAP | $O(m+N)$ | $O(m+N)$ | $(3m+2N)\mathcal{Q}$ |

cubically in the number of tasks $m$ only. In contrast, the MURID-TAP algorithm provides the most efficient per-iteration computational complexity, scaling linearly in the sum of the number of robots and tasks $(m+N)$. Consequently, the MURID-TAP algorithm provides an efficient method for solving multi-robot problems when access to adequate computation resources poses a limiting constraint (and in problems with a large number of robots/tasks). However, when the availability of computation resources does not prove prohibitive, the MUR-TAP and MURD-TAP algorithms should be considered, as, generally, the MURID-TAP algorithm requires a greater number of iterations for convergence, given its inexact update procedures.

In problems with low communication bandwidth and situations where communication between robots comes at a premium, the communication complexity of the algorithms plays a critical role in selecting an efficient distributed algorithm. MURD-TAP provides the lowest communication complexity, independent of the number of robots in the problem; however, MURID-TAP also scales efficiently with respect to its communication complexity, scaling linearly in the sum of the number of robots and tasks.

In situations with limited local data storage resources, MURD-TAP and MURID-MP algorithms provide more efficient methods for solving multi-robot problems. However, MURID-TAP scales linearly in the number of robots, which can be consequential in problems with a large number of robots. In contrast, the data storage complexity of MURD-TAP does not depend on the number of robots. In general, selection of an efficient algorithm depends on the relative computation, communication, and data storage resources available to each robot in the multi-robot problem.

## VII. SIMULATIONS

In this section, we examine the performance of our distributed algorithms in the multi-robot task assignment problem. We compare our methods to other distributed methods in each of these problems, assessing the convergence rates of each method to the optimal task assignment, computed centrally after collating all the problem data. We begin with the task assignment problem with linear objective functions and an equal number of robots and tasks, which is amenable to other distributed task assignment algorithms. Thereafter, we consider the multi-robot task assignment problem with nonlinear objective functions and an unequal number of robots and tasks, a problem which is unsuitable for many distributed task assignment algorithms. Lastly, we apply our algorithms to the multi-robot persistent surveillance problem, where we consider periodic assignments of robots to aerial surveillance stations over time, with constraints on the minimum allowable capacity level of each robot's battery. We execute all the algorithms on a laptop computer with an Intel i7 processor with 16 GB RAM and use the interior-point methods for quadratic programming available in Gurobi [52].

### A. Linear Multi-Robot Task Assignment Problem

We consider the multi-robot task assignment problem in (2) where the objective function $f(x)$ consists of a sum of linear functions, given by

$$f(x) = \sum_{i=1}^{N} c_i^{\mathsf{T}} x_i \tag{57}$$

with $x_i \in \mathbb{R}^m$ denoting the optimization variable of robot $i$ and $x = [x_1^{\mathsf{T}}, \cdots, x_N^{\mathsf{T}}]^{\mathsf{T}} \in \mathbb{R}^{Nm}$. We examine the convergence rate of our algorithms — MUR-TAP (primal algorithm), MURD-TAP (exact dual algorithm), and MURID-TAP (inexact dual algorithm) — to the optimal task assignment, assessing the convergence of each algorithm in terms of the percentage relative error with respect to the optimal solution of the problem. We define the relative error with respect to the optimal task assignment as

$$RE(x) = \frac{\|x - x^\star\|_2}{\|x^\star\|_2} \tag{58}$$

where $x^\star$ denotes the optimal task assignment. In each task assignment problem, we randomly generate the vectors in the objective function.

*1) Convergence to the Optimal Solution:* We examine the convergence rate of each algorithm to the optimal solution on randomly-generated connected communication networks. We begin with a task assignment problem with 5 robots and 5 tasks. In Table II, we provide the number of optimization variables maintained by each robot and the mean and standard deviation of the number of communication rounds and total computation time in milliseconds required by each robot in these algorithms to achieve a relative error of at most $1\mathrm{e}^{-11}\%$ across 60 problems.

TABLE II
NUMBER OF OPTIMIZATION VARIABLES (# OF VAR.) AND THE MEAN AND STANDARD DEVIATION OF THE NUMBER OF COMMUNICATION ROUNDS (# OF COMM.) AND TOTAL COMPUTATION TIME (COMP. TIME) IN MILLISECONDS, PER ROBOT, IN 60 TASK ASSIGNMENT PROBLEMS WITH 5 ROBOTS AND 5 TASKS.

| Algorithm | # of Var. | # of Comm. | Comp. Time (msec) |
|---|---|---|---|
| MUR-TAP | 25 | $25 \pm 14$ | $4.164 \pm 2.306$ |
| MURD-TAP | 6 | $31 \pm 17$ | $3.730 \pm 1.934$ |
| MURID-TAP | 15 | $45 \pm 20$ | $1.608 \pm 0.641$ |

From Table II, each robot in MURD-TAP optimizes over 6 variables, while each robot in MUR-TAP and MURID-TAP optimizes over 25 variables and 15 variables respectively. Consequently, the MURD-TAP algorithm requires the least data storage capacity, which could be essential in robots with limited

onboard storage. Although each robot in MUR-TAP optimizes over the greatest number of optimization variables compared to the other algorithms, each robot requires the fewest number of communication rounds to compute the optimal task assignment. In MUR-TAP, each robot computes the optimal task assignment within a mean of 25 communication rounds, while each robot in MURD-TAP and MURID-TAP requires a mean of 31 and 45 communication rounds respectively. However, the relatively larger size of the optimization problem contributes to the longer computation time required by MUR-TAP for convergence, with each robot obtaining its optimal assignment after a mean computation time of 4.164 msec. With MURID-TAP, each robot takes the shortest mean computation time of about 1.608 msec to obtain the optimal task assignment; however, MURID-TAP requires the greatest number of communication rounds for convergence to the optimal task assignment. MURD-TAP requires a shorter mean computation time of 3.730 msec to converge to the optimal solution compared to MUR-TAP, but convergence of MURD-TAP is attained after a greater number of communication rounds.

In Figure 1, we show the relative error of the local variables of all robots with the MUR-TAP, MURD-TAP, and MURID-TAP algorithms during one trial. MUR-TAP attains the fastest convergence rate, with each robot computing an optimal task assignment within 35 communication rounds. In contrast, with MURD-TAP, each robot requires about 40 communication rounds to compute its optimal task assignment. MURID-TAP achieves the slowest convergence rate among the three algorithms, requiring about 50 communication rounds for convergence.

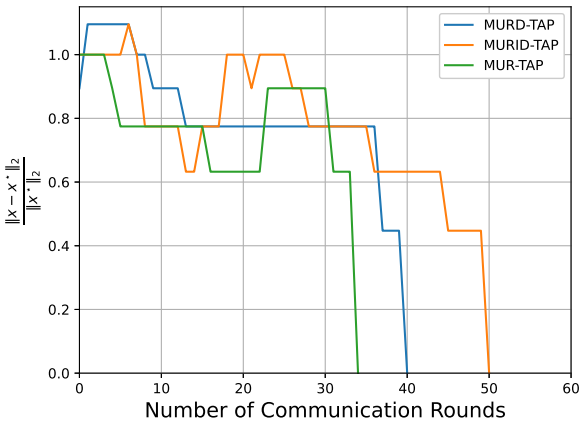


Fig. 1. Relative error of MUR-TAP, MURD-TAP, and MURID-TAP on the multi-robot task assignment problem with 5 robots and 5 tasks. MUR-TAP provides the fastest convergence rate, converging within 35 communication rounds. With MURD-TAP, the relative error of the local variables of all robots converges to zero faster than that of MURID-TAP.

