Coupled Task Assignment for Hierarchical Multi-Agent Systems in Disaster Response Missions: A Nested Hungarian Approach

Ahmed Khalil*, Yoonjae Lee†, and Efstathios Bakolas‡
The University of Texas at Austin, Austin, TX, 78712

Gregory M. Gremillion§
U.S. Army DEVCOM Army Research Laboratory, Adelphi, MD, 20783

This work formulates a novel variant of the classical linear task assignment problem, which we refer to as the coupled linear task assignment (CLTA) problem. This formulation models the task assignment process involving two-echelon agents and tasks, where the goal is to optimally assign tasks to agents within the same echelon to maximize a global utility while complying with the coupling constraints imposed by the problem's hierarchical structure. After discussing the problem's inherent properties that prevent the use of standard relaxation techniques, we propose a variant of the Hungarian algorithm, the nested Hungarian algorithm, that solves the problem exactly in polynomial time. We demonstrate the superior performance of the proposed algorithm by benchmarking it against seven open-source and commercial MILP solvers. Finally, we apply our proposed approach to a fictitious disaster response scenario motivated by the January 2025 Southern California wildfires in the Greater Los Angeles area.

I. Nomenclature

$\mathbf{0}$	=	Zero vector or matrix
$\mathbf{1}$	=	One vector or matrix
$\mathcal{A}_u$	=	Set of upper-level agents
$\mathcal{T}_u$	=	Set of upper-level tasks
$\mathcal{A}_\ell$	=	Set of lower-level agents
$\mathcal{T}_\ell$	=	Set of lower-level tasks
n_u	=	Number of upper-level agents, where $n_u = \lvert\mathcal{A}_u\rvert$
m_u	=	Number of upper-level tasks, where $m_u = \lvert\mathcal{T}_u\rvert$
n_ℓ	=	Number of lower-level agents, where $n_\ell = \lvert\mathcal{A}_\ell\rvert$
m_ℓ	=	Number of lower-level tasks, where $m_\ell = \lvert\mathcal{T}_\ell\rvert$
$\mathcal{A}_\ell^i$	=	Set of subagents of upper-level agent i, where $\mathcal{A}_\ell^i \subseteq \mathcal{A}_\ell$
$\mathcal{T}_\ell^j$	=	Set of subtasks of upper-level task j, where $\mathcal{T}_\ell^j \subseteq \mathcal{T}_\ell$
$\mathcal{A}_u^k$	=	Set of superagents of lower-level agent k, where $\mathcal{A}_u^k \subseteq \mathcal{A}_u$
$\mathcal{T}_u^l$	=	Set of supertasks of lower-level task l, where $\mathcal{T}_u^l \subseteq \mathcal{T}_u$
c_{ij}	=	Utility of assigning upper-level agent i to upper-level task j
d_{kl}	=	Utility of assigning lower-level agent k to lower-level task l
x_{ij}	=	Binary assignment variable for upper-level assignment between agent i and task j
y_{kl}	=	Binary assignment variable for lower-level assignment between agent k and task l

*Graduate Student, Department of Aerospace Engineering and Engineering Mechanics, Email: akhalil@utexas.edu
†Ph.D. Candidate, Department of Aerospace Engineering and Engineering Mechanics, Email: yol033@utexas.edu
‡Associate Professor, Department of Aerospace Engineering and Engineering Mechanics, Associate Fellow AIAA, Email: bakolas@austin.utexas.edu
§Researcher, Humans in Complex Systems Division, Email: gregory.m.gremillion.civ@army.mil

II. Introduction

Disaster response scenarios require rapid, coordinated action across different geographical areas by multiple agents to limit damage, preserve lives, and safeguard critical infrastructure [1, 2]. Unmanned aerial vehicles (UAVs), also known as drones, have shown great promise in such conditions by aiding or replacing human responders in dangerous and hard-to-reach areas. Their successful deployment in numerous disaster relief efforts highlights their ability to perform tasks that would be difficult or impossible for humans, such as surveying damage, locating survivors, and delivering supplies [3]. Some of the advantages of drones include fast deployment, high mobility, and the ability to hover or maneuver in wreckage and confined spaces, enabling quick data collection and aid delivery in time-critical situations [3]. Indeed, UAVs have been used to locate earthquake victims and provide real-time imagery to rescue teams [4]. Recent work has proposed a two-echelon vehicle routing problem involving trucks and drones to support post-disaster humanitarian operations [5]. Such examples signify the critical role of multi-UAV systems in disaster management and motivate research into the optimal coordination of these heterogeneous systems.

Given the high stakes in disaster response and the need to consider complex couplings (*e.g.*, how drones and aircraft can cooperate to complete a task), in this work, we focus on the *coupled task assignment problem for hierarchical systems*. Rather than treating a disaster response as a monolithic set of tasks, the hierarchical task assignment formulation decomposes the mission into subtasks assigned at different levels (*e.g.*, high-level objectives for crewed aircraft or large UAVs and lower-level tasks for drone teams). This enables us to explicitly maximize a task utility function for the entire system, resulting in an assignment that optimally coordinates the two tiers of agents. In the disaster response model adopted in this paper, aircraft equipped with fleets of specialized drones are assigned to operate in designated areas based on both the planes' and drones' capabilities. Each aircraft differs in terms of speed, altitude range, and flight endurance, influencing how quickly it can reach specific locations and the types of terrain it can navigate. Higher-altitude planes can access distant areas faster, while low-altitude aircraft are better suited for regions with complex terrain or obstacles. Additionally, aircraft may originate from different locations, affecting their response times and the availability of resources they can collect en route to the disaster area. Figure 1 shows a fictitious instance of the disaster response scenario considered in this paper.

Once deployed, each aircraft releases its drones to perform specific tasks tailored to the needs of the disaster zone. The drones are equipped with distinct capabilities: water-fighting drones for fire suppression, lighting drones to illuminate areas for nighttime operations, medical drones for delivering supplies and first aid, and camera-equipped drones for real-time surveillance and search-and-rescue missions. By leveraging the unique capabilities of both aircraft and drones, the framework aims to maximize coverage, minimize response time, and deliver precise interventions tailored to the situation at hand. The challenge lies in matching the right aircraft to the appropriate areas and allocating drone tasks effectively, ensuring that resources are optimally utilized. To model this problem, we assume that the utility functions for both levels are linear.

Mixed-integer linear programming (MILP) provides a powerful framework for optimal decision-making in multi-UAV task assignment problems. In a typical MILP formulation of the task assignment problem, binary decision variables can represent whether a given agent (*e.g.*, aircraft or drone) is assigned to a specific task, while linear constraints enforce mission-related specifications (*e.g.*, one agent per task, capacity and time windows, etc.). This approach translates the assignment problem into an optimization model, where the objective is often to maximize the overall mission utility (or minimize operational cost). For example, a MILP approach was adopted for multi-robot task allocation in an unknown environment [6]. A MILP approach was also undertaken to optimally assign and schedule coordinated air-to-ground tasks for multiple air vehicles [7]. MILP has also been used to efficiently assign vehicles to suppress enemy air defense missions or intercept enemy incoming missiles [8, 9]. Additionally, MILP has been used to address task allocation and trajectory planning in the presence of obstacles and connectivity constraints [10]. In addition to MILP, a variety of alternative approaches have been explored to solve task assignment problems, such as auctions [11, 12], decentralized variants of the Hungarian algorithm [13], potential game-theoretic approaches [14], and the consensus alternating direction methods of multipliers (ADMM) [15], to name a few. Recent work also highlights how communication fidelity and delays impact the quality of task assignment decision-making in robotic networks [16].

In this work, we formalize the coupled linear task assignment (CLTA) problem, which links two linear assignment layers through additional constraints that capture the hierarchical dependencies between aircraft and drones. These couplings prevent the use of standard relaxation methods, such as linear programming relaxations. We therefore develop a variant of the Hungarian algorithm [17] that solves the CLTA problem optimally in polynomial time. We benchmark our proposed algorithm against several off-the-shelf MILP solvers. Additionally, we showcase how our methodology can be applied to fictitious simulations based on the January 2025 Southern California wildfires.

The rest of this paper is structured as follows. In Section III, we formulate the novel CLTA problem for hierarchical

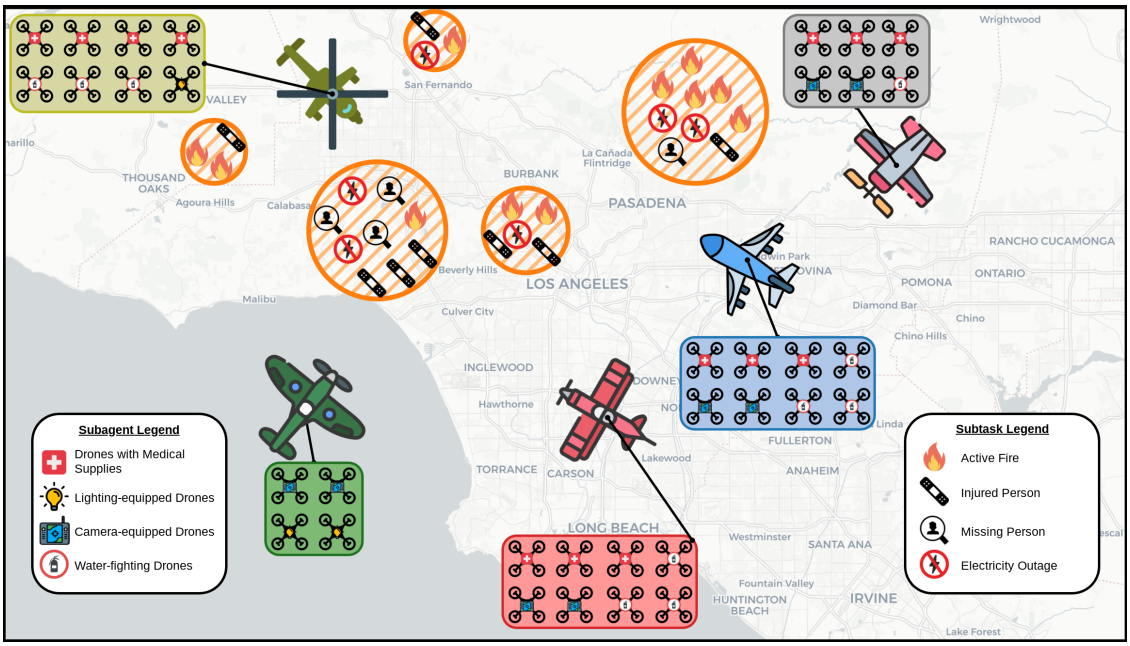


Figure 1 Pictorial description of a hierarchical multi-UAV system in a disaster response mission *(not to scale)*

multi-UAV systems and discuss its inherent properties that prevent the use of standard relaxation techniques. Section V presents the nested Hungarian algorithm, which optimally solves the CLTA problem in polynomial time. Numerical simulation results are demonstrated in Section VI. Finally, VII concludes the paper with directions for future research.

