2024 Tenth Indian Control Conference (ICC)
9 - 11 December, 2024
IISER Bhopal, India

A distributed algorithm for assigning UAVs to multiple heterogeneous positions of interest

Gangadari Varaprasad, Chayan Bhawal and Indrani Kar

***Abstract*— Unmanned Aerial Vehicles (UAVs) powered by rechargeable batteries face constraints in flight duration and computational capacity. To overcome these limitations, employing multiple UAVs and efficient algorithms becomes essential for handling extensive tasks beyond the capabilities of a single UAV. Many applications, including search and rescue missions, surveillance of tall buildings, and maintenance of heritage sites, often involve diverse and dispersed positions of interest (POIs). When UAVs are stationed across multiple depots and require deployment from these locations, a UAV assignment problem is introduced, considering the heterogeneity of POIs. This assignment problem is modelled as an integer linear programming (ILP) problem aimed at minimizing the total distance travelled by all UAVs. A distributed dual ascent algorithm is proposed to solve the ILP problem efficiently.**

I. INTRODUCTION

Quadrotor UAVs have a wide range of applications, including surveying, agricultural monitoring, infrastructure inspection, search and rescue, disaster response, wildlife conservation, environmental monitoring, delivery, aerial photography, and recreation [1]. Most commercially available quadrotors are battery-powered, with limited flight times of approximately 15-30 minutes [2] due to space and weight constraints. This necessitates using multiple UAVs for larger tasks. In missions like search and rescue, scanning tall buildings, or post-earthquake maintenance of heritage buildings, multiple separated structures, referred to as positions of interest (POIs), must be scanned. UAVs in such missions may not be co-located; they may be located across different depots. This scenario, where UAVs are launched from multiple depots to heterogeneous POIs, motivates the problem addressed in this paper: assigning UAVs from multiple depots to various POIs.

Assignment problems have been studied in the literature for a long time. In 1955, a classical solution to the assignment problem called the Hungarian algorithm was proposed in [3]. In [4], the algorithm was modified to the Kuhn-Munkres algorithm (KMA). There has been work in this area in recent years as well. A KMA-based assignment of UAVs to fixed target positions in the air for a vertical take-off problem is presented in [5]. Authors in [6] proposed a heuristic for the ground-to-air assignment problem of UAVs that minimizes the total distance travelled by the UAVs. A cooperative multi-target allocation method for multiple UAVs, utilizing the differential evolutionary algorithm is proposed in [7]. Authors in [8] have provided a comparison of the original KMA, KMA adapted for GPU (graphics processing unit) and a heuristic algorithm proposed in [6].

Assignment problems, such as those involving multiple UAVs and POIs, can be formulated as ILP problems. An effective method for solving ILP problems is to use Lagrangian relaxation combined with dual ascent algorithms. This paper addresses an assignment problem that aims to optimally allocate UAVs to POIs, considering the work demand at each POI. To our knowledge, no existing literature has tackled this scenario by formulating the assignment as an ILP problem and solving it using dual ascent algorithms.

Dual ascent algorithms iteratively optimize a dual problem, ensuring monotonous improvement (non-deterioration) of the dual objective [9]. The first dual-ascent algorithm for ILP was introduced in [10]. The authors in [11] proposed a dual-decomposition method for a class of MILP (Mixed Integer Linear Programming) problems to solve the problem in a distributed fashion. Similarly, [12] proposed a dual-ascent algorithm to solve the MILP problem addressed in [11]. However, these algorithms have a limitation: convergence is only asymptotically achieved when the number of agents (UAVs, in our case) is *significantly* greater than the number of POIs. The algorithm we proposed has no limitations on the ratio of UAVs to POIs. While this paper does not delve into convergence proofs, the simulation results are promising and demonstrate convergence even when the number of UAVs is similar to the number of POIs.

It is well known that ILP problems are challenging to solve, and most commercially available MILP solvers implement various search algorithms such as branch and bound, cutting planes, and branch and cut, among others [13]. Unlike standard MILP solvers that aim to solve any MILP problem, our approach is specifically designed to suit applications similar to the one presented in this paper. Thus, the main contribution of this paper is the proposal of a variant of the distributed dual ascent algorithm to address an optimization problem for UAV assignment, as outlined in Section II. Simulation results demonstrate that the proposed distributed algorithm could outperform commercially available MILP solver INTLINPROG. The solution methodology proposed in this paper is inspired by the approaches discussed in [11] and [12].