Next, we examine the multi-robot task assignment problem with 10 robots and 10 tasks. Table III provides the number of optimization variables maintained by each robot and the mean and standard deviation of the number of communication rounds and cumulative computation time in milliseconds required by each robot to achieve a relative error of at most $1e^{-11}\%$ across 60 problems.

TABLE III
NUMBER OF OPTIMIZATION VARIABLES (# OF VAR.) AND THE MEAN AND STANDARD DEVIATION OF THE NUMBER OF COMMUNICATION ROUNDS (# OF COMM.) AND CUMULATIVE COMPUTATION TIME (COMP. TIME) IN MILLISECONDS, PER ROBOT, IN 60 TASK ASSIGNMENT PROBLEMS WITH 10 ROBOTS AND 10 TASKS.

| Algorithm | # of Var. | # of Comm. | Comp. Time (msec) |
|---|---|---|---|
| MUR-TAP | 100 | $85 \pm 38$ | $29.421 \pm 12.832$ |
| MURD-TAP | 11 | $39 \pm 22$ | $5.68 \pm 3.055$ |
| MURID-TAP | 30 | $87 \pm 38$ | $1.526 \pm 0.603$ |

As in the problem with 5 robots, MUR-TAP requires the greatest number of optimization variables, requiring each robot to maintain 100 variables, which results in a notable increase in the mean computation time per robot. In MUR-TAP, each robot takes a mean computation time of over 29 msec. In contrast, each robot in MURD-TAP maintains 11 variables, the smallest number of optimization variables maintained by each robot across all the algorithms. In addition, MURD-TAP requires the fewest number of communication rounds for convergence, with each robot computing the optimal task assignment within a mean of 39 communication rounds. Although each robot in MURID-TAP requires the greatest number of communication rounds to compute the optimal solution, MURID-TAP offers the shortest mean computation time of about 1.526 msec, compared to a mean computation time of 5.68 msec required by MURD-TAP. In MURID-TAP, each robot does not utilize a nested iterative method to update its local variables; rather, each robot updates its local variables using simple closed-form solutions, contributing to the shorter cumulative computation time required by MURID-TAP.

Figure 2 shows the relative error of the task assignment of all robots in MUR-TAP, MURD-TAP, and MURID-TAP to an optimal task assignment in the problem with 10 robots and 10 tasks during one trial. In all the algorithms, each robot requires a greater number of communication rounds to compute its optimal task assignment compared to the number of communication rounds required in the task assignment problem with 5 robots and 5 tasks. In this trial, MUR-TAP achieves a slightly faster convergence rate compared to MURID-TAP, with each robot in MUR-TAP computing its optimal task assignment within 69 communication rounds, while MURID-TAP requires about 70 communication rounds for convergence. In MURD-TAP, each robot computes its optimal task assignment within 45 communication rounds.

In general, in problems with a relatively small number of robots and tasks, the MUR-TAP algorithm provides the fastest convergence rate compared to the other algorithms, given that each robot in the MUR-TAP algorithm computes a feasible assignment for all robots at each iteration of the algorithm. Each robot communicates with its neighbors across multiple iterations to reach consensus on a joint optimal assignment for all robots. In contrast, MURD-TAP and MURID-TAP do not provide a feasible assignment for all robots at each iteration until convergence. However, in larger problems, the effects of the computation and communication complexity of

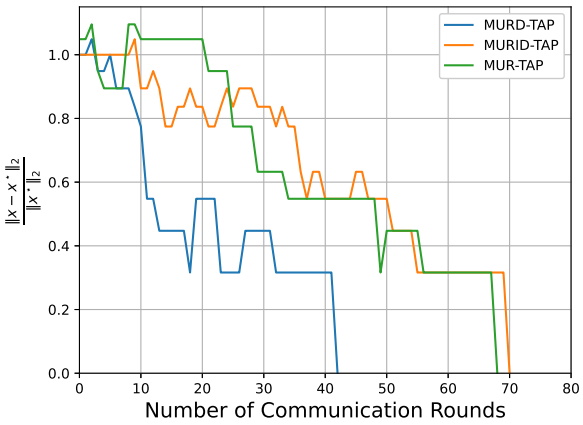


Fig. 2. Relative error of MUR-TAP, MURD-TAP, and MURID-TAP on the multi-robot task assignment problem with 10 robots and 10 tasks. Compared to the task assignment problem with 5 robots, each robot in all the algorithms requires a greater number of communication rounds to compute its optimal task assignment. In this trial, MURD-TAP provides the fastest convergence rate compared to the other algorithms.

each algorithm become more apparent, resulting in slower convergence of the MUR-TAP algorithm.

In the subsequent simulations, we consider communication networks with different topology in evaluating the convergence rate of our algorithm and, in addition, compare our algorithm to other distributed task assignment algorithms.

*2) Convergence across Networks with Different Topology:* In this simulation study, we assume each robot has limited onboard storage, and thus, each robot requires a distributed algorithm with a low data storage overhead. Considering these limitations, we select the MURD-TAP algorithm to solve the task assignment problems in these scenarios. We examine the convergence rate of MURD-TAP to the optimal solution of the multi-robot task assignment problem with 20 robots and 20 tasks across randomly-generated connected communication networks with different connectivity ratios. We denote the connectivity ratio of a graph as $\kappa = \frac{2\ |E|}{N(N-1)}$, noting that a fully-connected network has a connectivity ratio of 1. In Table IV, we show the mean and standard deviation of the number of communication rounds and cumulative computation time in milliseconds required by each robot to compute a task assignment with a relative error of at most $1\mathrm{e}^{-11}\%$ across 60 problems.

From Table IV, we note that the number of communication rounds required for convergence decreases significantly as the connectivity ratio of a communication network increases from about 0.2 to 0.6, with a much smaller difference as the connectivity ratio increases beyond 0.6. Likewise, the cumulative computation time per robot decreases by about 7 msec over this range of the connectivity ratio. On fully-connected communication networks, each robot requires about 54 communication rounds and about 11.541 msec in total computation time, to compute its optimal task assignment, compared to about 94 communication rounds and about 19.745 msec on communication networks with a connectivity ratio of 0.253. Consequently, the connectivity ratio of the underlying communication network influences the convergence rate of our algorithms, with a greater connectivity ratio corresponding to faster convergence.

TABLE IV
THE MEAN AND STANDARD DEVIATION OF THE NUMBER OF COMMUNICATION ROUNDS (# OF COMM.) AND CUMULATIVE COMPUTATION TIME (COMP. TIME) IN MILLISECONDS OF MURD-TAP, PER ROBOT, IN 60 TASK ASSIGNMENT PROBLEMS WITH 20 ROBOTS AND 20 TASKS ON RANDOMLY-GENERATED CONNECTED NETWORKS WITH DIFFERENT CONNECTIVITY RATIOS.

| Connectivity Ratio ($\kappa$) | # of Comm. | Comp. Time (msec) |
|---|---|---|
| 0.253 | $94 \pm 32$ | $19.745 \pm 6.819$ |
| 0.595 | $61 \pm 25$ | $13.096 \pm 5.166$ |
| 0.879 | $56 \pm 20$ | $11.679 \pm 4.044$ |
| 1.000 | $54 \pm 18$ | $11.541 \pm 3.790$ |

In Figure 3, we show the relative error of our algorithm during one trial across the communication networks presented in Table IV. Our algorithm attains its slowest convergence rate when $\kappa = 0.253$, the minimum connectivity ratio in Table IV, with a notable improvement in the convergence rate for larger values of $\kappa$. As a result, on communication networks with low connectivity ratios, each robot requires a greater number of communication rounds to achieve consensus with its neighbors on an optimal task assignment.