III. Coupled Linear Task Assignment

This section presents a mathematical formulation of the coupled task assignment problem for hierarchical groups of agents and tasks. To generalize our formulation to other real-world problems, we will refer to the aircraft as upper-level agents and the drones as lower-level agents. The corresponding tasks are referred to as upper-level tasks and lower-level tasks. Throughout, we will use the term *entity* to refer to either an agent or a task.

A. Problem Formulation

The coupled linear task assignment (CLTA) problem considers n_u upper-level agents and m_u upper-level tasks, as well as n_ℓ lower-level agents and m_ℓ lower-level tasks. Let $\mathcal{A}_u = \{1, 2, \ldots, n_u\}$, $\mathcal{T}_u = \{1, 2, \ldots, m_u\}$, $\mathcal{A}_\ell = \{1, 2, \ldots, n_\ell\}$, and $\mathcal{T}_\ell = \{1, 2, \ldots, m_\ell\}$ represent the sets of indices of the upper-level agents, upper-level tasks, lower-level agents, and lower-level tasks, respectively. Furthermore, let $\mathcal{E}_u \subseteq \mathcal{A}_u \times \mathcal{T}_u$ and $\mathcal{E}_\ell \subseteq \mathcal{A}_\ell \times \mathcal{T}_\ell$ denote the sets of feasible upper-level agent-task pairs and lower-level agent-task pairs, respectively.

Each upper-level agent i has a nonempty associated set $\mathcal{A}_\ell^i \subseteq \mathcal{A}_\ell$ of subagents (*i.e.*, lower-level agents) that are coupled to i, and each upper-level task j has a nonempty associated set $\mathcal{T}_\ell^j \subseteq \mathcal{T}_\ell$ of subtasks (*i.e.*, lower-level tasks) that are coupled to j. In this work, we assume a partition structure for the couplings. Formally,

$$\bigcup_{i \in \mathcal{A}_u} \mathcal{A}_\ell^i = \mathcal{A}_\ell, \qquad \mathcal{A}_\ell^i \cap \mathcal{A}_\ell^{i'} = \emptyset, \quad \forall i, i' \in \mathcal{A}_u,\ i \neq i',$$
$$\bigcup_{j \in \mathcal{T}_u} \mathcal{T}_\ell^j = \mathcal{T}_\ell, \qquad \mathcal{T}_\ell^j \cap \mathcal{T}_\ell^{j'} = \emptyset, \quad \forall j, j' \in \mathcal{T}_u,\ j \neq j'.$$

The coupling is imposed such that a subagent can only choose a subtask if the corresponding superagent has also chosen the relevant supertask. Specifically, given an upper-level agent-task matching $\mathcal{M}_u \subset \mathcal{E}_u$, the corresponding set of admissible lower-level agent-task pairs is defined by

$$\mathcal{E}_\ell(\mathcal{M}_u) = \{(k, l) \in \mathcal{E}_\ell : (i_k, j_l) \in \mathcal{M}_u\}, \tag{1}$$

where i_k denotes the index of the superagent of k and j_l denotes the index of the supertask of l.

Lastly, let $c : \mathcal{E}_u \to \mathbb{R}_+$ and $d : \mathcal{E}_\ell \to \mathbb{R}_+$ denote the upper-level and lower-level utility functions, respectively. The objective of the CLTA problem is to find the pair of upper-level matching $\mathcal{M}_u^\star \subset \mathcal{E}_u$ and lower-level matching $\mathcal{M}_\ell^\star \subset \mathcal{E}_\ell(\mathcal{M}_u^\star)$ that maximizes the global utility given by $J(\mathcal{M}_u, \mathcal{M}_\ell) = \sum_{e\in\mathcal{M}_u} c(e) + \sum_{e\in\mathcal{M}_\ell} d(e)$.

B. Optimization Model

The proposed mathematical programming model of the CLTA problem maximizes the global utility subject to coupling, one-to-one assignment, and binary constraints:

$$\text{maximize} \quad \sum_{i\in\mathcal{A}_u}\sum_{j\in\mathcal{T}_u} c_{ij}x_{ij} + \sum_{k\in\mathcal{A}_\ell}\sum_{l\in\mathcal{T}_\ell} d_{kl}y_{kl} \tag{2a}$$

$$\text{subject to} \quad y_{kl} \le x_{ij}, \qquad \forall k \in \mathcal{A}_\ell^i, \forall l \in \mathcal{T}_\ell^j, \forall i \in \mathcal{A}_u, \forall j \in \mathcal{T}_u, \tag{2b}$$

$$\sum_{i\in\mathcal{A}_u} x_{ij} \le 1, \qquad \forall j \in \mathcal{T}_u, \tag{2c}$$

$$\sum_{j\in\mathcal{T}_u} x_{ij} \le 1, \qquad \forall i \in \mathcal{A}_u, \tag{2d}$$

$$\sum_{k\in\mathcal{A}_\ell} y_{kl} \le 1, \qquad \forall l \in \mathcal{T}_\ell, \tag{2e}$$

$$\sum_{l\in\mathcal{T}_\ell} y_{kl} \le 1, \qquad \forall k \in \mathcal{A}_\ell, \tag{2f}$$

$$x_{ij} \in \{0,1\}, \qquad \forall i \in \mathcal{A}_u, \forall j \in \mathcal{T}_u, \tag{2g}$$

$$y_{kl} \in \{0,1\}, \qquad \forall k \in \mathcal{A}_\ell, \forall l \in \mathcal{T}_\ell. \tag{2h}$$

The binary decision variables $x_{ij} \in \{0,1\}, \forall i \in \mathcal{A}_u, j \in \mathcal{T}_u$ and $y_{kl} \in \{0,1\}, \forall k \in \mathcal{A}_\ell, l \in \mathcal{T}_\ell$ are introduced such that $x_{ij} = 1$ if task j is assigned to agent i at the upper level, and $y_{kl} = 1$ if task l is assigned to agent k at the lower level. Additionally, $c_{ij} \ge 0$ denotes the utility of assigning upper-level task j to upper-level agent i, and $d_{kl} \ge 0$ denotes the utility of assigning lower-level task l to lower-level agent k. Constraint (2b) ensures that no lower-level assignment is allowed unless its corresponding upper-level assignment is also selected. Constraints (2c) and (2d) enforce that each upper-level task is assigned to at most one agent, and each upper-level agent can take on at most one task. Similarly, constraints (2e) and (2f) enforce one-to-one assignments at the lower level. Finally, constraints (2g) and (2h) restrict all decision variables to binary values.

C. Illustrative Example

To illustrate the CLTA problem, consider $n_u = m_u = 2$ and $n_\ell = m_\ell = 4$. Let $\mathcal{A}_u = \{1,2\}$, $\mathcal{T}_u = \{1,2\}$, $\mathcal{A}_\ell = \{1,2,3,4\}$, and $\mathcal{T}_\ell = \{1,2,3,4\}$. The coupling sets in this example are defined as: $\mathcal{A}_\ell^1 = \{1,2\}$, $\mathcal{A}_\ell^2 = \{3,4\}$, $\mathcal{T}_\ell^1 = \{1,2\}$, and $\mathcal{T}_\ell^2 = \{3,4\}$. Let $\mathcal{G}_u = (\mathcal{A}_u \cup \mathcal{T}_u, \mathcal{A}_u \times \mathcal{T}_u)$ and let $\mathcal{G}_\ell = (\mathcal{A}_\ell \cup \mathcal{T}_\ell, \mathcal{A}_\ell \times \mathcal{T}_\ell)$ denote the (complete) bipartite graphs representing all feasible agent-task assignments at the upper and lower levels. Figure 2 provides a graphical overview of the graphs in both levels, where dashed rectangles group nodes into sets, with the corresponding set label displayed. In the lower-level graph, solid rectangles enclose nodes representing groups of agents or tasks coupled to a specific upper-level entity. Additionally, node colors are used to highlight the coupling between the two levels.

In Figure 2, edges connecting nodes indicate potential assignments between agents and tasks. Each edge has a corresponding weight that reflects the utility of the assignment. The objective is to select the optimal subset of these edges (*i.e.*, matching) to maximize the overall utility while satisfying the hierarchical relationships and assignment constraints. Note that due to the large number of edges at the lower level, the corresponding weights for these edges are omitted.

To model the assignment process, we introduce $\boldsymbol{x} \in \{0,1\}^{n_u m_u}$ and $\boldsymbol{y} \in \{0,1\}^{n_\ell m_\ell}$ to represent the upper-level and the lower-level decision vectors, respectively, where each binary variable represents whether an agent selects a task:

$$\boldsymbol{x} = \begin{bmatrix} x_{11} & x_{12} & x_{21} & x_{22} \end{bmatrix}^\top, \text{ and}$$

$$\boldsymbol{y} = \begin{bmatrix} y_{11} & y_{12} & y_{13} & y_{14} & y_{21} & y_{22} & y_{23} & y_{24} & y_{31} & y_{32} & y_{33} & y_{34} & y_{41} & y_{42} & y_{43} & y_{44} \end{bmatrix}^\top.$$

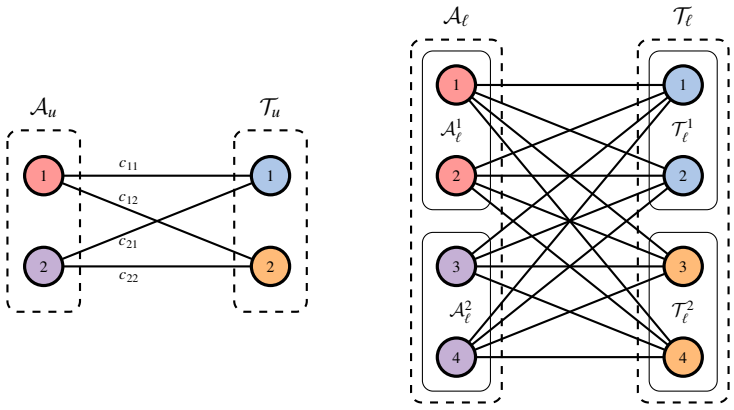


Figure 2 Upper-level and lower-level agent-task graphs

The coupling constraints in (2b) ensure that lower-level decisions remain consistent with the upper-level assignments. Table 1 includes all of the coupling constraints in order of lower-level tasks.