The subsequent sections of this paper are organized as follows: Section II presents the detailed mathematical formulation of the problem. Section III presents the simulation results for various cases. Section IV presents the concluding remarks and future research prospects.

All the authors are with the Department of Electronics and Electrical Engineering, IIT Guwahati, India. The email ids of the authors in order are: ganga.varaprasad@iitg.ac.in, bhawal@iitg.ac.in, indranik@iitg.ac.in.

II. PROBLEM FORMULATION

Consider a scenario where multiple UAV launching stations (depots) are located throughout a city, as illustrated in Figure 1. In the event of an earthquake or disaster that potentially damages buildings, it becomes necessary to scan them for maintenance. Suppose certain buildings are identified as positions of interest (POIs). When a disaster occurs, UAVs are deployed from their respective depots, travel to their assigned POIs, perform the required coverage, and return to their depots. Since POIs vary in dimensions, each requires a different amount of coverage, i.e., different work demands. First, we explain how the optimization problem is formulated and then formalize the same.

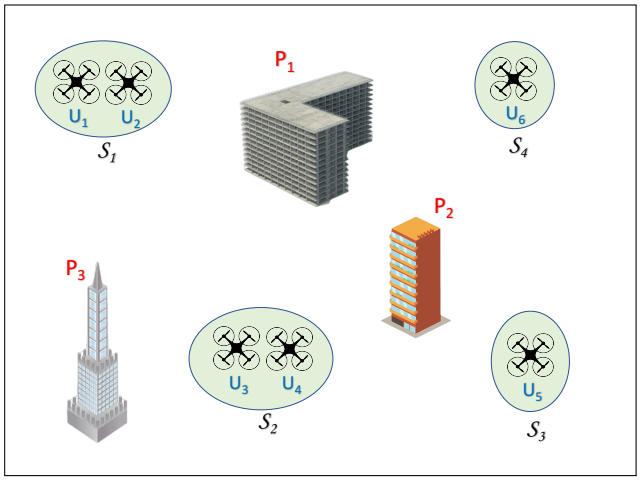


Fig. 1. A scenario with multiple depots $\{S_1,...,S_4\}$, UAVs $\{U_1,...,U_6\}$ and POIs $\{P_1,P_2,P_3\}$

Assumptions:

1) The total number of UAVs at hand equals or exceeds the number of POIs. This ensures that each POI is assigned at least one UAV and no UAV is assigned to multiple POIs.
2) All the UAVs together have sufficient battery capacity to perform the coverage and safely return to their respective depots.
3) All the UAVs are similar and have the same battery capacity.

In scenarios where the number of POIs exceeds the number of UAVs, some UAVs would need to be assigned to multiple POIs to ensure complete coverage. However, this paper does not address such scenarios. This will be part of our future work.

Optimization function: A UAV performs no useful work while travelling between its depot and the assigned POI. Therefore, we aim to find an assignment that minimizes these travel distances. The cost function of the optimization problem is defined as the sum of all such Euclidean distances the UAVs would travel for a given assignment (solution).

Constraints: A few constraints must be met for the assignment problem to be meaningful.

1) Each UAV must be assigned to exactly one POI.
2) The assignment must ensure that all UAVs have enough battery power to travel from their respective depots to the assigned POIs, complete their tasks, and return to their depots.

Note that while each UAV is assigned to only one POI, each POI may be assigned multiple UAVs. Additionally, since our focus is on assigning UAVs to POIs rather than assigning depots to POIs, we can assume that only one UAV is launched from each depot in the problem formulation without any loss of generality.

To formalize the problem, we define the notation to be used as follows:

$\mathbb{R}$	*Set of real numbers.*
$\mathbb{Z}_+$	*Set of positive integers.*
$\mathbb{R}_+$	*Set of positive real numbers.*
$\mathbb{F}^m$	*Set of m-tuple vectors with elements from set* $\mathbb{F}$.
$\mathbb{F}^{n\times m}$	*Set of* $n\times m$ *matrices with elements from set* $\mathbb{F}$.
$S =: \{S_1,S_2,S_3,...\}$	*Set of depots.*
$U =: \{U_1,U_2,U_3,...\}$	*Set of UAVs.*
$P =: \{P_1,P_2,P_3,...\}$	*Set of POIs.*
$N_u \in \mathbb{Z}_+$	*Number of UAVs in the swarm*
$N_p \in \mathbb{Z}_+$	*Number of POIs*
i	*index of UAV, i.e.,* $i \in \{1,\cdots,N_u\}$.
j	*Index of POI, i.e.,* $j \in \{1,\cdots,N_p\}$.
$\alpha \in \mathbb{R}_+$	*Battery consumption per unit distance traveled by a UAV.*
$M \in \mathbb{R}_+$	*Additional battery power required as a safety margin by a UAV.*
$A =: [a_{ij}] \in \mathbb{R}^{n\times m}$	*A is an* $n\times m$ *matrix with* a_{ij} *being the element at i-th row and j-th column of A.*
$A =: [a_i] \in \mathbb{R}^n$	*A is an n-tuple vector with* a_i *being the i-th element of A.*
$C =: [C_i] \in \mathbb{R}_+^{N_u}$	C_i *is the battery capacity of UAV* U_i.
$W =: [W_j] \in \mathbb{R}_+^{N_p}$	W_j *is the work demand at POI* P_j.
$D =: [d_{ij}] \in \mathbb{R}_+^{N_u\times N_p}$	d_{ij} *is the Euclidean distance between UAV* U_i *and POI* P_j.

A. ILP formulation:

Let us consider that there are N_u UAVs to be assigned to N_p POIs. Let the binary matrix $x =: [x_{ij}] \in \mathbb{Z}_+^{N_u\times N_p}$ be the solution matrix representing the assignment of UAVs to POIs, where $x_{ij} = 1$ implies that U_i is assigned to P_j.
In addition to the previously defined notation, define $B =: [B_{ij}] \in \mathbb{R}_+^{N_u\times N_p}$, where B_{ij} is the net battery capacity available for U_i when assigned to P_i, after accounting for the battery consumption for travelling to the POI and back to the depot.

Having defined all the notation, the optimization problem is formulated as an ILP problem, as shown in equation (1), which can be solved using any MILP solver.

$$\min_{x_{ij}} \quad J = \sum_{j=1}^{N_p} \sum_{i=1}^{N_u} d_{ij} x_{ij} \tag{1a}$$

subject to:

$$\sum_{j=1}^{N_p} x_{ij} = 1, \qquad \forall i \tag{1b}$$

$$\sum_{i=1}^{N_u} B_{ij} x_{ij} > W_j + M, \qquad \forall j \tag{1c}$$

The meaning of the optimization problem is as follows:

Optimization function: For any given assignment, equation (1a) is the sum of all the distances that UAVs would travel between UAVs and POIs.

$$\text{For instance, let } x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix} \implies N_u = 3 \text{ and } N_p = 2, \tag{2}$$

where, x indicates that U_1 is assigned to P_2, while U_2 and U_3 are assigned to P_1.

Using equation (1a) for x in equation (2), we obtain $J = d_{12} + d_{21} + d_{31}$. Thus, J is the sum of the distances from U_1 to P_2, U_2 to P_1 and U_3 to P_1. In this way, different values of x yield different J. The objective is to find x that provides the minimum value of J while satisfying the constraints.

Constraints:
1) Inequality (1b) ensures that each UAV is assigned to only one POI.
2) Inequality (1c) ensures that each POI is assigned sufficient UAVs to perform the coverage.
On using the constraint equation (1c) for x given in equation (2), we obtain two inequalities:

$$\begin{aligned} B_{12} > W_2 + M \quad &\implies \quad B_{12} - W_2 > M \\ B_{21} + B_{31} > W_1 + M \quad &\implies \quad B_{21} + B_{31} - W_1 > M \end{aligned}$$

Thus, the total power remaining after a UAV completes a round trip from the depot to the POI and performs the coverage work at the POI must always exceed the safety margin M. The safety margin M ensures that a quadrotor can safely return to its depot even if it consumes more power than expected during the trip or coverage.

B. Distributed ILP:

To distribute the computation, we reformulate the problem in equation (1). We consider that a central workstation coordinates with the UAVs to solve the optimization problem. Let $x_i \in \mathbb{Z}_+^{N_p}$, $d_i \in \mathbb{R}_+^{N_p}$ and $B_i \in \mathbb{R}_+^{N_p}$ are made up of i^{th} rows of x, D and B respectively. Further, let A_i be a diagonal matrix whose diagonal elements are made of B_i and $b = W + M$. Now, equations (1a) and (1c) can be reformulated as equations (3a) and (3b) respectively.

$$\min_{x_1, x_2 \ldots, x_{N_u}} \quad J = \sum_{i=1}^{N_u} d_i^T x_i \tag{3a}$$

subject to:

$$\sum_{i=1}^{N_u} A_i x_i > b \tag{3b}$$

Note that constraint (1b) is not included in the above problem because the design of Algorithm 1 inherently ensures that this constraint is satisfied.

