Consensus-based ADMM for Task Assignment in Multi-Robot Teams

Ravi N. Haksar[1], Olaoluwa Shorinwa[1], Patrick Washington[2], and Mac Schwager[2]

[1] Department of Mechanical Engineering
[2] Department of Aeronautics & Astronautics
450 Serra Mall, Stanford, CA, USA 94305
`[rhaksar, shorinwa, phw, schwager]@stanford.edu`

Abstract. In this work, we leverage the alternating direction method of multipliers (ADMM) framework to solve task assignment for a multi-robot team. While ADMM is a well-established method, it has yet to be utilized in multi-robot problems as the standard formulation requires a centralized update step, a paradigm that conflicts with decentralization as a means of robustness. Therefore, we describe the formulation of separable optimizations in order to produce decentralized ADMM algorithms. Here, our aim is to provide an additional tool for solving cooperative team-based problems in robotics. For the decentralized algorithms, we discuss the conditions for convergence to the optimal centralized solution. We present simulation results for task assignment to demonstrate the benefits of ADMM compared to state-of-the-art methods.

Keywords: ADMM, task assignment, consensus methods, cooperative teams

1 Introduction

Cooperative team-based problems in robotics frequently involve solving a constrained optimization problem in a decentralized fashion, as in multi-robot formation control, task assignment, and target tracking. While the field of distributed optimization has recently made significant strides, much of this work has yet to be translated to the field of robotics. Typically, decentralized optimization methods for multi-robot systems are tailored for specific problem aspects, such as robot dynamics, sensor models, and communication architecture. In this work, we describe a more general framework for describing cooperative robotics problems, propose decentralized algorithms, and provide conditions for robots to determine the globally optimal solution. In particular, we discuss our approach in the context of multi-robot task assignment where robots are required to cooperatively complete a set of tasks. We consider two problem statements for the task assignment problem. First, there are equal number of tasks and robots and each task is assigned a unique robot. Second, there are less tasks than robots and each task must be assigned at least one robot. We consider a framework that naturally captures both of these problem statements and allows for a single algorithmic approach for simplicity and flexibility.

We leverage recent advances of the alternating direction method of multipliers (ADMM) framework for our decentralized algorithms. ADMM has been studied extensively and applied to problems mainly in machine learning and signal processing but few works have explored ADMM algorithms in robotics. Notably, the abstractions in distributed optimization do not directly translate to the abstractions typically considered in robotics. For example, agents in distributed optimization are information processors (e.g. CPU and GPU cores or remote servers) whereas agents in robotics are physical vehicles (e.g. mobile robots or drones) with limited processing and communication resources. Therefore, we aim to provide a guide for other practitioners in robotics to take advantage of the benefits of ADMM. We specifically discuss consensus-style variants of ADMM which are appealing for multi-robot systems due to the robustness to robot failure and the relatively limited processing required by any one robot. Ultimately, we provide an iterative algorithm in which each robot performs a local optimization and some update steps, communicates locally with neighbors, and then repeats until convergence. This structure is similar to classic consensus methods in multi-robot systems.

The main contributions of this work are: (1) We define a class of multi-robot problems which includes the task assignment problem; (2) We describe decentralized algorithms based on ADMM; (3) We discuss the conditions under which the robots will reach the globally optimal solution; and (4) We provide simulation results demonstrating the benefits of ADMM in comparison to state-of- the-art methods for task assignment.

1.1 Related Work

The ADMM framework is well established and studied within the optimization community; see [2] for a review. There has been a recent resurgence of ADMM approaches as many applications in statistics and machine learning with large datasets can be formulated as structured convex optimization problems amenable to parallel processing. Notably, the typical ADMM formulation with a separable objective requires a centralized update step: a central node gathers information from "worker" nodes, a global parameter is updated, and this parameter is communicated to the worker nodes. The worker nodes then perform individual computations and the process repeats. This abstraction is fundamentally at odds with the abstractions in multi-robot applications where no single robot has access to (or gathers) all information. Therefore, we leverage recent work on ADMM [4,13] to develop decentralized algorithms suitable for robotics.

Variations of ADMM have been developed that do not require a central node [10,20]. However, these approaches either require stricter conditions or consider a more specific optimization statement. In contrast, we leverage recent work [4,13] to formulate algorithms for a more general class of problems in robotics.

There are few existing works in robotics that leverage ADMM. Park et al. [17] use the typical ADMM approach for multi-target tracking which requires a central node. Choudhary et al. [7] leverage ADMM as a parallelized optimization solver for distributed map storage and updates in SLAM. Notably, Van Parys and Pipeleers [19] tailor an ADMM approach without a central node to a specific motion planning problem for a team of vehicles and note that their approach has no convergence guarantees.

For task assignment, the Hungarian algorithm [11] is the fastest known centralized algorithm; in contrast, we propose ADMM as a decentralized optimization method for task assignment problems. We specifically consider linear sum assignment problems (LSAPs). Branch and bound methods have been used in the centralized case [15] however there are few decentralized versions. Falsone et al. [8] propose an algorithm to solve a multi-robot mixed-integer linear program (MILP) but require a centralized update step. Testa et al. [18] propose a decentralized cutting plane method but require the objective function to be integer-valued. Lastly, Bürger et al. [3] propose a distributed simplex method for MILPs but do not consider inequality constraints, as we do in this work.

Distributed variants of the Hungarian algorithm have been proposed [6, 9]. Both of these methods recover the centralized performance and we use [6] as a benchmark method. We note that these methods are more complicated to implement and analyze than the methods we develop. Lastly, auction (or market) methods are also popular [5, 12, 14, 21] although some of these methods require shared memory or a centralized coordinator. We also use [5] and [12] as benchmark methods and note that [5] is known to provide sub-optimal assignments.

Our main objective is to develop decentralized ADMM algorithms for task assignment and to demonstrate its competitive performance with other methods. More broadly, we aim to provide an additional tool for solving multi-robot optimization problems.

The remainder of this work is organized as follows. Section 2 introduces the multi-robot task assignment problem and a general optimization statement. We also discuss the necessary problem structure for ADMM methods to converge. In Section 3, we present decentralized algorithms to solve the general optimization problem. Simulations results with comparisons to benchmark methods are presented in Section 4. Concluding remarks are provided in Section 5.