1	2	3	4
$y_{11} \leq x_{11}$	$y_{12} \leq x_{11}$	$y_{13} \leq x_{12}$	$y_{14} \leq x_{12}$
$y_{21} \leq x_{11}$	$y_{22} \leq x_{11}$	$y_{23} \leq x_{12}$	$y_{24} \leq x_{12}$
$y_{31} \leq x_{21}$	$y_{32} \leq x_{21}$	$y_{33} \leq x_{22}$	$y_{34} \leq x_{22}$
$y_{41} \leq x_{21}$	$y_{42} \leq x_{21}$	$y_{43} \leq x_{22}$	$y_{44} \leq x_{22}$

Table 1 Coupling constraints ordered by lower-level tasks

By concatenating all the binary decision variables, x_{ij} and y_{kl}, into a single vector, $\boldsymbol{z} = [\boldsymbol{x}^\top \ \boldsymbol{y}^\top]^\top$, the coupling constraints can be written in the compact form:

$$P\boldsymbol{z} \leq q,$$

where P is a matrix comprised of entries in $\{-1, 0, 1\}$ and q is a vector of zeros. Figure 3 is a visual representation of the matrix P, which represents the vectorized coupling constraints in Table 1 ordered by the lower-level tasks.

In this representation, the red squares denote entries of -1, the blue squares represent 1, and the white squares indicate 0. Note that in this example, all of the lower-level assignments are coupled to an upper-level assignment. To decouple a lower-level assignment from an upper-level assignment, one can remove its corresponding row.

IV. LP Relaxation and Integrality Gap

Boolean (or 0-1 integer) linear programs (BLPs) can represent a variety of combinatorial optimization problems that are known to be NP-hard. Examples include the traveling salesman problem [18, 19], for which solutions cannot be found in polynomial time unless P=NP. While there exist certain classes of these problems that can be solved efficiently, in particular, when the constraint matrix is totally unimodular [20, 21], or more generally when the problem is totally dual integral [22–24], these conditions are generally not satisfied.

Definition 1 (Totally Unimodular Matrix)**.** A matrix is totally unimodular (TUM) if every square submatrix has a determinant of $0, 1$, or -1.

TUM matrices are particularly significant because, with an integral right-hand side, they ensure integer optimal solutions to linear programs. Specifically, if a matrix P is TUM and a vector q is integral, then the binary constraints $\boldsymbol{z} \in \{0, 1\}^n$ can be relaxed to its convex hull, *i.e.*, $\boldsymbol{z} \in [0, 1]^n$, by appending bounds:

$$\begin{aligned} &\text{maximize} && w^\top \boldsymbol{z} \\ &\text{subject to} && \bar{P}\boldsymbol{z} \leq \bar{q}, \end{aligned} \tag{3}$$

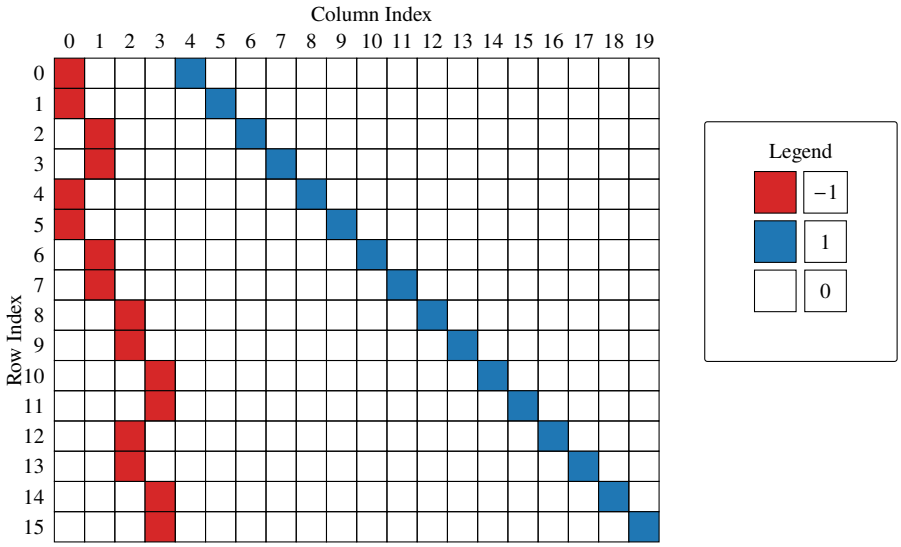


Figure 3 Coupling constraint matrix P

where

$$\bar{P} \coloneqq \begin{bmatrix} P \\ I \\ -I \end{bmatrix}, \quad \bar{q} \coloneqq \begin{bmatrix} q \\ \mathbf{1} \\ \mathbf{0} \end{bmatrix}.$$

The linear program in Equation (3) then has integral optima for any utility vector w. However, the CLTA problem given in (2) lacks the total unimodularity property.

Remark 1. A trivial CLTA instance is one where each upper-level agent/task has only one lower-level agent/task, *i.e.*, $|\mathcal{A}_\ell^i| = 1, \forall i \in \mathcal{A}_u$ and $|\mathcal{T}_\ell^j| = 1, \forall j \in \mathcal{T}_u$. In such a case, the lower-level can be augmented into the upper-level, and the problem can be solved as a standard linear task assignment problem.

Proposition 1. Any nontrivial CLTA instance that contains, for some fixed upper-level pair (i, j), at least two feasible lower-level choices on one side (*i.e.*, either a single lower-level agent with two candidate lower-level tasks or vice versa) has a constraint matrix that is not TUM.

Proof. Consider the simplest case with one upper-level agent, i, and one upper-level task, j. At the lower level, let there be one lower-level agent $k_1 \in \mathcal{A}_\ell^i$ and two lower-level tasks $l_1, l_2 \in \mathcal{T}_\ell^{j*}$. Let x denote the upper-level decision for (i, j), and $y_1 \coloneqq y_{k_1 l_1}$, $y_2 \coloneqq y_{k_1 l_2}$ the lower-level decisions. Ignoring box constraints, the relevant inequalities are:

$$y_1 \leq x, \quad y_2 \leq x, \quad y_1 + y_2 \leq 1, \tag{4}$$

which can be written as:

$$\underbrace{\begin{bmatrix} -1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}}_{A} \begin{bmatrix} x \\ y_1 \\ y_2 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{5}$$

The 3×3 submatrix A has determinant 2, hence it is not unimodular. Since every nontrivial coupled linear task assignment problem will contain the above submatrix, by definition, every nontrivial coupled linear task assignment problem will not have a totally unimodular constraint matrix. □

Remark 2. The TUM condition is sufficient but not necessary for integrality. Nevertheless, the LP relaxation of the CLTA problem can attain a fractional optimal solution. Consider the nontrivial instance from Proposition 1 with one

*The symmetric case with two lower-level agents and one lower-level task is analogous.

upper-level pair (i, j), a single lower-level agent $k_1 \in \mathcal{A}^i_\ell$, and two lower-level tasks $l_1, l_2 \in \mathcal{T}^j_\ell$. Relax the binary constraints to $x, y_1, y_2 \in [0, 1]$, and consider the following cost functions $c_{ij} = 0$ and $d_{k_1 l_1} = d_{k_1 l_2} = 1$. The feasible polyhedron is described by $y_1 \le x$, $y_2 \le x$, $y_1 + y_2 \le 1$, and $0 \le x, y_1, y_2 \le 1$, whose vertices are:

$$(0,0,0),\ (1,0,0),\ (1,1,0),\ (1,0,1),\ \left(\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2}\right). \tag{6}$$

However, evaluating the linear objective $c_{ij}x + d_{k_1 l_1} y_1 + d_{k_1 l_2} y_2$ at these vertices yields the values $\{0, 0, 1, 1, 1\}$, respectively, so the fractional vertex $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ is optimal. Consequently, CLTA problems cannot be solved efficiently using standard linear programming (LP) relaxation techniques while guaranteeing integral optima, as is done for classical linear task assignment problems. The polyhedron can be plotted on a 3D plot, as Figure 4 illustrates the three half-spaces $y_1 \le x$ (red), $y_2 \le x$ (orange), and $y_1 + y_2 \le 1$ (blue); vertices are shown as black circles.