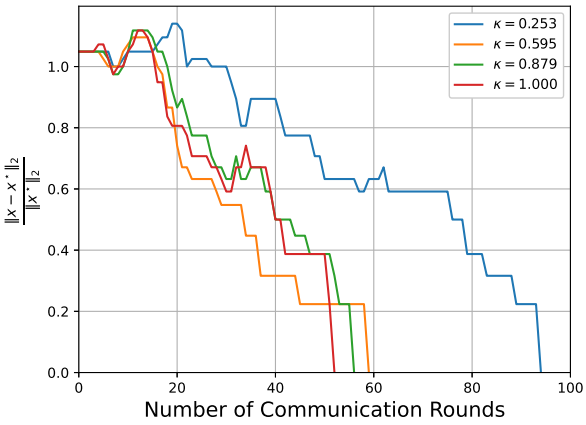


Fig. 3. Relative error of MURD-TAP on the multi-robot task assignment problem with 20 robots and 20 tasks across randomly-generated connected networks with different connectivity ratios. Our algorithm attains faster convergence rates in communication networks with larger connectivity ratios.

*3) Comparison to Benchmark Methods:* We assume each robot has limited access to computational and data storage resources. Consequently, we examine the convergence rate of MURD-TAP and MURID-TAP for the multi-robot task assignment problem in comparison to that of other distributed task assignment algorithms, including the consensus-based auction method (CBAA) [29], market-based consensus method (MBC) [17], and the distributed Hungarian method (DH) in [1]. While the MBC, DH, MURD-TAP, and MURID-TAP algorithms provide an optimal task assignment, CBAA does not guarantee an optimal task assignment. In addition,

the MBC method requires all robots with conflicting task preferences to communicate with a merchant, which can be played by any of the robots, to resolve conflicting choices between the robots — which might be infeasible in randomly-generated connected networks and chain networks, making the algorithm unsuitable in these problems. As a result, we consider a multi-robot task assignment problem with 50 robots, communicating over a fully-connected network, and 50 tasks. Table V presents the mean and standard deviation of the number of communication rounds and cumulative computation time in milliseconds required by each robot to compute a solution with a percentage relative error of at most $1\mathrm{e}^{-11}\%$ with respect to an optimal task assignment across 35 problems. In addition, we indicate any guarantees provided by each algorithm on producing an optimal task assignment (Opt) and the amenablity of each algorithm to task assignment problems with nonlinear objective functions (NL).

TABLE V
THE MEAN AND STANDARD DEVIATION OF THE NUMBER OF COMMUNICATION ROUNDS (# OF COMM.) AND CUMULATIVE COMPUTATION TIME (COMP. TIME) IN MILLISECONDS, PER ROBOT, OF DIFFERENT ALGORITHMS, AS WELL AS GUARANTEES ON PRODUCING AN OPTIMAL SOLUTION (OPT) AND AMENABILITY TO PROBLEMS WITH NONLINEAR OBJECTIVE FUNCTIONS (NL) IN 35 TASK ASSIGNMENT PROBLEMS WITH 50 ROBOTS AND 50 TASKS. IN THE *Opt-NL* COLUMN, "N" INDICATES FALSE, WHILE "Y" INDICATES TRUE.

| Algorithm | # of Comm. | Comp. Time (msec) | Opt-NL |
|---|---|---|---|
| CBAA [29] | $5 \pm 1$ | $0.5 \pm 0.271$ | N-N |
| MBC [17] | $1049 \pm 256$ | $21.6 \pm 7.7$ | Y-N |
| DH [1] | $68 \pm 9$ | $210 \pm 43.8$ | Y-N |
| MURD-TAP (ours) | $172 \pm 46$ | $63.329 \pm 16.591$ | Y-Y |
| MURID-TAP (ours) | $385 \pm 94$ | $8.598 \pm 1.846$ | Y-Y |

CBAA requires the minimum cumulative computation time and the fewest number of communication rounds to compute a feasible task assignment for all robots. However, we note that CBAA did not produce an optimal task assignment in all the problems that we considered. Among the optimal task assignment algorithms — MBC, DH, MURD-TAP, and MURID-TAP — the MURID-TAP algorithm requires the shortest mean cumulative computation time to produce an optimal task assignment. However, each robot in the MURID-TAP algorithm requires a greater number of communication rounds to compute an optimal task assignment compared to the DH and MURD-TAP algorithms. Nonetheless, MURID-TAP requires fewer communication rounds to compute the optimal task assignment compared to the MBC algorithm, which requires the greatest number of communication rounds. In addition, the MBC algorithm is unsuitable for problems with nonlinear objective functions. Although the DH algorithm requires the fewest communication rounds among the optimal algorithms, the DH algorithm requires a greater cumulative computation time per robot compared to MURD-TAP and MURID-TAP. Moreover, the DH algorithm is not amenable to task assignment problems with nonlinear objective functions and general affine constraints. In contrast, MURD-TAP and MURID-TAP offer a balanced trade-off with respect to computation time, the number of communication rounds required for convergence, and versatility, producing an optimal solution in general task assignment problems with nonlinear objective functions and affine constraints. MURD-TAP produces an optimal task assignment within fewer communication rounds while requiring a shorter cumulative computation time. Conversely, MURID-TAP achieves the fastest cumulative computation time with a greater number of communication rounds.

### *B. Nonlinear Multi-Robot Task Assignment Problem*

We consider the multi-robot task assignment problem in (2) with nonlinear objective functions and an unequal number of robots and tasks, where the objective function $f_i(x_i)$ of robot $i$ is given by

$$f_i(x_i) = \sum_{\tau=1}^{m} \frac{1}{1 + \alpha_{i,\tau} \mathrm{e}^{\beta_{i,\tau} x_{i,\tau}}} \tag{59}$$

with $x_i \in \mathbb{R}^m$ denoting the optimization variable of robot $i$. We examine the convergence rate of MURID-TAP on the nonlinear task assignment problem on randomly-generated connected communication networks, noting the simple update procedures involved in the algorithm. We randomly generate the vectors $\alpha_i \in \mathbb{R}^m$ and $\beta_i \in \mathbb{R}^m$ arising in the objective function of robot $i$, with $\alpha_i \geq 0$ and $\beta_i \geq 0$. In general, the optimal solution of (2) does not always correspond to an integer-valued solution. However, when an integer-valued optimal solution exists, this solution corresponds to the optimal task assignment. We generate the objective function, given by (59), such that an integer-valued optimal solution exists.

For the nonlinear task assignment problem with $N = 20$ robots and $m = 15$ tasks, we examine the number of communication rounds and the cumulative computation time in milliseconds required by each robot to compute a solution with a relative error of at most $1\mathrm{e}^{-11}\%$, with the mean and standard deviation across 60 problems provided in Table VI.

TABLE VI
THE MEAN AND STANDARD DEVIATION OF THE NUMBER OF COMMUNICATION ROUNDS (# OF COMM.) AND CUMULATIVE COMPUTATION TIME (COMP. TIME) IN MILLISECONDS, PER ROBOT, IN 60 NONLINEAR TASK ASSIGNMENT PROBLEMS WITH 20 ROBOTS AND 15 TASKS.

| Algorithm | # of Comm. | Comp. Time (msec) |
|---|---|---|
| MURID-TAP | $276 \pm 12$ | $3.799 \pm 0.205$ |

With MURID-TAP, each robot computes its optimal task assignment efficiently, within a mean computation time of 3.799 msec. The cumulative computation required by each robot depends on the computational difficulty in evaluating the gradient of the objective function, required in the update procedure in (53). Figure 4 shows the relative error of the solution computed by the robots during one trial. Each robot computes its optimal task assignment within 300 communication rounds.