Notation For vectors and matrices, the operators $=$, $\geq$, and $\leq$ are element-wise comparisons. The k-dimensional vector of all ones is $\mathbf{1}_k$ and the $k \times k$ identity matrix is I_k. Square brackets with a subscript refer to elements of a vector or matrix, e.g. $[x]_j$ refers to element j of vector x and $[A]_{i,j}$ refers to element i, j of matrix A. The Hadamard product (element-wise product) of two matrices A and B is denoted by $A \circ B$. The Frobenius norm is indicated by $\|\cdot\|_F$.

2 Problem Statement

We begin by defining the task assignment problem and then discuss a more general optimization statement that represents a range of cooperative multi-robot problems.

2.1 Multi-robot Task Assignment

The multi-robot team consists of n robots that are required to cooperatively complete m tasks. The vector $x_i \in \{0,1\}^m$ describes the tasks that robot i has been assigned. The elements of x_i are binary and $[x_i]_j = 1$ indicates robot i is assigned to task j. Conversely, $[x_i]_j = 0$ means robot i is not assigned to task j. The cost of the assignment of a robot to different tasks is described by the vector $\alpha_i \in \mathbb{R}^m$. We also enforce robots to only be assigned to one task. We consider two variations of an additional constraint,

(Case 1) there are equal number of tasks and robots ($n = m$) and each task is assigned to only one robot (and vice versa);
(Case 2) there are more robots than tasks ($n > m$) and each task must be assigned at least one robot.

The communication network for the team is represented by an undirected graph $G = (V, E)$ with vertex set V and edge set E. Each vertex corresponds to a robot thus $V = \{1, \ldots, n\}$ and an edge (i, j) exists if robot i and robot j are in communication with each other. The *neighbor set* $\mathcal{N}(i) = \{j \in V \mid (i, j) \in E\}$ describes the set of robots in communication with robot i.

Assumption 1 (Connected Network) *We assume the communication network G is* connected*. In other words, there exists a path between every pair of vertices $i, j \in V$ which may consist of more than one edge.*

We note that some prior work assumes a fully-connected graph, in which each robot is connected to every other robot, whereas we assume a less strict condition. The following optimization summarizes the task assignment problem.

Problem 1 (Multi-robot Task Assignment)

$$\underset{x_1,\ldots,x_n \in \mathbb{R}^m}{\text{minimize}} \quad \sum_{i=1}^{n} \alpha_i^T x_i \tag{1a}$$

$$\text{subject to} \quad [x_i]_j \in \{0,1\} \ \forall j \in \{1,\ldots,m\}, i \in V \tag{1b}$$

$$\mathbf{1}_m^T x_i = 1 \ \forall i \in V \tag{1c}$$

$$\underbrace{\sum_{i=1}^{n} x_i = \mathbf{1}_m}_{\text{Case 1}} \ \text{ or } \ \underbrace{\sum_{i=1}^{n} x_i \geq \mathbf{1}_m}_{\text{Case 2}} \tag{1d}$$

Problem 1 is an integer linear program (ILP). The most appropriate solution methods are those that directly consider integer constraints (e.g., branch and bound or cutting plane methods), but it is difficult to produce decentralized versions of these methods. Therefore, we instead consider a linear programming relaxation where the integer constraint is replaced with a linear constraint. After solving the relaxed program, the resulting solution is rounded to an integer solution. In Problem 1, the constraint (1b) is replaced by,

$$[x_i]_j \in \{0,1\} \to 0 \leq [x_i]_j \leq 1 \quad \forall j \in \{1,\ldots,m\}, i \in V. \tag{2}$$

For Case 1, solving the relaxed problem results in an integer-valued solution with no need for rounding. To see this, note that for each vector x_i in Problem 1, the basis of the feasible set is the standard basis $\{e_i \in \mathbb{R}^m \mid 1 \leq i \leq m\}$. By the fundamental theorem of linear programming, the optimal solution $x_i^\star$ either lies at a vertex or on a face of the convex polytope defined by the feasible set. The solution only lies on the polytope face if robots have equal cost of assignment for some (or all) of the tasks which results in multiple optimal assignments with equal objective value. Handling multiple optimal solutions is non-trivial [3, 6] and therefore we only consider problems with a unique optimal solution.

For Case 2, there is no general characterization of when the relaxed solution will be integer-valued. We adopt a simple rounding strategy and show through simulations that this strategy is effective (see Section 4). The goal is to solve Problem 1 in a decentralized fashion. We first discuss a class of problems that include the relaxed formulation of Problem 1.

2.2 General Cooperative Multi-robot Problems

We consider the following statement as a generalized optimization for multi-robot cooperative problems,

Problem 2 (Multi-robot Optimization Problem)

$$\underset{x_1,\ldots,x_n \in \mathbb{R}^m}{\text{minimize}} \quad \sum_{i=1}^{n} f_i(x_i) + g_i(x_i) \quad \text{subject to} \quad \sum_{i=1}^{n} A_i x_i = b \text{ and } \sum_{i=1}^{n} C_i x_i \leq d,$$

with $A_i \in \mathbb{R}^{l_1 \times m}$, $b \in \mathbb{R}^{l_1}$, $C_i \in \mathbb{R}^{l_2 \times m}$, $d \in \mathbb{R}^{l_2}$, $f_i : \mathbb{R}^m \mapsto \mathbb{R}$, *and* $g_i : \mathbb{R}^m \mapsto \mathbb{R}$.

In the objective function of Problem 2, the function f_i encodes the cost function associated with robot i. The function g_i captures regularization or non-smooth components, such as constraint sets on x_i. Let $M_i \in \mathbb{R}^{n \times m}$ be a matrix with row i containing all ones and the remaining entries are zero. Then, Problem 2 is equivalent to the relaxation of Problem 1 when,

$$\text{(Case 1)} \quad A_i = \begin{bmatrix} M_i \\ I_m \end{bmatrix} \in \mathbb{R}^{(n+m)\times m}, \; b = \mathbf{1}_{n+m}, \text{ and } C_i = d = 0; \tag{3}$$

$$\text{(Case 2)} \quad A_i = M_i, \; b = \mathbf{1}_m, \; C_i = -I_m, \text{and } d = -\mathbf{1}_m; \tag{4}$$

and the functions $f_i(x_i)$ and $g_i(x_i)$ are,

$$f_i(x_i) = \alpha_i^T x_i, \qquad g_i(x_i) = \begin{cases} 0 & \text{if } 0 \leq [x_i]_j \leq 1 \; \forall j \in \{1,\ldots,m\}, \\ \infty & \text{otherwise.} \end{cases} \tag{5}$$

In the next section, we develop ADMM-based algorithms for Problem 2 and thus the relaxation of Problem 1 as well. We first list the assumptions required for the guarantees of ADMM methods.