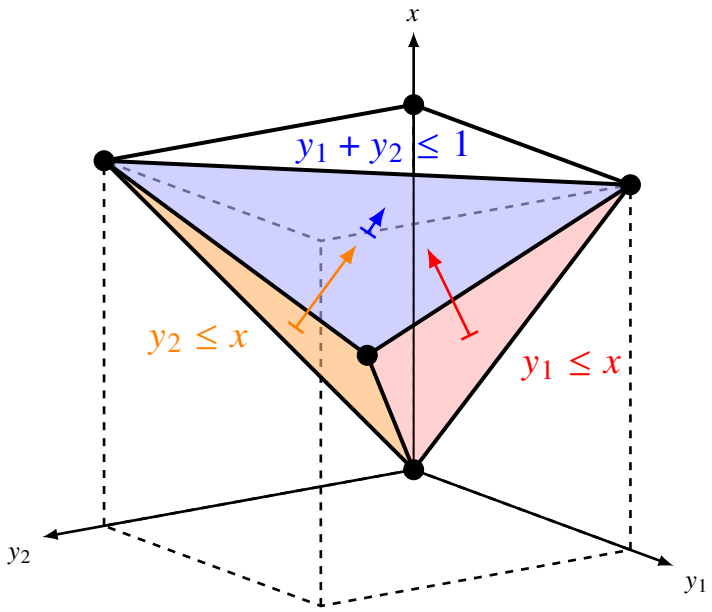


Figure 4 Polyhedron defined by the simplest nontrivial CLTA problem

V. Nested Hungarian Method

The presence of a fractional vertex in the relaxed polytope (Remark 2) implies that standard LP relaxations cannot generally be relied upon to recover integral CLTA solutions. We therefore pursue a combinatorial method based on the Hungarian algorithm. The idea is to solve, for every upper-level pair (i, j), the associated lower-level assignment to optimality and augment its contribution into a combined utility $\hat{c}_{ij}$. We can then solve a single upper-level assignment with these combined utilities and, finally, extract the corresponding lower-level matches. This yields a simple nested Hungarian procedure with polynomial runtime.

We first impose the following assumption to allow for the use of the Hungarian algorithm as a subroutine to solve auxiliary assignment problems in our proposed algorithm. Note that this assumption is not restrictive and can always hold, as any imbalance can be addressed by introducing dummy agents or tasks with zero utility.

Assumption 1 (Balanced Assignment)**.** The number of agents and tasks in the upper level is equal, *i.e.* $n_u = m_u$. Additionally, for all $i \in \mathcal{A}_u$ and $j \in \mathcal{T}_u$, the numbers of agents and tasks in the lower level are equal, *i.e.* $|\mathcal{A}^i_\ell| = |\mathcal{T}^j_\ell|$.

The proposed solution method for the CLTA problem is summarized in Algorithm 1. The algorithm, which is a variant of the Hungarian algorithm for single-level assignment problems, works as follows. In line 3, for all upper-level agent-task pairs (i, j), we find the corresponding optimal lower-level assignment vector $\boldsymbol{y}^{i\to j}$ by solving the following

Algorithm 1: Nested Hungarian algorithm for the CLTA problem

Input : Index sets $\mathcal{A}_u, \mathcal{T}_u, \mathcal{A}_\ell, \mathcal{T}_\ell$; coupling maps $\{\mathcal{A}_\ell^i\}_{i\in\mathcal{A}_u}$, $\{\mathcal{T}_\ell^j\}_{j\in\mathcal{T}_u}$; utilities c_{ij}, d_{kl}.
Output : Upper assignments $\boldsymbol{x}^\star \in \{0,1\}^{n_u m_u}$; lower assignments $\boldsymbol{y}^\star \in \{0,1\}^{n_\ell m_\ell}$.

Solve every lower-level assignment and construct the combined upper-level utility:
1 **for** $i \in \mathcal{A}_u$ **do**
2 **for** $j \in \mathcal{T}_u$ **do**
3 $\boldsymbol{y}^{i\to j} \leftarrow$ Solve (7) using HUNGARIAN (*e.g.*, [25]) `// Lower-level assignment for fixed (i, j)`
4 $\hat{c}_{ij} \leftarrow c_{ij} + \sum_{k\in\mathcal{A}_\ell^i} \sum_{l\in\mathcal{T}_\ell^j} d_{kl}\, y_{kl}^{i\to j}$ `// Combined upper-level utility`

Solve combined upper-level assignments:
5 $\boldsymbol{x}^\star \leftarrow$ Solve (8) using HUNGARIAN (*e.g.*, [25]) `// Upper-level assignment`

Extract lower-level assignments:
6 **for** $i \in \mathcal{A}_u$ **do**
7 **for** $j \in \mathcal{T}_u$ **do**
8 **if** $x_{ij}^\star = 1$ **then**
9 **for** $k \in \mathcal{A}_\ell^i$ **do**
10 **for** $l \in \mathcal{T}_\ell^j$ **do**
11 $y_{kl}^\star \leftarrow y_{kl}^{i\to j}$ `// Select the stored lower-level assignment`
12 **return** $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$

lower-level assignment problem (for fixed i and j) using the Hungarian algorithm:

$$\text{maximize} \quad \sum_{k\in\mathcal{A}_\ell^i} \sum_{l\in\mathcal{T}_\ell^j} d_{kl} y_{kl} \tag{7a}$$

$$\text{subject to} \quad \sum_{k\in\mathcal{A}_\ell^i} y_{kl} \le 1, \qquad \forall l \in \mathcal{T}_\ell^j, \tag{7b}$$

$$\sum_{l\in\mathcal{T}_\ell^j} y_{kl} \le 1, \qquad \forall k \in \mathcal{A}_\ell^i, \tag{7c}$$

$$y_{kl} \in \{0,1\}, \qquad \forall k \in \mathcal{A}_\ell^i, \forall l \in \mathcal{T}_\ell^j. \tag{7d}$$

Subsequently, we update the (i,j)th combined upper-level utility $\hat{c}_{ij}$ as in line 4. Once this procedure is complete for all i and j, we solve in line 5 the following upper-level assignment problem with combined utilities $\hat{c}_{ij}$:

$$\text{maximize} \quad \sum_{i\in\mathcal{A}_u} \sum_{j\in\mathcal{T}_u} \hat{c}_{ij} x_{ij} \tag{8a}$$

$$\text{subject to} \quad \sum_{i\in\mathcal{A}_u} x_{ij} \le 1, \qquad \forall j \in \mathcal{T}_u, \tag{8b}$$

$$\sum_{j\in\mathcal{T}_u} x_{ij} \le 1, \qquad \forall i \in \mathcal{A}_u, \tag{8c}$$

$$x_{ij} \in \{0,1\}, \qquad \forall i \in \mathcal{A}_u, \forall j \in \mathcal{T}_u, \tag{8d}$$

which gives us the optimal upper-level assignment vector $\boldsymbol{x}^\star$ for the CLTA problem. We then extract the optimal lower-level assignment variables corresponding to each optimal upper-level agent-task pair from the previously computed lower-level assignments.

Algorithm 1 runs in polynomial time and is guaranteed to achieve the optimal assignment for the CLTA problem, as verified in the following proposition.

Proposition 2. For the CLTA problem in (2), under Assumption 1, Algorithm 1 runs in time $O\left((n_u)^2 (n_\ell)^3\right)$ and attains the optimal solution.

Proof. Time complexity: For each upper-level agent $i \in \mathcal{A}_u$ and each upper-level task $j \in \mathcal{T}_u$, the Hungarian algorithm is used in line 3 to compute the optimal lower-level assignment $\boldsymbol{y}^{i\to j}$ by solving a problem of size $n_\ell \times n_\ell$ (under Assumption 1). At each stage, the Hungarian algorithm takes $O((n_\ell)^2)$ arithmetic operations (if implemented with the appropriate data structures [26]). The computational complexity of the Hungarian algorithm involving n_ℓ stages is thus $O((n_\ell)^3)$ [25]. Since there are $(n_u)^2$ invocations of the Hungarian algorithm, the total complexity for computing $\boldsymbol{y}^{i\to j}$ is $O((n_u)^2(n_\ell)^3)$. The final step of applying the Hungarian algorithm in line 5 has a complexity of $O((n_u)^3)$, which is dominated by what precedes in the nested loops. Thus, the overall time complexity is $O((n_u)^2(n_\ell)^3)$.

Optimality: By construction, the algorithm computes the optimal lower-level assignments $\boldsymbol{y}^{i\to j}$ for each pair (i, j) using the Hungarian algorithm, ensuring that the lower-level assignments are optimal for the given upper-level assignment. Thus, the combined utility $\hat{c}_{ij}$ accurately reflects the total utility of assigning upper-level agent i to upper-level task j. The Hungarian algorithm is then applied to compute the optimal upper-level assignment $\boldsymbol{x}^\star$. Since the Hungarian algorithm guarantees an optimal assignment, $\boldsymbol{x}^\star$ is optimal for the combined utility. Finally, the optimal lower-level assignments $\boldsymbol{y}^\star$ are extracted by selecting the previously computed lower-level assignments $\boldsymbol{y}^{i\to j}$ corresponding to the optimal upper-level assignments $\boldsymbol{x}^\star$. □

VI. Numerical Simulations

A. Wildfire Response Simulations

For numerical simulations, we consider a CLTA problem motivated by the January 2025 Southern California wildfires in the Greater Los Angeles area. At the upper level, five airtankers (ULAs) are based at Los Angeles International (LAX), John Wayne/Santa Ana (SNA), Hollywood–Burbank (BUR), Ontario (ONT), and Long Beach (LGB) airports. These five ULAs must each be assigned to one of five simultaneous wildfires (Palisades, Eaton, Hurst, Sunset, and Kenneth), resulting in a 5×5 upper-level utility matrix. We consider two simulation scenarios. In the balanced case, every airtanker has eight drones, and each wildfire has eight subtasks, resulting in a total of 40 lower-level agents and 40 lower-level tasks. Formally, $|\mathcal{A}_\ell^i| = 8$ for all i and $|\mathcal{T}_\ell^j| = 8$ for all j, so $|\mathcal{A}_\ell| = |\mathcal{T}_\ell| = 40$. In the unbalanced case, the number of drones and subtasks associated with each pair varies. To preserve the one-to-one assignment structure, we add dummy agents with zero utility until each aircraft has eight lower-level agents and each fire has exactly eight lower-level tasks.

To capture the operational trade-offs inherent in wildfire emergency response, we develop a hierarchical utility function framework comprising two levels. At each level, the utility functions are formulated as a function of problem-specific factors. We note that the factors included in this framework are not exhaustive; additional components can be readily integrated to accommodate further complexities found in real-world applications.

Weighted Suitability Design

We define a single weighted suitability function and use it at both levels. It measures how well a capability profile aligns with a demand profile by computing per-attribute agreement ratios in $[0, 1]$ (perfect agreement yields 1) and taking a convex combination with normalized attribute weights.