### *C. Case Study: Persistent Surveillance*

In the previous discussion, we considered single or one-shot task assignment problems; however, many multi-robot systems

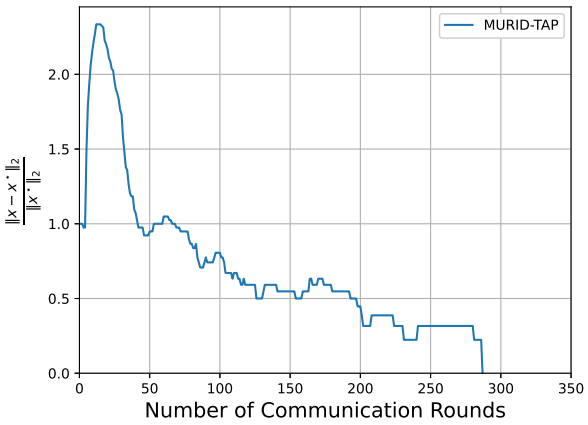


Fig. 4. Relative error of MURID-TAP on the nonlinear multi-robot task assignment problem with 20 robots and 15 tasks on randomly-generated connected networks.

require periodic assignments of robots to tasks over time, such as persistent surveillance problems with battery constraints. In this case study, we consider multi-robot persistent surveillance where a group of robots must maintain coverage over a specified region. Given the battery constraints of each robot, the robots must alternate between performing the surveillance task and charging their batteries. As a result, assignment of robots to the tasks must occur periodically. Moreover, robots at surveillance stations can only move to a charging station if another robot chooses to swap stations with it. We consider a problem with 12 robots and four dynamic surveillance stations, where the location of the surveillance stations are prescribed by a given trajectory. Figure 5 shows four robots with spotlights surveilling a city, with eight charging stations (the green squares) located around the perimeter of the city. At this time instant, one robot (the robot within the red circle) is in the process of swapping stations, moving from a surveillance station to a charging station, as depicted in the Figure 5. We use the terms *robot* and *drone* interchangeably. In the subsequent discussion, we provide the state dynamics models of the robots, their battery levels, and the surveillance stations. In addition, we specify the objective cost associated with each task assignment and compute the optimal task assignment for the surveillance problem.

*1) State Dynamics:* The state consists of a position $p_i \in \mathbb{R}^3$ and battery level $b_i \in \mathbb{Z}$ for each robot. At each time step, each robot can be at a charging station, at a surveillance station, or flying between stations. When at a charging station, the robot charges at a rate of $r_c$ per time step up to a maximum battery level of $b_{\max}$. Otherwise, the robot discharges at a rate of $r_d$ per time step down to a minimum battery level of 0, at which point the robot shutdowns as its battery capacity is fully drained. We provide the dynamics model for the battery capacity at time $t$ in (60).

$$b_i^{t+1} = \begin{cases} 0 & \text{if } b_i^t = 0 \\ \min(b_i^t + r_c, b_{\max}) & \text{if } i \text{ charging} \\ \max(b_i^t - r_d, 0) & \text{otherwise} \end{cases} \tag{60}$$

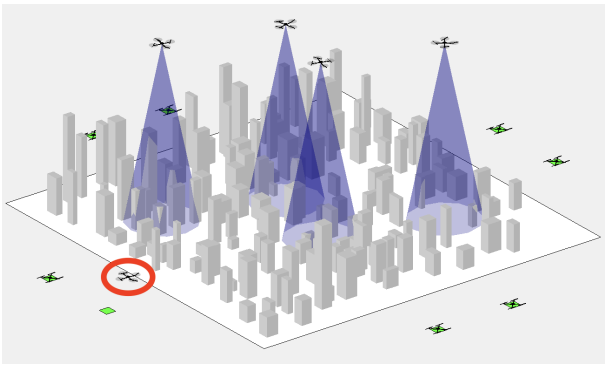

Fig. 5. One time step of a simulation. The charging stations are green squares around the perimeter of the city. The four drones over the city with spotlights are performing the surveillance task. One drone near the lower left (the robot within the red circle) is in the process of a replacement.

Stations follow known paths and are represented by functions $s_j(t) : \mathbb{Z} \to \mathbb{R}^3$. Charging stations are at fixed points in space, meaning that $s_j(t_1) = s_j(t_2)\ \forall t_1, t_2 \in \mathbb{Z}$. Surveillance stations may be at fixed points or follow paths.

Actions $a_i^t \in \mathbb{Z}$ dictate where each robot should go at time step $t$. If $a_i^t = j$, then robot $i$ should stay at station $j$ if it is there already or fly to station $j$ if it is at a different station. If robot $i$ needs to change its station, it chooses its goal location $g_i^t$ according to

$$g_i^t = s_{a_i^t}\left(\min_{\Delta t \geq 1}\left(\Delta t \mid \|s_{a_i^t}(t+\Delta t) - p_i^t\|_2 \leq v\Delta t\right)\right) \tag{61}$$

where $v \in \mathbb{R}$ denotes its speed. The goal point is the first point along the assigned station's path that the robot can reach no later than it needs to in order to intercept the station when moving at a speed $v$. Once it has a goal location, the robot moves toward the goal point at a rate of $v$ per time step, with the dynamics of the robot given by

$$p_i^{t+1} = p_i^t + \frac{g_i^t - p_i^t}{\|g_i^t - p_i^t\|_2} \min\left(\|g_i^t - p_i^t\|_2, v\right). \tag{62}$$

The surveillance paths are set so that they do not require movement faster than $v$.

*2) Costs and Assignments:* Assignments are influenced by constraints on the process of swapping between stations, as well as the costs associated with swapping stations. We delineate the problem constraints: First, each surveillance station must always have a drone at it. The drone at a surveillance station may only leave once a replacement has arrived, at which point it travels to the charging station that the replacement vacated. Second, once a replacement is sent, a second replacement may not be sent until the original replacement arrives. Third, a charging drone should not replace a surveillance drone if the latter has a higher battery level. Fourth, drones may not change assignments when traveling between stations. Fifth, drones may not swap between charging stations.

Costs determine the replacements. If drone $i$ is charging at station $j$, the cost of staying at station $j$ is

$$c_{ij} = -(b_i^t - b_{\max})^2, \tag{63}$$

which gives the drone less incentive to keep charging as its battery level rises. If, however, station $j$ is a surveillance station, the cost of assigning drone $i$ to be the replacement is

$$c_{ij} = d_{ij}^2 - (b_i^t - b_j^t), \tag{64}$$

where $d_{ij}$ is the required travel distance, computed with Eq. (61), and $b_j^t$ denotes the battery level of the drone currently at station $j$. This cost encourages replacements where the drone at the charging station has a much larger battery capacity and discourages replacements that require a large travel distance.

We compute the task assignments using MURID-TAP at each assignment episode, comparing the resulting assignments to the optimal solution obtained using the Hungarian method. In all episodes, MURID-TAP produces the optimal task assignment, depicted in Figure 6 for 2000 assignment episodes. The assignment cost obtained using MURID-TAP overlays the assignment cost achieved using the Hungarian method at all episodes. A limit cycle of the total objective cost emerges as the task proceeds, with the robots alternating between surveillance and charging stations. Further, we show the battery capacities of the 12 robots in Figure 7, with a limit cycle also emerging. The battery capacity of each robot ranges from 0 (fully discharged) to 100 (fully charged). From Figure 7, we note that the battery capacities of all robots remain above 20 at all episodes during the task, highlighting that no robot ends up with a fully discharged battery. In addition, the robots maintain a mean battery capacity of about 70 (see the supplementary video for more details).