Assumption 2 *The functions f_i and g_i are closed, proper, and strictly convex. For Problem 2, the Lagrangian $\mathcal{L}$ has a saddle point, a unique minimum is obtained, and strong duality holds.*

For the task assignment problem, f_i and g_i satisfy Assumption 2. The relaxation of the binary constraint (2) allows for a convex formulation of the task assignment problem. Otherwise, this constraint produces a non-convex problem which precludes the standard convergence guarantees of ADMM.

3 Decentralized Optimization Methods

We now develop decentralized algorithms to solve Problem 2.

3.1 Primal Problem Approach

Problem 2 can equivalently be expressed with the variables $X_i \in \mathbb{R}^{m\times n}$ with one such variable associated with each robot,

$$\begin{aligned} \underset{\substack{X_1,\dots,X_n\in\mathbb{R}^{m\times n}\\ Y_{ij}\in\mathbb{R}^{m\times n}}}{\text{minimize}} \quad & \sum_{i=1}^{n} \phi_i(X_i) + \psi_i(X_i) \\ \text{subject to} \quad & \sum_{j=1}^{n} A_j X_i e_j = b \text{ and } \sum_{j=1}^{n} C_j X_i e_j \le d, \ \forall i \in V \\ & X_i = Y_{ij} \text{ and } X_j = Y_{ij} \ \forall j \in \mathcal{N}(i), i \in V, \end{aligned} \tag{6}$$

where $e_j \in \mathbb{R}^n$ is the standard basis vector, $\phi_i(X_i) = f_i(X_i e_i)$, and $\psi_i(X_i) = g_i(X_i e_i)$. The quantities A_j, b, C_j, and d are defined in (3) (Case 1) and (4) (Case 2) for task assignment. The variables Y_{ij} enforce agreement of solutions between each robot and its neighbors. We have introduced the functions ϕ_i and ψ_i to maintain the same abstraction as Problem 2: the objective is a sum of functions where each function may only be known by a single robot i. For the primal approach applied to task assignment, each robot reasons about the assignments of all robots in the team and uses only its own assignments in its contribution to the objective value, since $x_i = X_i e_i$ if $X_i = [x_1 \cdots x_n]$. In addition, $\psi_i(X_i)$ can encode more problem structure. For example, in the relaxed task assignment problem, all robot assignments must be in the $[0, 1]$ interval. Therefore, each robot i can constrain all entries of its solution X_i to be within this interval, rather than only column i, to improve convergence.

It suffices to solve (6) in a decentralized fashion and the result will be optimal for Problem 2, as we discuss next, due to the consensus variables Y_{ij}.

Proposition 1 ([13]). *Optimization (6) is equivalent to Problem 2 with the same optimal objective value. If $X_i^\star$ $\forall i \in V$ is optimal for (6) and $x_i^\star$ $\forall i \in V$ is optimal for Problem 2 then $X_i^\star = [x_1^\star \cdots x_n^\star]$ $\forall i \in V$.*

Proof. By Assumption 1, the communication network is connected and all robots have the same solution $X_i = X \ \forall i \in V$ due to the consensus constraints Y_{ij}. Then, (6) simplifies to,

$$\underset{X \in \mathbb{R}^{m\times n}}{\text{minimize}} \ \sum_{i=1}^{n} \phi_i(X) + \psi_i(X) \text{ subject to } \sum_{j=1}^{n} A_j X e_j = b \text{ and } \sum_{j=1}^{n} C_j X e_j \leq d.$$

Let $X = [x_1 \cdots x_n]$ with $x_i \in \mathbb{R}^m$ so $Xe_j = x_j$. By definition of ϕ_i and ψ_i, Problem 2 and (6) are equivalent statements. We also assume there is a unique optimal solution and thus both statements have the same optimal solution. □

The formulation in (2) allows us to apply the ADMM method. First, we form an *augmented Lagrangian* $\mathcal{L}_a$ for (6),

$$\begin{aligned}\mathcal{L}_a =& \sum_{i=1}^{n} \phi_i(X_i) + \psi_i(X_i) + \frac{\rho}{2}\sum_{i=1}^{n}\sum_{j\in\mathcal{N}(i)} \left\|X_i - Y_{ij}\right\|_F^2 + \left\|X_j - Y_{ij}\right\|_F^2 \\ &+ \sum_{i=1}^{n}\sum_{j\in\mathcal{N}(i)} \mathbf{1}_m^T \Big(V_{ij} \circ (X_i - Y_{ij}) + W_{ij} \circ (X_j - Y_{ij})\Big)\mathbf{1}_n.\end{aligned}$$

We use the dual variables $V_{ij}, W_{ij} \in \mathbb{R}^{m\times n}$ for each local agreement constraint but have omitted the other constraints in (6) (i.e., they have not been dualized). In addition, $\rho > 0$ is a penalty parameter on violation of the robot agreement constraint. ADMM is an iterative process that traditionally consists of three steps per iteration. First, $\mathcal{L}_a$ is minimized with respect to (w.r.t.) the variables $X_i \ \forall i \in V$ (subject to the remaining constraints in (6) that were not dualized) considering the other variables Y_{ij}, V_{ij}, W_{ij} to be fixed. A unique set of minimizers is guaranteed to exist by the strict convexity of $\mathcal{L}_a$ and the assumption that Problem 2 is well-defined. Next, $\mathcal{L}_a$ is minimized w.r.t. Y_{ij} with all other variables held constant. Finally, the multipliers V_{ij}, W_{ij} are updated via gradient scent. This process then repeats for subsequent iterations until convergence.