Let $\mathcal{K}_0$ be the finite index set of all attribute types considered (*e.g.*, capacity, endurance, sensor quality). Let $(a_k)_{k\in\mathcal{K}_0}$ denote nonnegative capability attributes (aircraft or drone), $(b_k)_{k\in\mathcal{K}_0}$ the corresponding nonnegative demand attributes (wildfire or task), and $(w_k)_{k\in\mathcal{K}_0}$ nonnegative attribute weights. We compare only attributes present in all three vectors; formally,

$$\mathcal{K} \coloneqq \{k \in \mathcal{K}_0 \mid a_k \text{ is defined},\ b_k \text{ is defined},\ w_k \text{ is defined}\} = \operatorname{supp}(a) \cap \operatorname{supp}(b) \cap \operatorname{supp}(w), \tag{9}$$

where for a vector v, $\operatorname{supp}(v)$ denotes the set of indices for which v is defined. Define normalized weights $\tilde{w}_k \coloneqq w_k \big/ \sum_{h\in\mathcal{K}} w_h$ (if $\sum_{h\in\mathcal{K}} w_h = 0$, set the suitability to 0 by convention). For each $k \in \mathcal{K}$, set

$$r_k \coloneqq \begin{cases} 1, & \max\{a_k, b_k\} = 0, \\ \dfrac{\min\{a_k, b_k\}}{\max\{a_k, b_k\}}, & \text{otherwise}, \end{cases} \qquad \mathsf{Suit}(a, b, w) \coloneqq \sum_{k\in\mathcal{K}} \tilde{w}_k\, r_k \ \in [0, 1]. \tag{10}$$

For airtankers, Suit captures compatibility between tanker capabilities (*e.g.*, retardant capacity, maneuverability) and wildfire demands (*e.g.*, drop volume, terrain constraints). For drones, the same construction captures how drone metrics (*e.g.*, payload, endurance, sensor quality) match task demands. The ratio form gives full credit when both sides are zero for an attribute and smoothly penalizes mismatches otherwise.

Airtanker–Wildfire Utility Design

At the upper level, the pairing of airtankers to wildfires also includes a response-efficiency component that favors short arrival times. Let $p_i \in \mathbb{R}^2$ be the location of base i, $q_j \in \mathbb{R}^2$ the location of fire j, and $v_i > 0$ the maximum speed of airtanker i. Define

$$t_{ij} \coloneqq \frac{\|p_i - q_j\|_2}{v_i}, \qquad \mathsf{RespEff}_{ij} \coloneqq \frac{1}{1 + t_{ij}} \in (0, 1]. \tag{11}$$

In operational settings, t_{ij} may incorporate factors more realistic than the Euclidean-speed metric, such as winds, routing, and airspace restrictions.

Drone–Task Utility Design

We model two effects at the lower level: a hard feasibility check (specialization) and a graded fit via the same suitability function (10). Each task l specifies (i) a set of required categorical capabilities R_l^{cat} (*e.g.*, `suppression`, `lighting`, `mapping`) and (ii) quantitative minimum requirements indexed by $G_l^{\min}$ with thresholds $\{\theta_{l,g}\}_{g \in G_l^{\min}}$ (*e.g.*, payload, endurance, sensor resolution). Each drone k provides (i) a set of available categorical capabilities S_k^{cat} and (ii) quantitative metrics $\{\psi_{k,g}\}$. Drone k satisfies the specialization of task l if *all* requirements are met:

$$R_l^{\text{cat}} \subseteq S_k^{\text{cat}} \quad \text{and} \quad \psi_{k,g} \geq \theta_{l,g} \;\; \forall\, g \in G_l^{\min}. \tag{12}$$

The specialization indicator is:

$$\mathsf{SpecMatch}_{kl} \coloneqq \begin{cases} 1, & R_l^{\text{cat}} \subseteq S_k^{\text{cat}} \text{ and } \psi_{k,g} \geq \theta_{l,g} \;\; \forall g \in G_l^{\min}, \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

Thus, $\mathsf{SpecMatch}_{kl} = 1$ enforces a hard feasibility filter: a drone is eligible only if it has every required tag and meets every minimum threshold.

Upper and Lower Utilities

With a constant $\lambda \in [0, 1]$, the upper-level utility combines response efficiency and the unified suitability:

$$c_{ij} \coloneqq \lambda\, \mathsf{RespEff}_{ij} + (1 - \lambda)\, \mathsf{Suit}(\phi_i, \omega_j, w), \tag{14}$$

where ϕ_i are aircraft capabilities, ω_j are wildfire demands, and w are attribute weights.

The lower-level utility of assigning drone k to task l is given by:

$$d_{kl} \coloneqq \mathsf{SpecMatch}_{kl} \times \mathsf{Suit}(\psi_k, \theta_l, w_l), \tag{15}$$

where ψ_k are drone metrics, θ_l are task demands, and w_l are task-specific weights.

Assignment Results

Figure 5 depicts the balanced case. Each airtanker is shown as a '⋆' symbol, and every wildfire as an 'x' symbol enclosed by a dashed circle. The two elements in a pair of assignments share the same fill color. Around every star, there are eight circular markers, each representing a drone, while each dashed circle is ringed by eight '+' symbols that represent the associated lower-level tasks. Within every aircraft-fire assignment, there are eight drone-task pairings. The algorithm dispatches the Hollywood–Burbank (BUR) airtanker to the Eaton (EAT) fire, John Wayne/Santa Ana (SNA) to Kenneth (KEN), Long Beach (LGB) to Palisades (PAL), Ontario (ONT) to Hurst (HUR), and Los Angeles International (LAX) to Sunset (SUN). In the accompanying bipartite diagrams of Figure 7, each aircraft–wildfire sector contains eight drones and eight subtasks, and the solution forms the required one-to-one matching without any dummy vertices.

Figure 6 shows the unbalanced case. To balance the assignments, we introduce dummy agents/tasks, which are depicted with a white fill color. Any assignment that involves at least one dummy agent/task will have a black border color. The upper-level assignments remain the same, except that Ontario (ONT) is now assigned to Kenneth (KEN), and John Wayne/Santa Ana (SNA) is assigned to Hurst (HUR). The accompanying bipartite diagrams, which visualize the assignments, are provided in Figure 8.

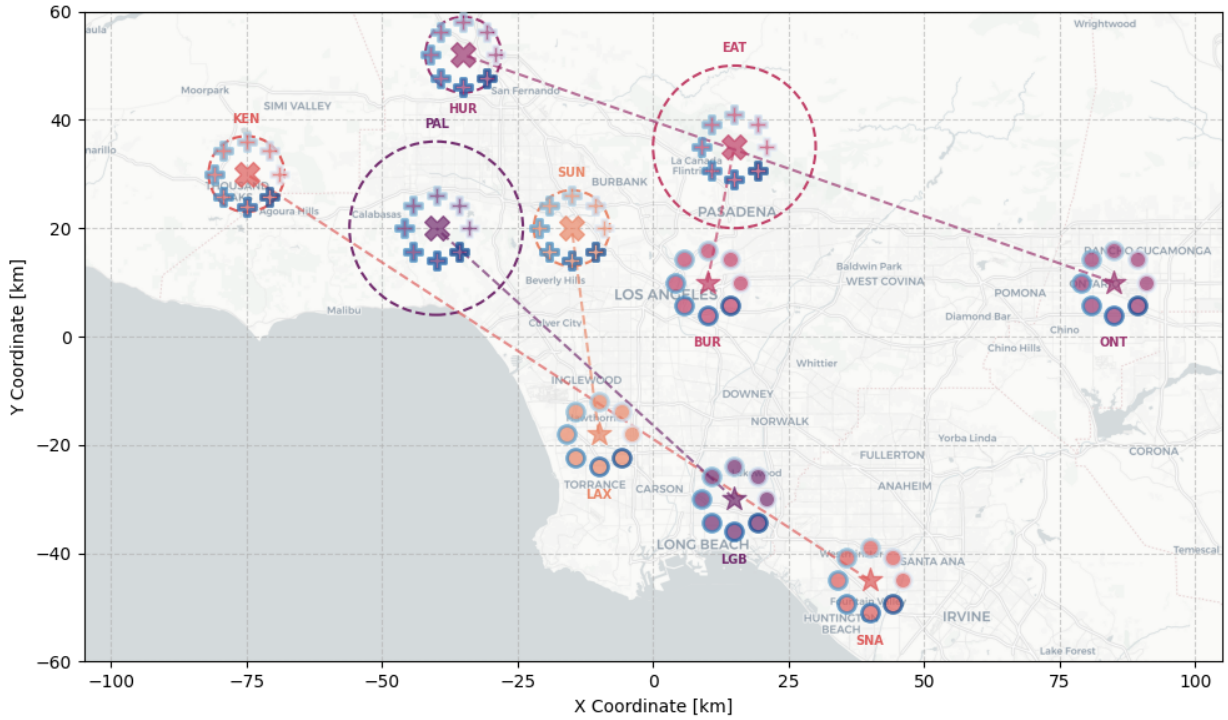


Figure 5 Hierarchical task assignment for Los Angeles fires with balanced structure