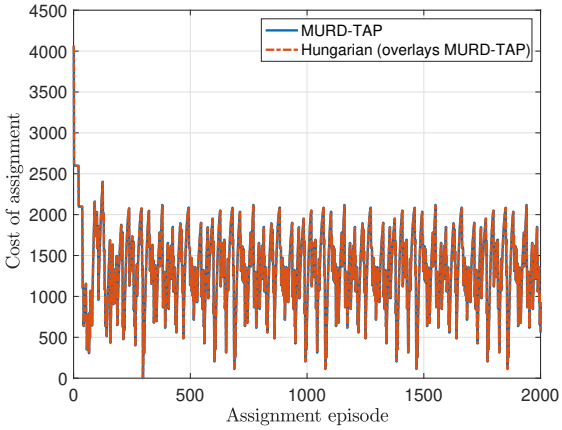


Fig. 6. Assignment of 12 robots to two surveillance stations on a persistent surveillance task using MURD-TAP. MURD-TAP produces the optimal task assignment at each episode, with a limit cycle emerging as the task proceeds.

## VIII. Conclusions

We derive distributed algorithms for solving multi-robot task assignment problems that only require a connected communication graph among the robots. We present three related algorithms, all derived from variants of the Alternating Direction Method of Multipliers (ADMM), each with its own advantages and disadvantages. All three algorithms are proven to produce the optimal task assignment. The two algorithms based on the dual problem formulation also provide improved privacy to each robot, as each robot does not share its own task assignment while communicating with its neighbors. We provide extensive empirical results showing the trade-offs among the three algorithms in terms of computational speed, storage requirements, and communication complexity. We also show faster convergence rates attained by our algorithms in comparison to other distributed methods for multi-robot task assignment. Lastly, we demonstrated the potential of our algorithms in a more complex sequential task assignment problem in which a group of drones must coordinate with one another to perform aerial surveillance over a city while periodically returning to charging stations to charge their batteries. Our task assignment algorithms allow the robots to achieve a limit cycle persistent monitoring behavior, balancing recharging and surveillance activities amongst themselves in a fully distributed fashion in perpetuity.

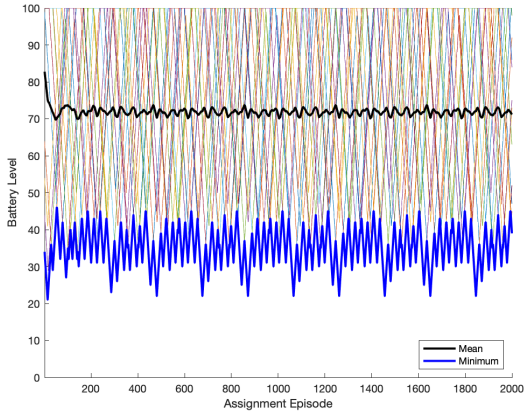


Fig. 7. Battery levels of the 12 robots during the simulation. The mean and minimum battery levels are highlighted to demonstrate the limit cycle and show that no batteries are completely discharged.

## Appendix

*Proof of Theorem 1*

We can express the primal mathematical program in (10) as

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{\sigma}}{\text{minimize}} \ & \sum_{i=1}^{N} f_i(\boldsymbol{x}_i) \\ \text{subject to} \ & A_i \boldsymbol{x}_i \leq b_i \quad \forall i \in \mathcal{V} \\ & \mathcal{F}\boldsymbol{x} = \mathcal{G}\boldsymbol{\sigma} \end{aligned} \tag{65}$$

where $\mathcal{F} \in \mathbb{R}^{2|\mathcal{E}|Nm \times mN^2}$ and $\mathcal{G} \in \mathbb{R}^{2|\mathcal{E}|Nm \times 2|\mathcal{E}|Nm}$ define the mapping between the local $x$ variable of each robot and the slack variables in $\boldsymbol{\sigma}$, as expressed by the equality constraints in (10). The augmented Lagrangian for the problem in (65) is given by

$$\begin{aligned} \mathcal{L}_a^{\mathcal{F}}(\boldsymbol{x}, \boldsymbol{\sigma}, \mu) = & \sum_{i=1}^{N} f_i(\boldsymbol{x}_i) + \mu^{\mathsf{T}} \Big( \mathcal{F}\boldsymbol{x} - \mathcal{G}\boldsymbol{\sigma} \Big) \\ & + \frac{\rho}{2} \left\| \mathcal{F}\boldsymbol{x} - \mathcal{G}\boldsymbol{\sigma} \right\|_2^2 \end{aligned} \tag{66}$$

where $\mu \in \mathbb{R}^{2|\mathcal{E}|Nm}$ represents a Lagrange multiplier for the equality constraint in (65). We have not relaxed the inequality constraint in (65).

By simplifying the minimization problem for the primal update procedure in (15), robot $i$ computes its primal variables from the problem

$$\begin{aligned} \underset{\boldsymbol{x}_i}{\text{minimize}} \; & \left\{ f_i(\boldsymbol{x}_i) + \rho|\mathcal{N}_i| \left\|\boldsymbol{x}_i\right\|_2^2 + \boldsymbol{x}_i^{\mathsf{T}} q_i^k \right. \\ & \left. - \rho \boldsymbol{x}_i^{\mathsf{T}} \sum_{j \in \mathcal{N}_i} \left(\boldsymbol{x}_i^k + \boldsymbol{x}_j^k\right) \right\} \\ \text{subject to } \; & A_i \boldsymbol{x}_i \leq b_i \end{aligned} \tag{67}$$

at each iteration. The local objective function of each robot as defined in (7) is convex, proper, and closed. Consequently, the minimization problem in (67) has a proper, closed, and coercive objective function. From Weirstrass theorem [53], the minimization problem has a minimum value which is attained at a solution $\tilde{x}$, within the feasible set of the problem. As such, a minimizer exists for the minimization problems arising at each iteration of our algorithm.

At iteration $k$, the iterate $\boldsymbol{x}^{k+1}$ minimizes $\mathcal{L}_a^{\mathcal{F}}(\boldsymbol{x}, \boldsymbol{\sigma}^k, \mu^k)$. Likewise, $\boldsymbol{\sigma}^{k+1}$ minimizes $\mathcal{L}_a^{\mathcal{F}}(\boldsymbol{x}^{k+1}, \boldsymbol{\sigma}, \mu^k)$. From the optimality conditions arising in the primal update procedure, the iterates generated by our algorithm satisfy

$$\begin{aligned} &\frac{1}{\rho}\left(\left\|\mu^k - \mu^\star\right\|_2^2 - \left\|\mu^{k+1} - \mu^\star\right\|_2^2\right) \\ &\quad + \rho\left(\left\|\mathcal{G}\left(\boldsymbol{\sigma}^k - \boldsymbol{\sigma}^\star\right)\right\|_2^2 - \left\|\mathcal{G}\left(\boldsymbol{\sigma}^{k+1} - \boldsymbol{\sigma}^\star\right)\right\|_2^2\right) \\ &\qquad \geq \rho\left\|\mathcal{F}\boldsymbol{x}^{k+1} - \mathcal{G}\boldsymbol{\sigma}^{k+1}\right\|_2^2 + \rho\left\|\mathcal{G}\left(\boldsymbol{\sigma}^{k+1} - \boldsymbol{\sigma}^k\right)\right\|_2^2 \end{aligned} \tag{68}$$