In the following discussion, we use superscript k to indicate iterations of ADMM. Note that $\mathcal{L}_a$ is separable w.r.t. the robot solutions X_i and the variables Y_{ij}. With this decomposition, the update of each solution X_i is,

$$\begin{aligned} X_i^{k+1} = \arg\underset{X_i\in\mathbb{R}^{m\times n}}{\text{minimize}} \quad & \phi_i(X_i) + \psi_i(X_i) + \mathbf{1}_m^T\left((V_{ij}^k + W_{ji}^k)\circ X_i\right)\mathbf{1}_n \\ & + \rho \sum_{j\in\mathcal{N}(i)} \left\| X_i - \frac{X_i^k + X_j^k}{2}\right\|_F^2 \\ \text{subject to} \quad & \sum_{j=1}^{n} A_j X_i e_j = b \text{ and } \sum_{i=1}^{n} C_j X_i e_j \leq d.\end{aligned}$$

The variables Y_{ij} have a simple closed-form update, $Y_{ij}^{k+1} = \frac{1}{2\rho}\left(V_{ij}^k + W_{ij}^k\right) + \frac{1}{2}\left(X_i^{k+1} + X_j^{k+1}\right)$. The dual variable updates are,

$$V_{ij}^{k+1} = V_{ij}^k + \frac{\rho}{2}\left(X_i^{k+1} - Y_{ij}^{k+1}\right), \qquad W_{ij}^{k+1} = W_{ij}^k + \frac{\rho}{2}\left(X_j^{k+1} - Y_{ij}^{k+1}\right).$$

Algorithm 1 Primal Consensus ADMM

```
1: Given: initial $X_i^0 \in \mathbb{R}^{n \times m}$ and $Q_i^0 = 0$ for each robot $i$; penalty parameter $\rho > 0$
2: repeat
3:     for each $i \in V$ do
4:         $Q_i^{k+1}$ = Equation (7) and $X_i^{k+1}$ = Equation (8)
5: until iteration limit reached or robot solutions converge
```

If the dual variables are initialized to zero, then the dual update simplifies [13],

$$Q_i^{k+1} = Q_i^k + \rho \sum_{j \in \mathcal{N}(i)} \big(X_i^k - X_j^k\big), \tag{7}$$

where $Q_i^k = \sum_{j \in \mathcal{N}(i)} V_{ij}^k + W_{ji}^k$. Further, the X_i update for the task assignment problem can be written as,

$$\begin{aligned} X_i^{k+1} = \ & \underset{X_i \in \mathbb{R}^{m \times n}}{\arg\operatorname{minimize}} && \alpha_i^T X_i e_i + \mathbf{1}_m^T \big(Q_i^k \circ X_i\big)\, \mathbf{1}_n + \rho |N(i)| \big\|X_i\big\|_F^2 \\ & && - \rho \mathbf{1}_m^T \Big(\sum_{j \in \mathcal{N}(i)} \big(X_i^k + X_j^k\big) \circ X_i \Big) \mathbf{1}_n \\ & \text{subject to} && \sum_{j=1}^n A_j X_i e_j = b, \ \sum_{j=1}^n C_j X_i e_j \le d, \ \text{and } 0 \le X_i e_i \le 1, \end{aligned} \tag{8}$$

after substituting (5) and simplifying. Equation (8) is a constrained quadratic program (QP) which can be solved efficiently with a general-purpose QP solver. Equations (7) and (8) consist of local updates: each robot stores and updates two variables $Q_i^k, X_i^k \in \mathbb{R}^{m \times n}$ and communicates its solution X_i^k with its neighbors $\mathcal{N}(i)$ at each iteration k. Algorithm 1 summarizes the decentralized algorithm. Each robot is guaranteed to reach the optimal solution.

Proposition 2 ([2, §3.2-3.3], [13]). *Given Assumptions 1 and 2, each robot solution converges to the optimal solution: $X_i^k \to X^\star \ \forall i \in V$ as $k \to \infty$ where $\{x_i^\star \mid i \in V\}$ is the optimal solution of Problem 2 and $X^\star = [x_1^\star \cdots x_n^\star]$.*

Our problem formulation (6) differs from prior work due to the constraints in (8). These constraints do not modify the convergence argument since (8) is well-defined with a non-empty feasible set (Assumption 2). Therefore, we refer to the arguments in Proposition 2 of [13] and omit the proof here. Next, we develop an algorithm based on duality.

3.2 Dual Problem Approach

The dual formulation of Problem 2 is,

$$\underset{\nu \in \mathbb{R}^{l_1}, \lambda \in \mathbb{R}^{l_2}}{\operatorname{minimize}} \ \sum_{i=1}^n \Big(h_i(\nu, \lambda) + \frac{1}{n} \nu^T b + \frac{1}{n} \lambda^T d \Big) \text{ subject to } \lambda \ge 0, \tag{9}$$

where $h_i(\nu,\lambda) = \underset{x_i \in \mathbb{R}^m}{\text{maximize}} - f_i(x_i) - g_i(x_i) - \nu^T A_i x_i - \lambda^T C_i x_i$, after simplification of the standard dual problem derivation. An equivalent problem amenable to decentralization is,

$$\begin{array}{ll} \underset{\substack{\nu_1,\ldots,\nu_n \in \mathbb{R}^{l_1} \\ \lambda_1,\ldots,\lambda_n \in \mathbb{R}^{l_2} \\ t_{ij} \in \mathbb{R}^{l_1}, u_{ij} \in \mathbb{R}^{l_2}}}{\text{minimize}} & \displaystyle\sum_{i=1}^{n} \left(h_i(\nu_i, \lambda_i) + \frac{1}{n}\nu_i^T b + \frac{1}{n}\lambda_i^T d \right) \\ \text{subject to} & \nu_i = t_{ij}, \nu_j = t_{ij}, \lambda_i = u_{ij}, \text{ and } \lambda_j = u_{ij} \ \forall j \in \mathcal{N}(i), i \in V \\ & \lambda_i \geq 0 \ \forall i \in V \end{array} \tag{10}$$

Each robot reasons about the variables ν_i and λ_i, from which its decision x_i is recovered using h_i. Similar to Proposition 1, (9) and (10) are equivalent.