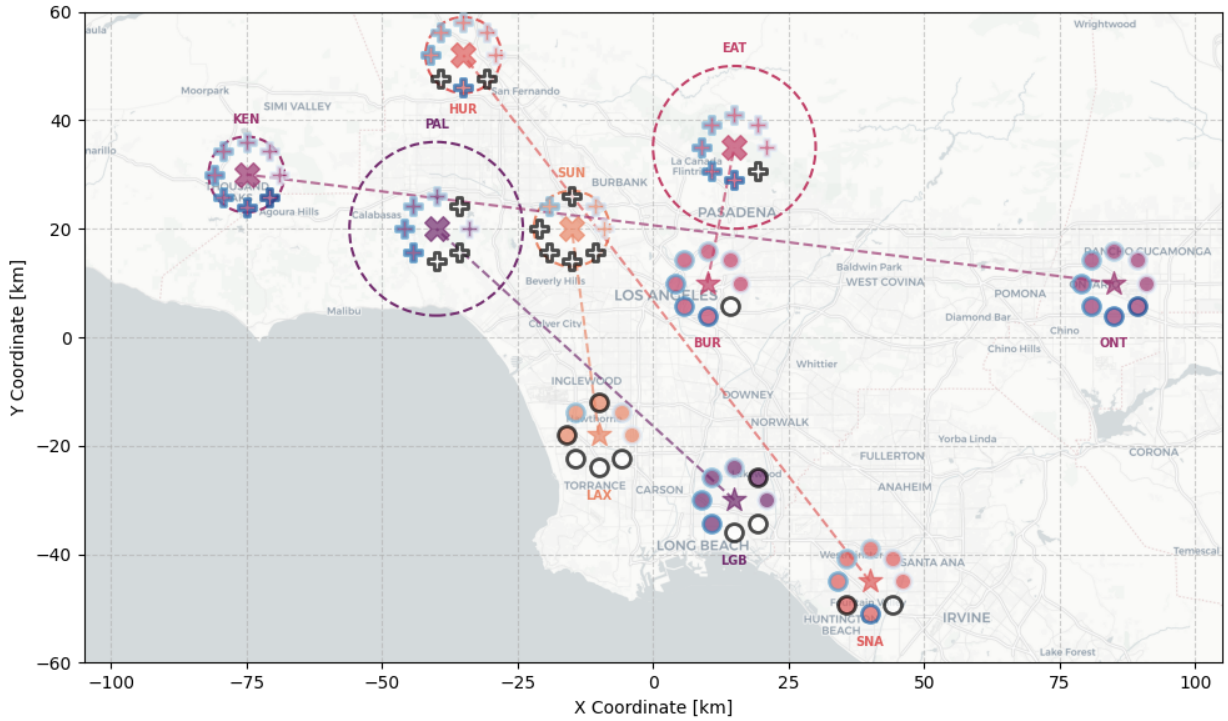


Figure 6 Hierarchical task assignment for Los Angeles fires with dummy variables

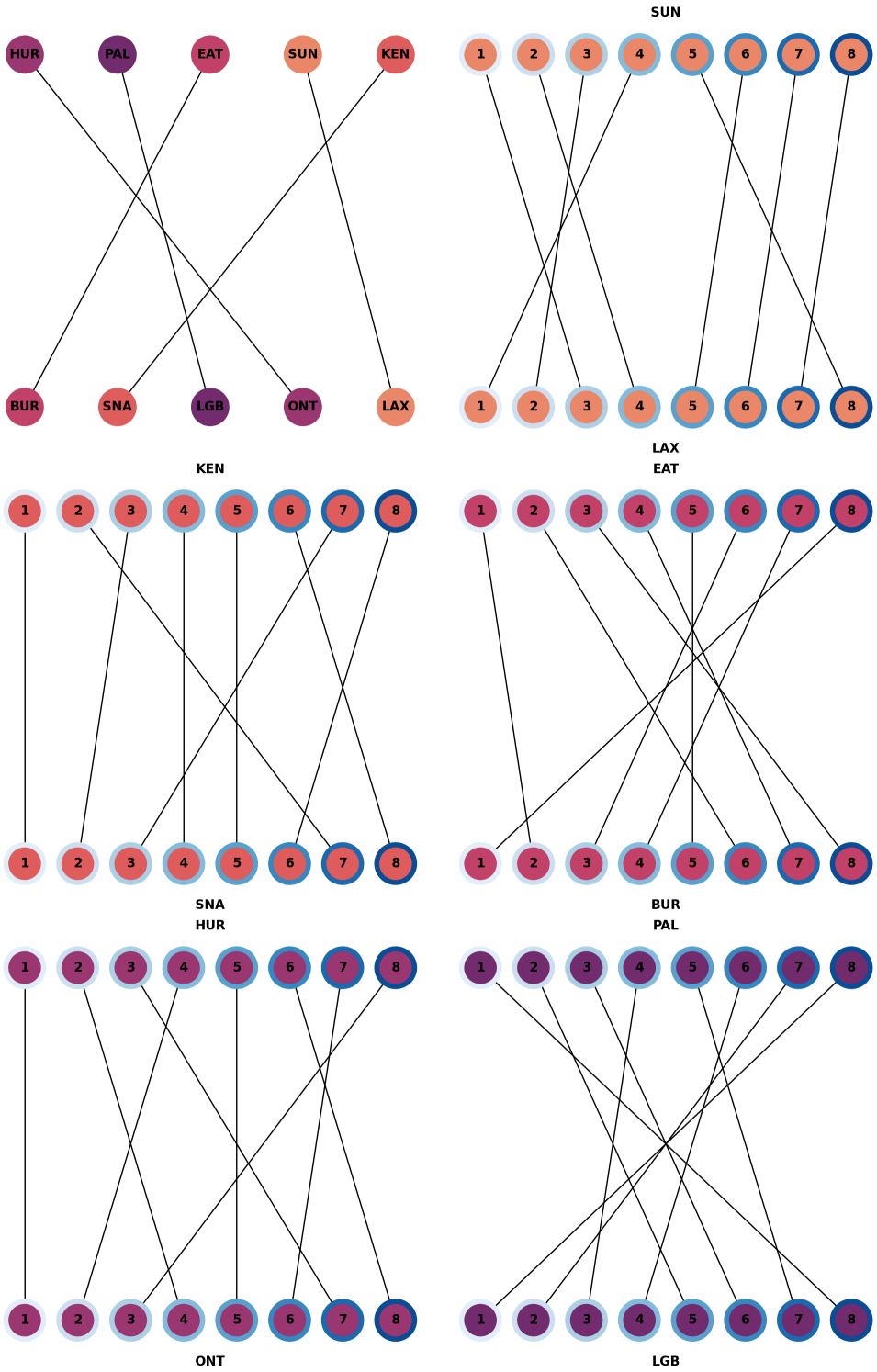


Figure 7 Upper-level and lower-level assignments with balanced structure

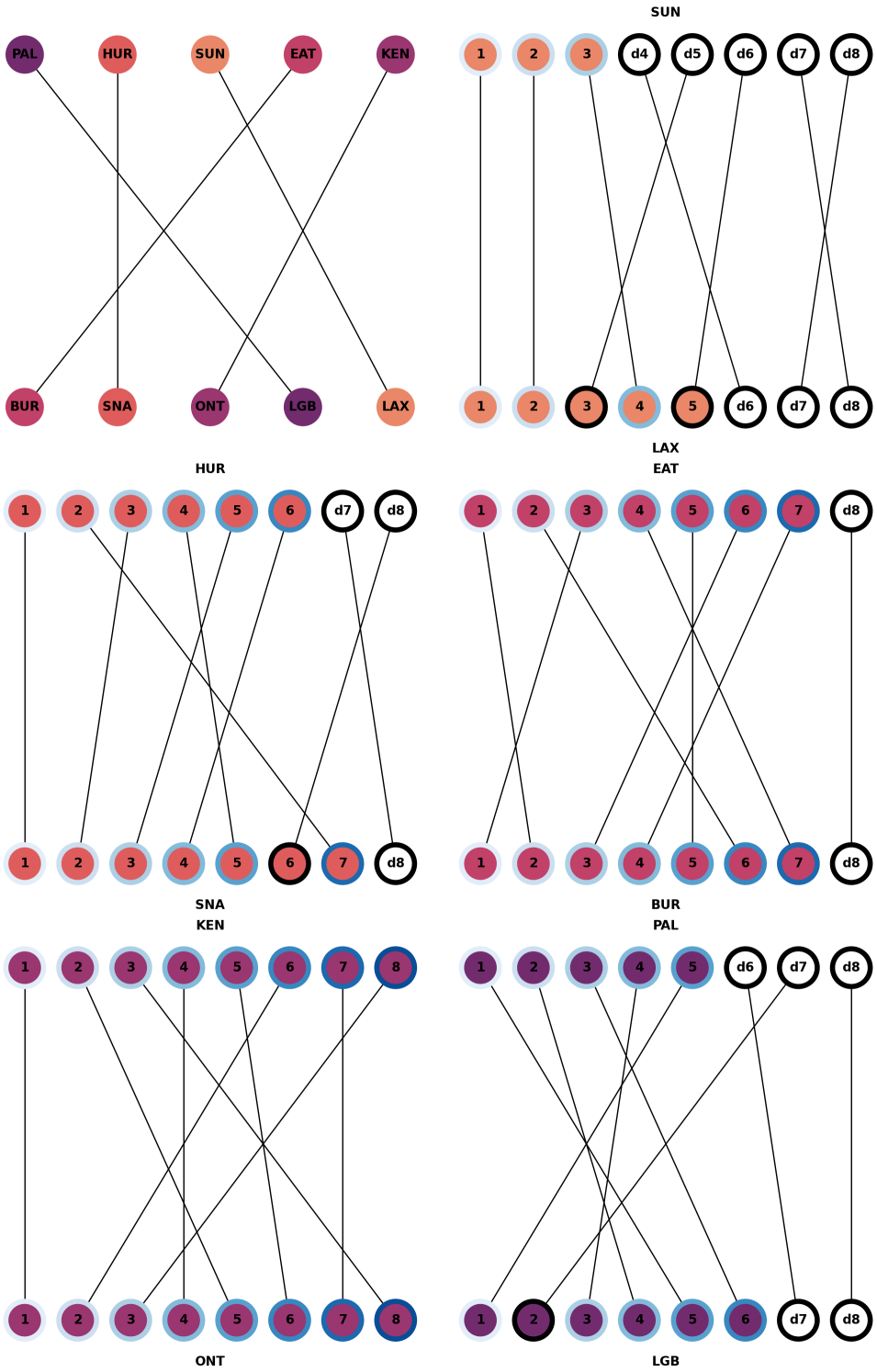


Figure 8 Upper-level and lower-level assignments with dummy variables

B. Benchmarks

We benchmark our proposed solution against various off-the-shelf MILP solvers, specifically CPLEX [27], MOSEK [28], PuLP (CBC backend) [29, 30], COIN-OR CLP [31], SCIP [32], HiGHS [33], and Gurobi [34]. All benchmarking experiments were conducted on a desktop computer equipped with an Intel(R) Core(TM) i9-10900K CPU at 3.70 GHz. We consider three different problem sizes: small, medium, and large. For each category, we uniformly sample the number of upper-level agents (n_u), the number of upper-level tasks (m_u), the number of lower-level agents (n_ℓ), and the number of lower-level tasks (m_ℓ). The parameter ranges are defined as follows:

- Small: $n_u = m_u \in [1, 5]$ and $n_\ell = m_\ell \in [1, 25]$.
- Medium: $n_u = m_u \in [5, 10]$ and $n_\ell = m_\ell \in [25, 100]$.
- Large: $n_u = m_u \in [11, 20]$ and $n_\ell = m_\ell \in [121, 400]$.