at each iteration, where $(\boldsymbol{x}^\star, \boldsymbol{\sigma}^\star, \mu^\star)$ represents a saddle-point of the Lagrangian of (65). From (68), the weighted error between the iterates $(\boldsymbol{\sigma}^k, \mu^k)$ and $(\boldsymbol{\sigma}^\star, \mu^\star)$

$$\frac{1}{\rho}\left\|\mu^k - \mu^\star\right\|_2^2 + \rho\left\|\mathcal{G}\left(\boldsymbol{\sigma}^k - \boldsymbol{\sigma}^\star\right)\right\|_2^2 \tag{69}$$

decreases sufficiently at each iteration, which shows that the iterates $(\mathcal{G}\boldsymbol{\sigma}^k, \mu^k)$ are bounded. Summing (68) over all iterations results in

$$\begin{aligned} &\rho\sum_{k=0}^{\infty}\left(\left\|\mathcal{F}\boldsymbol{x}^{k+1} - \mathcal{G}\boldsymbol{\sigma}^{k+1}\right\|_2^2 + \left\|\mathcal{G}\left(\boldsymbol{\sigma}^{k+1} - \boldsymbol{\sigma}^k\right)\right\|_2^2\right) \\ &\qquad \leq \frac{1}{\rho}\left\|\mu^0 - \mu^\star\right\|_2^2 + \rho\left\|\mathcal{G}\left(\boldsymbol{\sigma}^0 - \boldsymbol{\sigma}^\star\right)\right\|_2^2 \end{aligned} \tag{70}$$

which shows that

$$\begin{aligned} &\mathcal{F}\boldsymbol{x}^{k+1} - \mathcal{G}\boldsymbol{\sigma}^{k+1} \to 0 \quad \text{and} \\ &\mathcal{G}\left(\boldsymbol{\sigma}^{k+1} - \boldsymbol{\sigma}^k\right) \to 0 \end{aligned} \tag{71}$$

as $k \to \infty$. Consequently, the local primal variable of all robots converge to the same solution, satisfying the equality constraints in (10). In addition, the iterates satisfy

$$\begin{aligned} &\sum_{i=1}^{N} f_i\left(\boldsymbol{x}_i^{k+1}\right) - f_i\left(\boldsymbol{x}_i^\star\right) \\ &\quad \leq -\left(\mathcal{F}\boldsymbol{x}^{k+1} - \mathcal{G}\boldsymbol{\sigma}^{k+1}\right)^{\mathsf{T}}\mu^{k+1} \\ &\qquad + \rho\left(\mathcal{F}\boldsymbol{x}^{k+1} - \mathcal{G}\boldsymbol{\sigma}^{k+1}\right)^{\mathsf{T}}\mathcal{G}\left(\boldsymbol{\sigma}^{k+1} - \boldsymbol{\sigma}^k\right) \\ &\qquad - \rho\left(\mathcal{G}\left(\boldsymbol{\sigma}^{k+1} - \boldsymbol{\sigma}^\star\right)\right)^{\mathsf{T}}\mathcal{G}\left(\boldsymbol{\sigma}^{k+1} - \boldsymbol{\sigma}^k\right) \end{aligned} \tag{72}$$

along with

$$\sum_{i=1}^{N} f_i\left(\boldsymbol{x}_i^{k+1}\right) - f_i\left(\boldsymbol{x}_i^\star\right) \geq -\left(\mathcal{F}\boldsymbol{x}^{k+1} - \mathcal{G}\boldsymbol{\sigma}^{k+1}\right)^{\mathsf{T}}\mu^{k+1} \tag{73}$$

at each iteration, where $\boldsymbol{x}_i^\star = x^\star$. As $k \to \infty$,

$$\sum_{i=1}^{N} f_i\left(x_i^{k+1}\right) - f_i\left(\boldsymbol{x}_i^\star\right) \to 0 \tag{74}$$

from the boundedness of $\mathcal{G}\left(\boldsymbol{\sigma}^{k+1} - \boldsymbol{\sigma}^\star\right)$ and (71), showing that the objective value converges to its optimal value. Further, the iterates $(\boldsymbol{x}^k, \boldsymbol{\sigma}^k, \mu^k)$ converge to $(\boldsymbol{x}^\star, \boldsymbol{\sigma}^\star, \mu^\star)$, the saddle-point of the Lagrangian. Refer to [54], [55] for a detailed proof.

*Proof of Theorem 2*

The proof follows along the same lines as the proof of Theorem 1. For completeness, we provide the proof here. We can express the mathematical program in (25) as

$$\begin{aligned} \underset{\boldsymbol{y}, \boldsymbol{\vartheta}}{\text{maximize}} \; & \sum_{i=1}^{N} \left(-f_i^\star(-A_i^{\mathsf{T}} y_i) - y_i^{\mathsf{T}} b_i\right) \\ \text{subject to } \; & y_i \geq 0 \quad \forall i \in \mathcal{V} \\ & \tilde{\mathcal{F}}\boldsymbol{y} = \tilde{\mathcal{G}}\boldsymbol{\vartheta} \end{aligned} \tag{75}$$

where $\tilde{\mathcal{F}} \in \mathbb{R}^{2|\mathcal{E}|m \times Nm}$ and $\tilde{\mathcal{G}} \in \mathbb{R}^{2|\mathcal{E}|m \times 2|\mathcal{E}|m}$ define the mapping between the local $y$ variable of each robot and the slack variables $[\gamma_{ij}^{\mathsf{T}}, \zeta_{ij}^{\mathsf{T}},\ \forall (i,j) \in \mathcal{E}]^{\mathsf{T}}$ in $\boldsymbol{\vartheta}$. Note that the maximization problem in (30) has a proper, closed, and coercive objective function. Further, a maximizer exists for the optimization problem in (30). As a result, our algorithm for the class of mathematical programs $\mathcal{D}$ generates bounded iterates $(\boldsymbol{y}^k, \boldsymbol{\vartheta}^k, \varpi^k)$ which satisfy

$$\begin{aligned} &\frac{1}{\rho}\left(\left\|\varpi^k - \varpi^\star\right\|_2^2 - \left\|\varpi^{k+1} - \varpi^\star\right\|_2^2\right) \\ &\quad + \rho\left(\left\|\tilde{\mathcal{G}}\left(\boldsymbol{\vartheta}^k - \boldsymbol{\vartheta}^\star\right)\right\|_2^2 - \left\|\tilde{\mathcal{G}}\left(\boldsymbol{\vartheta}^{k+1} - \boldsymbol{\vartheta}^\star\right)\right\|_2^2\right) \\ &\qquad \geq \rho\left\|\tilde{\mathcal{F}}\boldsymbol{y}^{k+1} - \tilde{\mathcal{G}}\boldsymbol{\vartheta}^{k+1}\right\|_2^2 + \rho\left\|\tilde{\mathcal{G}}\left(\boldsymbol{\vartheta}^{k+1} - \boldsymbol{\vartheta}^k\right)\right\|_2^2 \end{aligned} \tag{76}$$

at each iteration, showing that the weighted error between $(\boldsymbol{\vartheta}^k, \varpi^k)$ and $(\boldsymbol{\vartheta}^\star, \varpi^\star)$

$$\frac{1}{\rho}\left\|\varpi^k - \varpi^\star\right\|_2^2 + \rho\left\|\tilde{\mathcal{G}}\left(\boldsymbol{\vartheta}^k - \boldsymbol{\vartheta}^\star\right)\right\|_2^2 \tag{77}$$

decreases at each iteration, where $(\boldsymbol{y}^\star, \boldsymbol{\vartheta}^\star, \varpi^\star)$ represents a saddle-point of the Lagrangian of (75).