Corollary 1. *Statements (9) and (10) have the same optimal solution.*

We now follow the same derivation as the primal approach to develop the individual update steps for each robot. An augmented Lagrangian is formed for (10) without the constraint $\lambda_i \geq 0$ and this function is again separable. Each robot individually solves the following optimization to update ν_i and λ_i,

$$\begin{array}{ll} \underset{\nu_i \in \mathbb{R}^{l_1}, \lambda_i \in \mathbb{R}^{l_2}}{\text{minimize}} & h_i(\nu_i, \lambda_i) + \frac{1}{n}\nu_i^T b + \frac{1}{n}\lambda_i^T d + \nu_i^T (v_{1,ij}^k + v_{2,ji}^k) + \lambda_i^T (u_{1,ij}^k + u_{2,ji}^k) \\ & \displaystyle + \rho \sum_{j \in \mathcal{N}(i)} \left\| \nu_i - \frac{\nu_i^{k-1} + \nu_j^{k-1}}{2} \right\|_2^2 + \left\| \lambda_i - \frac{\lambda_i^{k-1} + \lambda_j^{k-1}}{2} \right\|_2^2 \\ \text{subject to } \lambda_i \geq 0, & \end{array} \tag{11}$$

where the quantities $v_{1,ij}, v_{2,ij} \in \mathbb{R}^{l_1}$ and $w_{1,ij}, w_{2,ij} \in \mathbb{R}^{l_1}$ are dual variables for the variables t_{ij} and u_{ij}, respectively. The dual variable gradient ascent yields,

$$\begin{aligned} v_{1,ij}^{k+1} &= v_{1,ij}^k + \frac{\rho}{2}\left(\nu_i^{k-1} - \nu_j^{k-1}\right), & v_{2,ij}^{k+1} &= v_{2,ij}^k + \frac{\rho}{2}\left(\nu_j^{k-1} - \nu_i^{k-1}\right), \\ w_{1,ij}^{k+1} &= w_{1,ij}^k + \frac{\rho}{2}\left(\lambda_i^{k-1} - \lambda_j^{k-1}\right), & w_{2,ij}^{k+1} &= w_{2,ij}^k + \frac{\rho}{2}\left(\lambda_j^{k-1} - \lambda_i^{k-1}\right). \end{aligned}$$

Let $q_i^k = \sum_{j \in \mathcal{N}(i)} v_{1,ij}^k + v_{2,ji}^k$ and $r_i^k = \sum_{j \in \mathcal{N}(i)} w_{1,ij}^k + w_{2,ji}^k$. Then the previous updates are replaced by,

$$q_i^k = q_i^{k-1} + \rho \sum_{j \in \mathcal{N}(i)} \nu_i^{k-1} - \nu_j^{k-1}, \qquad r_i^k = r_i^{k-1} + \rho \sum_{j \in \mathcal{N}(i)} \lambda_i^{k-1} - \lambda_j^{k-1}, \tag{12}$$

if all variables $v_{1,ij}, v_{2,ij}, w_{1,ij}, w_{2,ij}$ are initialized to zero [13]. We again omit the variables t_{ij}, u_{ij} as they are redundant given the other variables ν_i, λ_i, x_i.

We note that solving (11) is difficult for task assignment as each robot must also solve the optimization,

$$h_i(\nu_i, \lambda_i) = \underset{x_i \in \mathbb{R}^m}{\text{maximize}} - \alpha_i^T x_i - \nu_i^T A_i x_i - \lambda_i^T C_i x_i \text{ subject to } 0 \leq x_i \leq 1,$$

using (5) and either (3) (Case 1) or (4) (Case 2). Therefore, we leverage problem structure to produce tractable updates for ν_i, λ_i, and x_i. The objective in (11) is concave in x_i given values of ν_i, λ_i and is convex in the augmented variable $\begin{bmatrix}\nu_i \ \lambda_i\end{bmatrix}^T$ given a value of x_i. Therefore, the minimax theorem [1, §2.6] can be applied to solve (11) by considering the maximization first along with the existence of a saddle point. For clarity in the following discussion, we define,

$$\theta(w, z, s, \epsilon_i) = \frac{1}{\rho}w - \frac{1}{\rho n}z - \frac{1}{\rho}s + \sum_{j \in \mathcal{N}(i)} \epsilon_i + \epsilon_j.$$

Applying the minimax theorem, substituting the definition of h_i, and completing the square for both ν_i and λ_i modifies the objective of (11) to,

$$\begin{aligned}
&\underset{x_i \in \mathbb{R}^m}{\text{maximize}} \ \underset{\nu_i \in \mathbb{R}^{l_1}, \lambda_i \in \mathbb{R}^{l_2}}{\text{minimize}} \ -\alpha_i^T x_i \\
&+ \rho|\mathcal{N}(i)| \left\| \nu_i - \frac{1}{2|\mathcal{N}(i)|}\theta(A_i x_i, b, q_i^k, \nu_i^{k-1}) \right\|_2^2 - \frac{\rho}{4|\mathcal{N}(i)|}\left\| \theta(A_i x_i, b, q_i^k, \nu_i^{k-1}) \right\|_2^2 \\
&+ \rho|\mathcal{N}(i)| \left\| \lambda_i - \frac{1}{2|\mathcal{N}(i)|}\theta(C_i x_i, d, r_i^k, \lambda_i^{k-1}) \right\|_2^2 - \frac{\rho}{4|\mathcal{N}(i)|}\left\| \theta(C_i x_i, d, r_i^k, \lambda_i^{k-1}) \right\|_2^2 \\
&\text{subject to } 0 \le x_i \le 1.
\end{aligned}$$

The inner minimization is separable in ν_i and λ_i given $x_i = x_i^k$, which leads to,

$$\nu_i^k = \frac{1}{2|\mathcal{N}(i)|}\theta(A_i x_i^k, b, q_i^k, \nu_i^{k-1}), \tag{13}$$

$$\lambda_i^k = \underset{\lambda_i \in \mathbb{R}^{l_2}}{\arg\text{minimize}} \ \left\| \lambda_i - \frac{1}{2|\mathcal{N}(i)|}\theta(C_i x_i^k, d, r_i^k, \lambda_i^{k-1}) \right\|_2^2 \text{ subject to } \lambda_i \ge 0. \tag{14}$$