Each problem instance is randomly generated by uniformly sampling values within the specified ranges. We evaluate solver performance based on runtime, comparing both the relative and absolute performance of each solver. We use performance profiles [35] to evaluate the efficiency of the solver across a set of N benchmark problems. For each problem p and solver s, let $t_{p,s}$ denote the solve time. The relative performance ratio is defined as:

$$u_{p,s} = \frac{t_{p,s}}{\min_{s'} t_{p,s'}},$$

which compares solver s to the best solver for problem p. The relative performance profile is the function:

$$f_s^r(\tau) = \frac{1}{N} \sum_{p=1}^{N} \mathbb{I}(u_{p,s} \leq \tau),$$

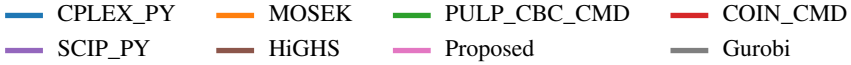
where $\mathbb{I}$ is the indicator function. The performance profile function gives the fraction of problems solved by s within a factor τ of the solve time of the best solver. To account for absolute speed (solve time), we also define the absolute performance profile:

$$f_s^a(\tau) = \frac{1}{N} \sum_{p=1}^{N} \mathbb{I}(t_{p,s} \leq \tau),$$

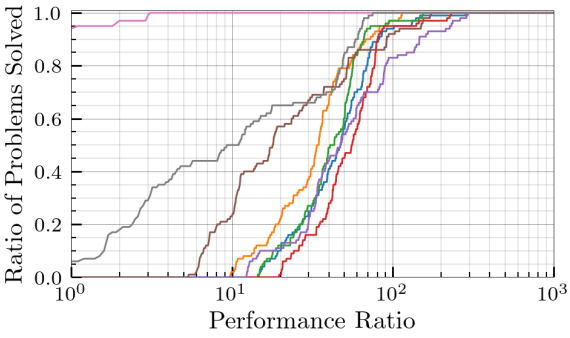
which measures the fraction of problems solved by s within τ seconds, independent of other solvers.

We present performance profiles for each problem size, small, medium, and large, as shown in Figures 9, 10, and 11, respectively. Figure 9a shows that, on the small test set, the proposed algorithm solves every instance at the lowest observed runtime, reaching a relative-profile value of one at a performance ratio of $\tau = 3$. The next-best MILP solver, Gurobi, achieves a relative-profile value of one at a performance ratio of $\tau = 70$, while the other solvers lag further behind. Figure 9b confirms the result as it shows that the proposed method clears the entire batch in well under 10^{-3} seconds, while all the other solvers take at least an order of magnitude longer. Figure 10a shows that, on the medium test set, the proposed algorithm solves every instance at the lowest observed runtime, attaining a relative-profile value of one at a performance ratio of $\tau = 1$. The next-best MILP solvers, Gurobi and CPLEX, don't achieve full coverage until $\tau \approx 20$. Figure 10b corroborates this result as the proposed method clears the entire medium batch in less than 10^{-2} seconds, whereas Gurobi and CPLEX require at least 10^{-1} seconds. Figure 11a presents the large benchmark results. Once again, the proposed algorithm has a relative profile value of one at $\tau = 1$, indicating that it achieves the best performance on every large problem instance. The next best solver, Gurobi, reaches full coverage only after $\tau \approx 100$. The absolute profile in 11b shows that the proposed algorithm solves all of the large problems in around 10^{-1} seconds, while other solvers take from 2 seconds to more than a minute.

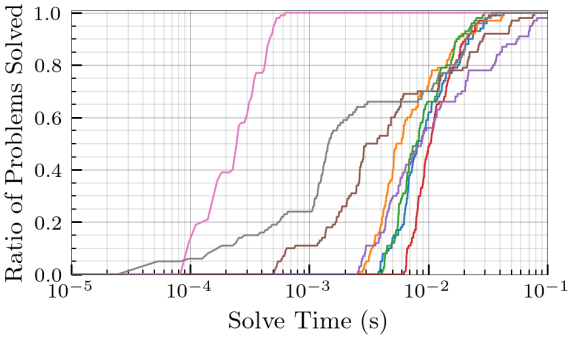
Figure 12 plots solve time against problem size, where size is defined as the total number of decision variables plus the total number of constraints. For very small problem instances, those with fewer than 10^2 combined variables and constraints, the proposed method and Gurobi exhibit very similar runtimes, both solving the problem in less than a millisecond. Once the problem size exceeds 10^2, the proposed algorithm's curve sits uniformly below Gurobi's, and the performance gap widens steadily with increasing scale.



Solver legend

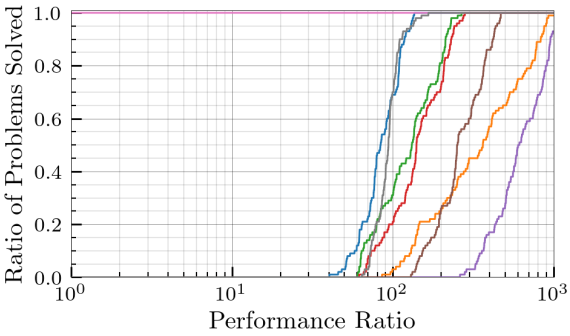


(a) Relative performance profile

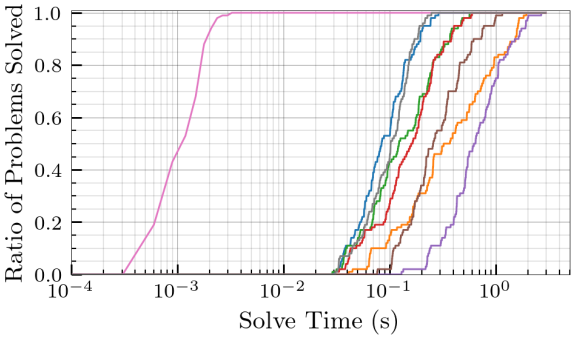


(b) Absolute performance profile

Figure 9 Performance profiles for the small problem set

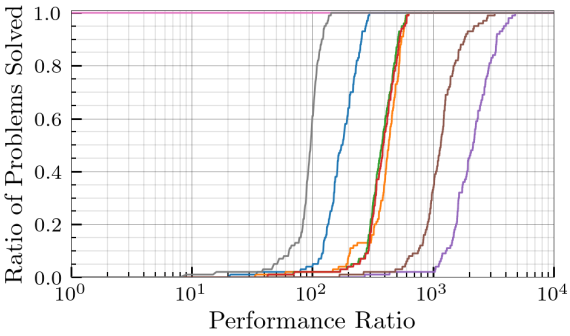


(a) Relative performance profile

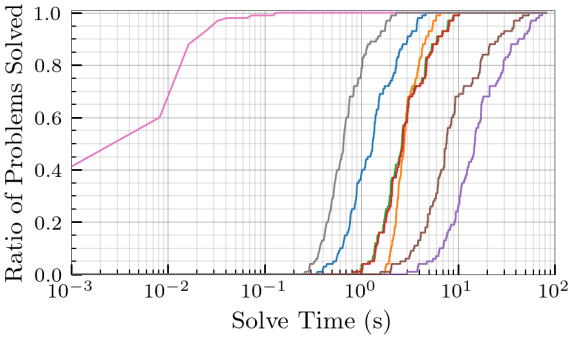


(b) Absolute performance profile

Figure 10 Performance profiles for the medium problem set



(a) Relative performance profile



(b) Absolute performance profile

Figure 11 Performance profiles for the large problem set

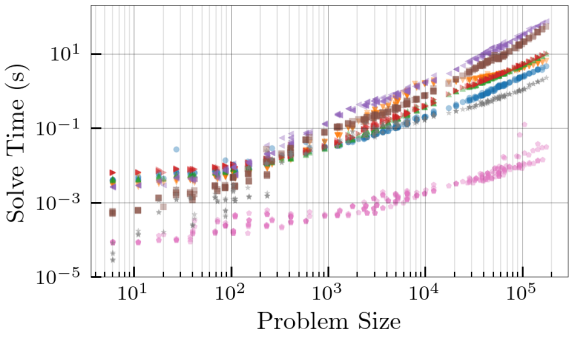


Figure 12 Solve time of each solver for different problem sizes

VII. Conclusion

In this work, we introduced the Coupled Linear Task Assignment (CLTA) problem to model hierarchical aircraft-drone coordination for disaster response. The formulation explicitly links two assignment layers through feasibility couplings, capturing the conditions under which lower-level (drone-task) decisions are permissible given upper-level (aircraft-wildfire) choices. We demonstrated that standard LP relaxations need not be integral, which clarifies why classical linear-assignment guarantees do not apply. The proposed algorithm, Nested Hungarian, exploits the CLTA's structure to obtain the optimal solution in polynomial time. We show that this procedure attains exact optimality with significantly lower computational cost than generic MILP solvers.

Empirically, synthetic simulations motivated by the January 2025 Southern California wildfire response in the Greater Los Angeles area illustrated how mission utilities can combine response efficiency and capability with demand suitability at both levels. Benchmarks across small, medium, and large instances showed that the specialized Nested Hungarian method consistently delivered the fastest runtimes (especially at larger scales) while matching MILP objective values in its domain of applicability.

A key next step is to decentralize CLTA by pushing computation to aircraft and drone teams and coordinating via lightweight consensus or auction protocols, so assignment updates remain feasible and near-optimal under intermittent, delayed, or lossy links to any central coordinator. We will analyze robustness guarantees, including feasibility preservation and bounded suboptimality, under packet loss, stale information, and asynchronous communication.