In addition, as $k \to \infty$,

$$\begin{aligned} &\tilde{\mathcal{F}}\boldsymbol{y}^{k+1} - \tilde{\mathcal{G}}\boldsymbol{\vartheta}^{k+1} \to 0 \quad \text{and} \\ &\tilde{\mathcal{G}}\left(\boldsymbol{\vartheta}^{k+1} - \boldsymbol{\vartheta}^{k}\right) \to 0, \end{aligned} \tag{78}$$

showing that the dual variables of all robots converge to the same solution. Ultimately, the objective value of (75) computed at $\boldsymbol{y}^k$ converges to its optimal value, with the iterates $(\boldsymbol{y}^k, \boldsymbol{\vartheta}^k, \varpi^k)$ converging to a saddle-point $(\boldsymbol{y}^\star, \boldsymbol{\vartheta}^\star, \varpi^\star)$ of the Lagrangian of (75).

*Proof of Theorem 3*

We can express the Lagrangian in (20) as

$$\mathcal{L}_a(\tilde{\boldsymbol{y}}, x) = \sum_{i=1}^{N} f_i(x_i) + \tilde{y}_i^{\mathsf{T}}(A_i x_i - b_i) \tag{79}$$

where robot $i$ maintains the local dual variable $\tilde{y}_i$, with the constraints

$$\tilde{y}_i = \tilde{y}_j \quad \forall (i,j) \in \mathcal{E}\ , \tag{80}$$

ensuring all the dual variables have the same value. By relaxing the constraints in (80), we obtain the augmented Lagrangian

$$\begin{aligned} \mathcal{L}_a(\tilde{\boldsymbol{y}}, x, \gamma, \zeta, p, w) = &\sum_{i=1}^{N} f_i(x_i) + \tilde{y}_i^{\mathsf{T}}(A_i x_i - b_i) \\ &- \sum_{(i,j)\in\mathcal{E}} \Big( p_{ij}^{\mathsf{T}}(\tilde{y}_i - \gamma_{ij}) + w_{ij}^{\mathsf{T}}(\tilde{y}_j - \zeta_{ij}) \Big) \\ &- \frac{\rho}{2} \sum_{(i,j)\in\mathcal{E}} \Big( \|\tilde{y}_i - \gamma_{ij}\|_2^2 + \|\tilde{y}_j - \zeta_{ij}\|_2^2 \Big) \end{aligned} \tag{81}$$

with Lagrange multipliers $p$ and $w$ and slack variables $\gamma$ and $\zeta$ for the equality constraints in (80) over the feasible set where $\gamma_{ij} = \zeta_{ij}$. From the existence of a saddle-point, we can compute the optimal primal solution $x^\star$ from the problem

$$\begin{aligned} \underset{x}{\text{minimize}}\ \underset{\tilde{\boldsymbol{y}}}{\text{maximize}}\ &\mathcal{L}_a^d(\tilde{\boldsymbol{y}}, x, \gamma^\star, \zeta^\star, p^\star, w^\star) \\ \text{subject to}\ &\tilde{y}_i \geq 0 \quad \forall i \in \mathcal{V} \end{aligned} \tag{82}$$

where $\gamma^\star$, $\zeta^\star$, $p^\star$, and $w^\star$ represent an optimal solution for the slack variables and Lagrange multipliers. The optimization problem in (82) simplifies to

$$\begin{aligned} \underset{x}{\text{minimize}}\ \underset{\tilde{\boldsymbol{y}}}{\text{maximize}}\ &\sum_{i=1}^{N} \mathcal{P}_i(\tilde{y}_i, x_i) \\ \text{subject to}\ &\tilde{y}_i \geq 0 \quad \forall i \in \mathcal{V} \end{aligned} \tag{83}$$

where

$$\begin{aligned} \mathcal{P}_i(\tilde{y}_i, x_i) = &f_i(x_i) + \tilde{y}_i^{\mathsf{T}}(A_i x_i - b_i) \\ &- r_i^{\star\mathsf{T}}\tilde{y}_i - \rho \sum_{j\in\mathcal{N}_i} \left\| \tilde{y}_i - \frac{\bar{y}_i + \bar{y}_j}{2} \right\|_2^2, \end{aligned} \tag{84}$$

upon convergence of the MURD-MP algorithm. The optimization problem in (83) decomposes into $N$ independent optimization problems, with the problem for robot $i$ given by

$$\begin{aligned} \underset{x_i}{\text{minimize}}\ \underset{\tilde{y}_i}{\text{maximize}}\ &\mathcal{P}_i(\tilde{y}_i, x_i) \\ \text{subject to}\ &\tilde{y}_i \geq 0 \end{aligned} \tag{85}$$

which simplifies to the same optimization problem in (34). Hence, the local primal variables computed by all robots using (34) corresponds to the optimal primal solution $x^\star$ of (18).

## REFERENCES


[1] S. Chopra, G. Notarstefano, M. Rice, and M. Egerstedt, "A distributed version of the Hungarian method for multirobot assignment," *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 932–947, 2017.

[2] J. Turner, Q. Meng, G. Schaefer, A. Whitbrook, and A. Soltoggio, "Distributed task rescheduling with time constraints for the optimization of total task allocations in a multirobot system," *IEEE transactions on cybernetics*, vol. 48, no. 9, pp. 2583–2597, 2017.

[3] J. Scherer and B. Rinner, "Multi-robot persistent surveillance with connectivity constraints," *IEEE Access*, vol. 8, pp. 15 093–15 109, 2020.

[4] L. C. B. Da Silva, R. M. Bernardo, H. A. De Oliveira, and P. F. Rosa, "Multi-uav agent-based coordination for persistent surveillance with dynamic priorities," in *2017 International Conference on Military Technologies (ICMT)*. IEEE, 2017, pp. 765–771.

[5] X. Bai, M. Cao, W. Yan, and S. S. Ge, "Efficient routing for precedence-constrained package delivery for heterogeneous vehicles," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 1, pp. 248–260, 2019.

[6] N. Mathew, S. L. Smith, and S. L. Waslander, "Planning paths for package delivery in heterogeneous multirobot teams," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 4, pp. 1298–1308, 2015.

[7] B. Shirani, M. Najafi, and I. Izadi, "Cooperative load transportation using multiple uavs," *Aerospace Science and Technology*, vol. 84, pp. 158–169, 2019.

[8] D. Weyns, N. Boucké, and T. Holvoet, "Gradient field-based task assignment in an agv transportation system," in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 2006, pp. 842–849.

[9] K. Vivaldini, L. F. Rocha, N. J. Martarelli, M. Becker, and A. P. Moreira, "Integrated tasks assignment and routing for the estimation of the optimal number of agvs," *The International Journal of Advanced Manufacturing Technology*, vol. 82, no. 1-4, pp. 719–736, 2016.

[10] M. Bürger, G. Notarstefano, F. Bullo, and F. Allgöwer, "A distributed simplex algorithm for degenerate linear programs and multi-agent assignments," *Automatica*, vol. 48, no. 9, pp. 2298–2304, 2012.

[11] R. N. Haksar, O. Shorinwa, P. Washington, and M. Schwager, "Consensus-based admm for task assignment in multi-robot teams," in *International Symposium on Robotics Research*, 2019.