The λ_i^k update has a simple solution: set $\lambda_i^k = \frac{1}{2|\mathcal{N}(i)|}\theta(C_i x_i^k, d, r_i^k, \lambda_i^{k-1})$ and change any elements of λ_i^k that are negative to zero. Given $\nu_i = \nu_i^k$ and $\lambda_i = \lambda_i^k$, the outer maximization is then,

$$\begin{aligned}
x_i^k = &\underset{x_i \in \mathbb{R}^m}{\arg\text{minimize}} \quad \alpha_i^T x_i + \frac{\rho}{4|\mathcal{N}(i)|}\left\| \theta(A_i x_i, b, q_i^k, \nu_i^{k-1}) \right\|_2^2 + \frac{1}{2}(\lambda_i^k)^T C_i x_i \\
&\text{subject to} \qquad 0 \le x_i \le 1,
\end{aligned} \tag{15}$$

after simplification; this form is also a constrained QP which can be solved efficiently. The solution of (11) is found by choosing a variable update order: the primal variable is updated using λ_i^{k-1} and then the dual variables ν_i^k,λ_i^k are updated with x_i^k. Algorithm 2 summarizes the resulting dual consensus method.

Proposition 3 ([4]). *Given Assumptions 1 and 2, each robot solution converges to the optimal solution: $\nu_i^k \to \nu^\star, \lambda_i^k \to \lambda^\star \ \forall i \in V$ as $k \to \infty$ where $\nu^\star, \lambda^\star$ is optimal for (9). Further, any limit point of $\{x_i^k \mid i \in V\}$ is optimal for Problem 2.*

Our formulation (10) differs from [4] due to the dual variables λ_i introduced for the inequality constraints in Problem 2. Using an augmented variable $\begin{bmatrix}\nu_i \ \lambda_i\end{bmatrix}^T$

Algorithm 2 Dual Consensus ADMM

```
1: Given: initial $x_i^0 \in \mathbb{R}^m$, $\nu_i^0 \in \mathbb{R}^{l_1}$, $\lambda_i^0 \in \mathbb{R}^{l_2}$ and $q_i^0 = r_i^0 = 0$ for each robot $i$;
   penalty parameter $\rho > 0$
2: repeat
3:     for each $i \in V$ do
4:         Update $q_i^k$, $r_i^k$ using Equation (12)
5:         $x_i^k$ = Equation (15), $\nu_i^k$ = Equation (13), and $\lambda_i^k$ = Equation (14)
6: until iteration limit reached or robot solutions converge
```

yields a comparable formulation as [4, §4]. The additional constraint $\lambda_i \geq 0$ does not modify the convergence argument as it is satisfied at each iteration by the update equation (14). We refer to Theorem 2 in [4] and omit the details here.

In many applications, it is desirable to eliminate optimization statements like (15) in favor of closed-form solutions. We now use proximal gradients [16] to achieve this. A first-order proximal update around x_i^{k-1} of the objective in (15) leads to the modified objective,

$$\left[\alpha_i + \frac{1}{2|\mathcal{N}(i)|} A_i^T \theta(A_i x_i^{k-1}, b, q_i^k, \nu_i^{k-1}) + \frac{1}{2} C_i^T \lambda_i^k \right]^T (x_i - x_i^{k-1}) + \frac{\gamma_i}{2} \left\| x_i - x_i^{k-1} \right\|_2^2,$$

where $\gamma_i > 0$ is a penalty parameter. This objective leads to the optimization,

$$x_i^k = \arg\underset{x_i \in \mathbb{R}^m}{\text{minimize}} \; \frac{\gamma_i}{2} \left\| x_i - x_i^{k-1} + \frac{1}{\gamma_i}\alpha_i + \frac{1}{2\gamma_i |\mathcal{N}(i)|} A_i^T \theta(A_i x_i^{k-1}, b, q_i^k, \nu_i^{k-1}) + \frac{1}{2\gamma_i} C_i^T \lambda_i^k \right\|_2^2 + g_i(x_i),$$

since the additional terms are constant w.r.t. x_i. The constraint in (15) is now enforced by g_i (5) in the objective as this form resembles the proximal operator, $\text{prox}(z; \gamma_i, g_i) = \arg\underset{y \in \mathbb{R}^m}{\text{minimize}} \; \frac{\gamma_i}{2} \|y - z\|_2^2 + g_i(y)$ [16]. This projection has a simple solution for (5): each element of z is clipped to the interval $[0, 1]$. Thus, the x_i update (15) is replaced by the following strategy: set $x_i^k = x_i^-$ where,

$$x_i^- = x_i^{k-1} - \frac{1}{\gamma_i}\alpha_i - \frac{1}{2\gamma_i |\mathcal{N}(i)|} A_i^T \theta(A_i x_i^{k-1}, b, q_i^k, \nu_i^{k-1}) - \frac{1}{2\gamma_i} C_i^T \lambda_i^k.$$

Then, change elements of x_i^k that are less than zero to zero and change elements greater than one to one. Convergence of Algorithm 2 with this modified update is guaranteed if the parameters γ_i are sufficiently large [4]. A similar idea of proximal updates for the primal approach is also possible. In the next section, we present simulations comparing the performance of the ADMM methods.

4 Experiments

In the following discussion, we refer to the algorithms we previously developed as ADMM task assignment (ADMM-TA). First, we examine the convergence

Table 1. Average number of iterations for convergence for the task assignment problem with 20 robots and 20 tasks over 50 trials of randomly generated costs.

	Primal	Dual	Inexact Dual
Average Iterations	28	159	387

rate of the ADMM-TA methods when using the primal (Algorithm 1), dual (Algorithm 2), and inexact dual (proximal gradient modification) formulations. For this study, we use these methods to solve the relaxation of Problem 1 with 20 robots and 20 tasks. Table 1 shows the average number of iterations required to reach 0.1% of the optimal centralized objective value. Convergence is slowest on the inexact dual ADMM-TA method but benefits from having very simple update equations. The primal ADMM-TA method achieves the fastest convergence rate and therefore we use this method for the remaining simulation studies. We also exploit structure for the primal method by constraining all entries of each robot's solution to be in the interval $[0, 1]$.