Acknowledgments

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-22-2-0091. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies; either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

[1] Bharosa, N., Lee, J., and Janssen, M., ''Challenges and obstacles in sharing and coordinating information during multi-agency disaster response: Propositions from field exercises,'' *Information Systems Frontiers*, Vol. 12, No. 1, 2010, pp. 49–65. https://doi.org/10.1007/s10796-009-9174-z, URL https://doi.org/10.1007/s10796-009-9174-z.

[2] Queralta, J. P., Taipalmaa, J., Can Pullinen, B., Sarker, V. K., Nguyen Gia, T., Tenhunen, H., Gabbouj, M., Raitoharju, J., and Westerlund, T., ''Collaborative Multi-Robot Search and Rescue: Planning, Coordination, Perception, and Active Vision,'' *IEEE Access*, Vol. 8, 2020, pp. 191617–191643. https://doi.org/10.1109/ACCESS.2020.3030190.

[3] Lyu, M., Zhao, Y., Huang, C., and Huang, H., ''Unmanned Aerial Vehicles for Search and Rescue: A Survey,'' *Remote Sensing*, Vol. 15, No. 13, 2023. https://doi.org/10.3390/rs15133266, URL https://www.mdpi.com/2072-4292/15/13/3266.

[4] Qi, J., Song, D., Shang, H., Wang, N., Hua, C., Wu, C., Qi, X., and Han, J., ''Search and Rescue Rotary-Wing UAV and Its Application to the Lushan Ms 7.0 Earthquake,'' *Journal of Field Robotics*, Vol. 33, No. 3, 2016, pp. 290–321. https://doi.org/https://doi.org/10.1002/rob.21615, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21615.

[5] Faiz, T. I., Vogiatzis, C., and Noor-E-Alam, M., ''Computational Approaches for Solving Two-Echelon Vehicle and UAV Routing Problems for Post-Disaster Humanitarian Operations,'' *Expert Systems with Applications*, Vol. 237, 2024, p. 121473. https://doi.org/https://doi.org/10.1016/j.eswa.2023.121473, URL https://www.sciencedirect.com/science/article/pii/S0957417423019759.

[6] Atay, N., and Bayazit, B., ''Mixed-Integer Linear Programming Solution to Multi-Robot Task Allocation Problem,'' 2006.

[7] Schumacher, C., Chandler, P., Pachter, M., and Pachter, L., ''UAV Task Assignment with Timing Constraints via Mixed-Integer Linear Programming,'' *AIAA 3rd" Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, 2004, p. 6410.

[8] Darrah, M., Niland, W., and Stolarik, B., ''Multiple UAV Dynamic Task Allocation Using Mixed Integer Linear Programming in a SEAD Mission,'' *Infotech@Aerospace*, 2005. https://doi.org/10.2514/6.2005-7164, URL https://arc.aiaa.org/doi/abs/10.2514/6.2005-7164.

[9] Weintraub, I. E., Von Moll, A., Casbeer, D. W., and Manyam, S. G., ''Virtual Target Selection for a Multiple-Pursuer–Multiple-Evader Scenario,'' *Journal of Aerospace Information Systems*, Vol. 0, No. 0, 2025, pp. 1–10. https://doi.org/10.2514/1.I011510, URL https://doi.org/10.2514/1.I011510.

[10] Afonso, R. J., Maximo, M. R., and Galvão, R. K., ''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, 2020, pp. 5464–5491.

[11] Choi, H.-L., Brunet, L., and How, J. P., ''Consensus-Based Decentralized Auctions for Robust Task Allocation,'' *IEEE Transactions on Robotics*, Vol. 25, No. 4, 2009, pp. 912–926. https://doi.org/10.1109/TRO.2009.2022423.

[12] Oh, G., Kim, Y., Ahn, J., and Choi, H.-L., ''Market-Based Distributed Task Assignment of Multiple Unmanned Aerial Vehicles for Cooperative Timing Mission,'' *Journal of Aircraft*, Vol. 54, No. 6, 2017, pp. 2298–2310. https://doi.org/10.2514/1.C032984, URL https://doi.org/10.2514/1.C032984.

[13] Samiei, A., and Sun, L., ''Distributed Matching-By-Clone Hungarian-Based Algorithm for Task Allocation of Multiagent Systems,'' *IEEE Transactions on Robotics*, Vol. 40, 2024, pp. 851–863. https://doi.org/10.1109/TRO.2023.3335656.

[14] Bakolas, E., and Lee, Y., ''Decentralized Game-Theoretic Control for Dynamic Task Allocation Problems for Multi-Agent Systems,'' *2021 American Control Conference (ACC)*, 2021, pp. 3228–3233. https://doi.org/10.23919/ACC50511.2021.9483030.

[15] Lee, Y., Khalil, A., Bakolas, E., and Gremillion, G., ''A Two-Phase Algorithm for Joint Task Assignment and Coordination in Hierarchical Multi-Vehicle Systems,'' AIAA SCITECH 2025 Forum, 2025. https://doi.org/10.2514/6.2025-2283, URL https://arc.aiaa.org/doi/abs/10.2514/6.2025-2283.

[16] Safwat, M., and Devasia, S., ''Accurate Decentralized Information Communication for Effective Decisions in Robotic Networks,'' *IEEE/ASME Transactions on Mechatronics*, 2024, pp. 1–12. https://doi.org/10.1109/TMECH.2024.3477307.

[17] Kuhn, H. W., ''The Hungarian Method for The Assignment Problem,'' *Naval Research Logistics Quarterly*, Vol. 2, No. 1-2, 1955, pp. 83–97. https://doi.org/https://doi.org/10.1002/nav.3800020109, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109.

[18] Korte, B., and Vygen, J., *NP-Completeness*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2018, pp. 385–421. https://doi.org/10.1007/978-3-662-56039-6_15, URL https://doi.org/10.1007/978-3-662-56039-6_15.

[19] Korte, B., and Vygen, J., *The Traveling Salesman Problem*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 527–562. https://doi.org/10.1007/978-3-540-71844-4_21, URL https://doi.org/10.1007/978-3-540-71844-4_21.

[20] Hoffman, A. J., and Kruskal, J. B., *Integral Boundary Points of Convex Polyhedra*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. https://doi.org/10.1007/978-3-540-68279-0_3, URL https://doi.org/10.1007/978-3-540-68279-0_3.

[21] Veinott, A. F., Jr., and Dantzig, G. B., ''Integral Extreme Points,'' *SIAM Review*, Vol. 10, No. 3, 1968, pp. 371–372. https://doi.org/10.1137/1010063, URL https://doi.org/10.1137/1010063.

[22] Giles, F., and Pulleyblank, W., ''Total Dual Integrality and Integer Polyhedra,'' *Linear Algebra and its Applications*, Vol. 25, 1979, pp. 191–196. https://doi.org/https://doi.org/10.1016/0024-3795(79)90018-1, URL https://www.sciencedirect.com/science/article/pii/0024379579900181.

[23] Edmonds, J., and Giles, R., ''A Min-Max Relation for Submodular Functions on Graphs,'' *Studies in Integer Programming*, Annals of Discrete Mathematics, Vol. 1, edited by P. Hammer, E. Johnson, B. Korte, and G. Nemhauser, Elsevier, 1977, pp. 185–204. https://doi.org/https://doi.org/10.1016/S0167-5060(08)70734-9, URL https://www.sciencedirect.com/science/article/pii/S0167506008707349.

[24] Schrijver, A., ''On Total Dual Integrality,'' *Linear Algebra and its Applications*, Vol. 38, 1981, pp. 27–32. https://doi.org/https://doi.org/10.1016/0024-3795(81)90005-7, URL https://www.sciencedirect.com/science/article/pii/0024379581900057.

[25] Mills-Tettey, G. A., Stentz, A., and Dias, M. B., ''The Dynamic Hungarian Algorithm for the Assignment Problem with Changing Costs,'' *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-07-27*, 2007.

[26] Lawler, E. L., *Combinatorial Optimization: Networks and Matroids*, Courier Corporation, 2001.

[27] *IBM ILOG CPLEX Optimizer*, IBM Corporation, 2019. URL http://www.ibm.com/software/integration/optimization/cplex-optimizer.

[28] MOSEK ApS, *MOSEK Optimizer*, 2019. URL https://www.mosek.com.

[29] Mitchell, S., O'Sullivan, M., and Dunning, I., ''PuLP: A Linear Programming Toolkit for Python,'' *INFORMS Journal on Computing*, Vol. 23, No. 4, 2011, pp. 631–633.

[30] COIN-OR Foundation, ''CBC (COIN-OR Branch-and-Cut Solver),'' , 2019. URL https://github.com/coin-or/Cbc.

[31] COIN-OR Foundation, ''CLP(COIN-OR Linear Programming Solver),'' , 2019. URL https://github.com/coin-or/Clp.

[32] Bolusani, S., Besançon, M., Bestuzheva, K., Chmiela, A., Dionísio, J., Donkiewicz, T., van Doornmalen, J., Eifler, L., Ghannam, M., Gleixner, A., Graczyk, C., Halbig, K., Hedtke, I., Hoen, A., Hojny, C., van der Hulst, R., Kamp, D., Koch, T., Kofler, K., Lentz, J., Manns, J., Mexi, G., Mühmer, E., Pfetsch, M. E., Schlösser, F., Serrano, F., Shinano, Y., Turner, M., Vigerske, S., Weninger, D., and Xu, L., ''The SCIP Optimization Suite 9.0,'' Technical report, Optimization Online, February 2024. URL https://optimization-online.org/2024/02/the-scip-optimization-suite-9-0/.

[33] Huangfu, Q., and Hall, J. A. J., ''Parallelizing the Dual Revised Simplex Method,'' *Mathematical Programming Computation*, Vol. 10, No. 1, 2018, pp. 119–142. https://doi.org/10.1007/s12532-017-0130-5.

[34] Gurobi Optimization, LLC, ''Gurobi Optimizer Reference Manual,'' , 2023. URL https://www.gurobi.com.

[35] Dolan, E. D., and Moré, J. J., ''Benchmarking Optimization Software with Performance Profiles,'' *Mathematical Programming*, Vol. 91, No. 2, 2002, pp. 201–213. https://doi.org/10.1007/s101070100263, URL https://doi.org/10.1007/s101070100263.