[12] R. Zhang and J. Kwok, "Asynchronous distributed admm for consensus optimization," in *International conference on machine learning*. PMLR, 2014, pp. 1701–1709.

[13] W. Deng, M.-J. Lai, Z. Peng, and W. Yin, "Parallel multi-block admm with o (1/k) convergence," *Journal of Scientific Computing*, vol. 71, no. 2, pp. 712–736, 2017.

[14] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5262–5276, 2010.

[15] E. Wei and A. Ozdaglar, "Distributed alternating direction method of multipliers," in *51st IEEE Conference on Decision and Control (CDC)*, 2012, pp. 5445–5450.

[16] S. Hosseini, A. Chapman, and M. Mesbahi, "Online distributed ADMM via dual averaging," in *53rd IEEE Conference on Decision and Control (CDC)*, 2014, pp. 904–909.

[17] L. Liu and D. A. Shell, "Optimal market-based multi-robot task allocation via strategic pricing." in *Robotics: Science and Systems*, vol. 9, no. 1, 2013, pp. 33–40.

[18] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[19] S. Giordani, M. Lujak, and F. Martinelli, "A distributed algorithm for the multi-robot task allocation problem," in *Trends in Applied Intelligent Systems*, N. García-Pedrajas, F. Herrera, C. Fyfe, J. M. Benítez, and M. Ali, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 721–730.

[20] M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Griffin, and A. Kleywegt, "Robot exploration with combinatorial auctions," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 2. IEEE, 2003, pp. 1957–1962.

[21] D. P. Bertsekas, "The auction algorithm: A distributed relaxation method for the assignment problem," *Annals of operations research*, vol. 14, no. 1, pp. 105–123, 1988.

[22] R. Zlot and A. Stentz, "Market-based multirobot coordination for complex tasks," *The International Journal of Robotics Research*, vol. 25, no. 1, pp. 73–101, 2006.

[23] E. Nunes and M. Gini, "Multi-robot auctions for allocation of tasks with temporal constraints," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, 2015.

[24] M. G. Lagoudakis, M. Berhault, S. Koenig, P. Keskinocak, and A. J. Kleywegt, "Simple auctions with performance guarantees for multi-robot task allocation," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 1. IEEE, 2004, pp. 698–705.

[25] S. Sariel and T. Balch, "Real time auction based allocation of tasks for multi-robot exploration problem in dynamic environments," in *Proceedings of the AAAI-05 Workshop on Integrating Planning into Scheduling*. sn, 2005, pp. 27–33.

[26] T. Lemaire, R. Alami, and S. Lacroix, "A distributed tasks allocation scheme in multi-uav context," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 4. IEEE, 2004, pp. 3622–3627.

[27] P. Sujit and R. Beard, "Distributed sequential auctions for multiple uav task allocation," in *2007 American Control Conference*. IEEE, 2007, pp. 3955–3960.

[28] S. L. Smith and F. Bullo, "Monotonic target assignment for robotic networks," *IEEE Transactions on Automatic Control*, vol. 54, no. 9, pp. 2042–2057, 2009.

[29] H. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 912–926, 2009.

[30] N. Michael, M. M. Zavlanos, V. Kumar, and G. J. Pappas, "Distributed multi-robot task assignment and formation control," in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 128–133.

[31] L. Luo, N. Chakraborty, and K. Sycara, "Distributed algorithm design for multi-robot task assignment with deadlines for tasks," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 3007–3013.

[32] T. Mercker, D. W. Casbeer, P. T. Millet, and M. R. Akella, "An extension of consensus-based auction algorithms for decentralized, time-constrained task assignment," in *Proceedings of the 2010 American Control Conference*. IEEE, 2010, pp. 6324–6329.

[33] M. M. Zavlanos, L. Spesivtsev, and G. J. Pappas, "A distributed auction algorithm for the assignment problem," in *2008 47th IEEE Conference on Decision and Control*, 2008, pp. 1212–1217.

[34] W. Yao, N. Qi, N. Wan, and Y. Liu, "An iterative strategy for task assignment and path planning of distributed multiple unmanned aerial vehicles," *Aerospace Science and Technology*, vol. 86, pp. 455–464, 2019.

[35] M. Otte, M. J. Kuhlman, and D. Sofge, "Auctions for multi-robot task allocation in communication limited environments," *Autonomous Robots*, vol. 44, no. 3, pp. 547–584, 2020.

[36] E. Schneider, E. I. Sklar, S. Parsons, and A. T. Özgelen, "Auction-based task allocation for multi-robot teams in dynamic environments," in *Conference Towards Autonomous Robotic Systems*. Springer, 2015, pp. 246–257.

[37] C. Schumacher, P. Chandler, M. Pachter, and L. Pachter, "Uav task assignment with timing constraints via mixed-integer linear programming," in *AIAA 3rd" Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, 2004, p. 6410.

[38] M. Darrah, W. Niland, and B. Stolarik, "Multiple uav dynamic task allocation using mixed integer linear programming in a sead mission," in *Infotech@ Aerospace*, 2005, p. 7164.

[39] R. J. Afonso, M. R. Maximo, and R. K. Galvão, "Task allocation and trajectory planning for multiple agents in the presence of obstacle and connectivity constraints with mixed-integer linear programming," *International Journal of Robust and Nonlinear Control*, vol. 30, no. 14, pp. 5464–5491, 2020.

[40] N. Atay and B. Bayazit, "Mixed-integer linear programming solution to multi-robot task allocation problem," 2006.

[41] A. Falsone, K. Margellos, and M. Prandini, "A decentralized approach to multi-agent MILPs: Finite-time feasibility and performance guarantees," *Automatica*, vol. 103, pp. 141–150, 2019.

[42] A. Camisa, I. Notarnicola, and G. Notarstefano, "A primal decomposition method with suboptimality bounds for distributed mixed-integer linear programming," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 3391–3396.

[43] S. Karaman and G. Inalhan, "Large-scale task/target assignment for uav fleets using a distributed branch and price optimization scheme," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 13 310–13 317, 2008.

[44] A. Testa, A. Rucco, and G. Notarstefano, "A finite-time cutting plane algorithm for distributed mixed integer linear programming," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 3847–3852.

[45] A. Settimi and L. Pallottino, "A subgradient based algorithm for distributed task assignment for heterogeneous mobile robots," in *52nd IEEE conference on decision and control*. IEEE, 2013, pp. 3665–3670.

[46] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," *Cooperative Robots and Sensor Networks 2015*, pp. 31–51, 2015.

[47] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[48] J. L. Miller and L. S. Franz, "A binary-rounding heuristic for multi-period variable-task-duration assignment problems," *Computers & operations research*, vol. 23, no. 8, pp. 819–828, 1996.

[49] J. W. Fowler, P. Wirojanagud, and E. S. Gel, "Heuristics for workforce planning with worker differences," *European Journal of Operational Research*, vol. 190, no. 3, pp. 724–740, 2008.

[50] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[51] T. Chang, M. Hong, and X. Wang, "Multi-agent distributed optimization via inexact consensus ADMM," *IEEE Transactions on Signal Processing*, vol. 63, no. 2, pp. 482–497, 2015.

[52] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2022. [Online]. Available: https://www.gurobi.com

[53] E. Bishop, "A generalization of the stone-weierstrass theorem." *Pacific Journal of Mathematics*, vol. 11, no. 3, pp. 777–783, 1961.

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

[55] L. Chen, D. Sun, and K.-C. Toh, "A note on the convergence of admm for linearly constrained convex optimization problems," *Computational Optimization and Applications*, vol. 66, no. 2, pp. 327–343, 2017.