4.1 Convergence of ADMM to Optimal Solution

Next, we examine the convergence of the primal ADMM-TA method to the optimal centralized solution of the task assignment problem. The Hungarian method is used to solve the task assignment problem without relaxations [11]. We consider 50 robots and 50 tasks using a fully-connected graph where the assignment costs were randomly generated. Figure 1 (left) shows the convergence to the optimal cost. During the first few iterations, each robot selects tasks with minimal cost without consensus on the global assignments. The consensus constraints enforce agreement on the assignments for all robots and consequently, each robot adjusts its solution to achieve global consensus. After convergence, the assignments are consistent with the problem constraints.

4.2 Performance on Various Graph Topologies

We also compare the performance of our method on different connected communication networks. The worst-case scenario for consensus methods is linear-chain graphs since information propagates slowly between the first and last robot. We compare the convergence rate of linear-chain to randomly generated and fully-connected graphs and also show that all topologies converge to the centralized optimal cost. We consider the case of 50 robots and 30 tasks, which our method easily accommodates, and randomly generated costs. Figure 1 (right) compares the convergence rates of the different topologies. The ADMM-TA estimate converges to the optimal solution on all the communication graphs. Notably, the rate of convergence of the ADMM-TA solution to the optimal solution does not depend significantly on the graph topology with the exception of the linear-chain graph. Even in the worst-case graph topology the ADMM-TA method converges in about 200 iterations. On more general graph topologies, convergence of the

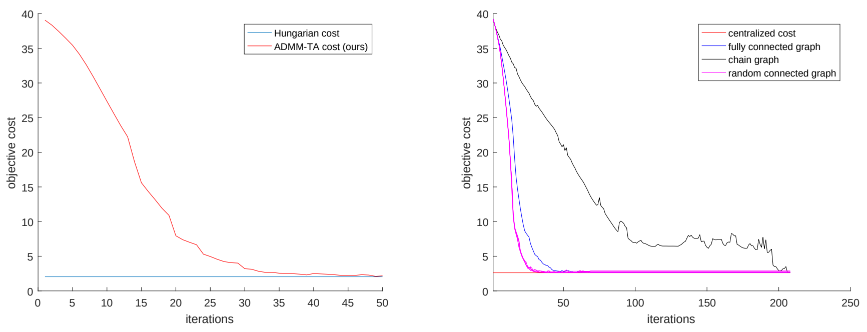

Fig. 1. (left) The ADMM-TA estimate converges to the Hungarian solution of the task assignment problem within 50 iterations for 50 robots and tasks. (right) Convergence of the ADMM-TA solution to the centralized solution on different communication graphs. The slowest rate of convergence is observed on the linear-chain graph.

ADMM-TA method requires roughly 30 iterations. We noticed faster convergence to the optimal solution on randomly-generated connected graphs which is counter-intuitive as typically the fully-connected graph topology produces the fastest convergence rate. We plan to investigate this behavior in future work.

4.3 Comparison with Benchmark Methods

We now compare the scaling of our ADMM-TA approach to other distributed methods for task assignment with respect to number of robots. For each team size, we use an equal number of tasks to as most prior work is designed for this case. We used the consensus-based auction approach (CBAA) [5], market-based consensus [12], and a distributed Hungarian method [6]. The CBAA method does not guarantee optimality after it terminates; in all of the trials we considered, CBAA did not give the optimal solution. In contrast, the market-based consensus and distributed Hungarian methods are optimal. Figure 2 (left) shows the number of iterations required by each method to converge (by their own criteria) on a fully-connected graph. For ADMM-TA, we again check if the cost is within 0.1% of the optimal cost to provide a worst-case scaling trend. ADMM-TA scales better than the market-consensus method but worse than CBAA, although we emphasize that CBAA is sub-optimal. In the distributed Hungarian method, each robot shares its local cost information to create a bipartite graph required by the Hungarian method. Consequently, the distributed Hungarian method produces the optimal estimate in one iteration as all robots receive the cost information of all other robots within one communication round.

Next, we compared the performance of our method to the distributed Hungarian method on randomly-generated connected graphs. We did not use the CBAA method since it did not produce optimal estimates as previously discussed. In addition, we did not include the market-consensus method as the algorithm was

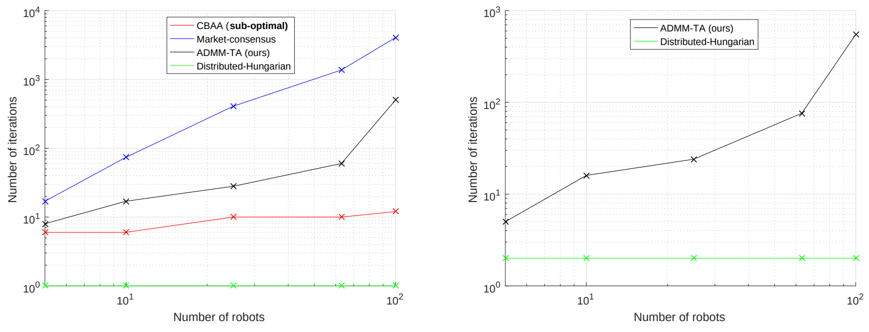

Fig. 2. (left) Comparison of the ADMM-TA method to other distributed methods for task assignment on a fully-connected graph. The ADMM-TA method produces the optimal solution after fewer iterations compared to the CBAA (sub-optimal) and market-consensus methods. (right) Comparison of the ADMM-TA method to the distributed Hungarian method on randomly-generated connected graphs.

not amenable to changes for arbitrary communication graphs. Figure 2 (right) shows the number of iterations required for each method to produce the optimal task assignments. The distributed Hungarian method once again quickly reaches the optimal solution while the ADMM-TA method requires roughly the same number of iterations as the fully-connected graph case. We note that the distributed Hungarian method is less general and requires more information to be communicated per iteration compared to ADMM-TA. In addition, ADMM-TA is a much simpler algorithm to implement and analyze.

4.4 Case Study: Persistent Surveillance

We have focused on single (or one-shot) assignments in the preceding studies. Realistic multi-robot systems might require assignments to be made periodically as tasks are accomplished or as the environment changes. One such situation is persistent surveillance subject to battery constraints. We consider the case in which a group of robots with a fully-connected communication graph must maintain coverage over a region while periodically swapping between surveillance paths and charging stations to ensure that no batteries fully discharge. To accomplish this objective, each robot computes a cost value based on battery levels and travel distances. To ensure persistent coverage, robots at surveillance stations can only charge if another robot chooses to swap with the surveillance robot. We consider ten robots maintaining coverage of two surveillance stations. At each assignment episode, we compute the assignments using our proposed ADMM-TA method and compare the resulting cost to the optimal cost obtained from the Hungarian method. Figure 3 shows the assignment costs over 1000 assignment episodes and the ADMM-TA method converges to an assignment with the optimal cost at every episode. As the simulation runs, the robots reach an oscillating pattern of swapping between surveillance and charging stations, resulting in a limit cycle of the total cost. See the supplementary video for additional details.

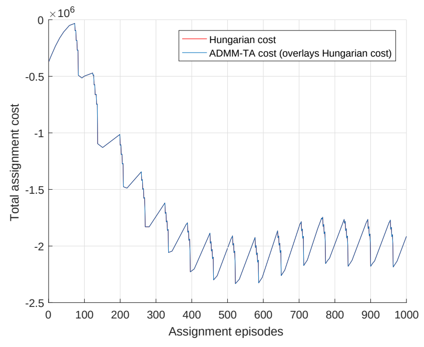

Fig. 3. The ADMM-TA method on a surveillance task with ten robots and two surveillance stations. The ADMM-TA method produces the optimal cost at all assignment episodes and the costs reach a limit cycle after equilibrium is attained.

5 Conclusion

Task assignment problems are prevalent in multi-robot search and rescue, manufacturing, exploration, and many others. In general, multi-robot problems often require a method of coordination into groups. We have proposed decentralized algorithms for solving task assignment problems and for solving a more general description of cooperative robotics problems. Our approach is competitive with current methods for task assignment and is relatively simple to implement. We also discuss a simple extension for practical problems which require task assignments over multiple episodes. Future work will focus on extending our framework to consider non-linear constraints and formulating problem statements which directly account for multi-episode assignments.

Acknowledgements This work was supported in part by DARPA YFA award D18AP00064, NSF CAREER award IIS-1646921, ONR award N00014-18-1-2830, and a NDSEG Fellowship. We are grateful for this support.

References

1. Bertsekas, D.: Convex analysis and optimization. Athena Scientific, Belmont, Mass (2003)
2. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine Learning 3(1), 1–122 (2011)

3. Bürger, M., Notarstefano, G., Bullo, F., Allgöwer, F.: A distributed simplex algorithm for degenerate linear programs and multi-agent assignments. Automatica 48(9), 2298–2304 (2012)
4. Chang, T., Hong, M., Wang, X.: Multi-agent distributed optimization via inexact consensus ADMM. IEEE Transactions on Signal Processing 63(2), 482–497 (2015)
5. Choi, H., Brunet, L., How, J.P.: Consensus-based decentralized auctions for robust task allocation. IEEE Transactions on Robotics 25(4), 912–926 (2009)
6. Chopra, S., Notarstefano, G., Rice, M., Egerstedt, M.: A distributed version of the Hungarian method for multirobot assignment. IEEE Transactions on Robotics 33(4), 932–947 (2017)
7. Choudhary, S., Carlone, L., Christensen, H.I., Dellaert, F.: Exactly sparse memory efficient SLAM using the multi-block alternating direction method of multipliers. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 1349–1356 (2015)
8. Falsone, A., Margellos, K., Prandini, M.: A decentralized approach to multi-agent MILPs: Finite-time feasibility and performance guarantees. Automatica 103, 141–150 (2019)
9. Giordani, S., Lujak, M., Martinelli, F.: A distributed algorithm for the multi-robot task allocation problem. In: García-Pedrajas, N., Herrera, F., Fyfe, C., Benítez, J.M., Ali, M. (eds.) Trends in Applied Intelligent Systems. pp. 721–730. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
10. Hosseini, S., Chapman, A., Mesbahi, M.: Online distributed ADMM via dual averaging. In: 53rd IEEE Conference on Decision and Control (CDC). pp. 904–909 (2014)
11. Kuhn, H.W.: The Hungarian method for the assignment problem. Naval Research Logistics Quarterly 2(1-2), 83–97 (1955)
12. Liu, L., Shell, D.: Optimal market-based multi-robot task allocation via strategic pricing. In: Proceedings of Robotics: Science and Systems (2013)
13. Mateos, G., Bazerque, J.A., Giannakis, G.B.: Distributed sparse linear regression. IEEE Transactions on Signal Processing 58(10), 5262–5276 (2010)
14. Michael, N., Zavlanos, M.M., Kumar, V., Pappas, G.J.: Distributed multi-robot task assignment and formation control. In: 2008 IEEE International Conference on Robotics and Automation. pp. 128–133 (2008)
15. Morrison, D.R., Jacobson, S.H., Sauppe, J.J., Sewell, E.C.: Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. Discrete Optimization 19, 79–102 (2016)
16. Parikh, N., Boyd, S.: Proximal algorithms. Foundations and Trends® in Optimization 1(3), 127–239 (2014)
17. Park, S., Ha, J., Cho, D., Choi, H.: A distributed ADMM approach to informative trajectory planning for multi-target tracking (2018)
18. Testa, A., Rucco, A., Notarstefano, G.: A finite-time cutting plane algorithm for distributed mixed integer linear programming. In: 2017 IEEE 56th Annual Conference on Decision and Control (CDC). pp. 3847–3852 (2017)
19. Van Parys, R., Pipeleers, G.: Online distributed motion planning for multi-vehicle systems. In: 2016 European Control Conference (ECC). pp. 1580–1585 (2016)
20. Wei, E., Ozdaglar, A.: Distributed alternating direction method of multipliers. In: 51st IEEE Conference on Decision and Control (CDC). pp. 5445–5450 (2012)
21. Zavlanos, M.M., Spesivtsev, L., Pappas, G.J.: A distributed auction algorithm for the assignment problem. In: 2008 47th IEEE Conference on Decision and Control. pp. 1212–1217 (2008)