Solution methodology (Algorithm):
The solution methodology described below is a variant of dual ascent algorithms [9].
Let $\lambda \in \mathbb{Z}_+^{N_p}$ be Lagrange multipliers, then the Lagrangian can be written as follows:

$$\mathscr{L} = \sum_{i=1}^{N_u} d_i^T x_i + \lambda^T \left(b - \sum_{i=1}^{N_u} A_i x_i \right) \tag{4}$$

Now, the optimization problem can be written as follows:

$$\mathscr{P} = \min_{x_1, x_2 \ldots, x_{N_u}} \max_{\lambda} \left\{ \sum_{i=1}^{N_u} d_i^T x_i + \lambda^T (b - \sum_{i=1}^{N_u} A_i x_i) \right\} \tag{5}$$

The equation (5) is called a primal problem, and its dual can be written as follows:

$$\mathscr{D} = \max_{\lambda} \min_{x_1, x_2 \ldots, x_{N_u}} \left\{ \sum_{i=1}^{N_u} d_i^T x_i + \lambda^T (b - \sum_{i=1}^{N_u} A_i x_i) \right\} \tag{6}$$

Rearranging the terms, we have the following

$$\mathscr{D} = \max_{\lambda} \left\{ \lambda^T b + \min_{x_1, x_2 \ldots, x_{N_u}} \{ \sum_{i=1}^{N_u} (d_i^T - \lambda^T A_i) x_i \right\} \tag{7}$$

The dualized equation (7) has a form where the term $\sum_{i=1}^{N_u} (d_i^T - \lambda^T A_i) x_i$ is collectively minimized by the UAVs. While each UAV minimizes its local cost $(d_i^T - \lambda^T) x_i$, a modification of $(d_i^T - \lambda^T A_i) x_i$, where A_i is replaced by an identity matrix, a central station maximizes equation (7) in an iterative way explained below.
For a given x_i the derivative of the Lagrangian (4) is:

$$\mathscr{G} := \frac{\partial}{\partial \lambda} \mathscr{L} = b - \sum_{i=1}^{N_u} A_i x_i \tag{8}$$

Note that equation (8) essentially corresponds to constraints (3b), offering insights into constraint violations. The solution is deemed infeasible if any element of $\mathscr{G}$ is non-negative (indicating constraint violation). Hence, our focus lies on enhancing the violated constraints. We define the violation vector $\mathscr{V} := \mathscr{G} > 0$, where the symbol '>' is an element-wise logical operation. Essentially, $\mathscr{V}$ is derived by setting all positive elements of $\mathscr{G}$ to 1 and the remaining to 0. Now the update rule for λ in k^{th} iteration is defined as follows:

$$\lambda^{(k+1)} = \lambda^{(k)} + \eta \mathscr{V} \tag{9}$$

$\eta \in \mathbb{R}_+$ is the increment size. Note that $\mathscr{V} = 0$, which indicates a feasible solution (no constraint is violated), is a valid stopping criterion for the iterative process.

Algorithm 1 Assign(C, W, D, α, M)

1: $k = 1$
2: $\lambda = \begin{bmatrix} 0 & 0 & \dots & 0 \end{bmatrix}^T_{N_p \times 1}$
3: $\mathbf{1} = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^T_{N_u \times 1}$
4: $B = C\mathbf{1}^T - 2\alpha D$
5: $b = W + M$
6: **repeat**
7: **for** $i = 1$ **to** N_u **do**
8: $\hat{d}_i = d_i - \lambda^{(k)}$
9: $j^* = \underset{j}{argmin}(\hat{d}_{ij})$
10: $x_{ij} = \begin{cases} 1, & \text{for } j = j^* \\ 0, & \forall j \neq j^* \end{cases}$
11: **end for**
12: $A_i = \text{diag}(B_i)$
13: $\mathcal{G} = b - \sum_{i=1}^{N_u} A_i x_i$
14: $\mathcal{V}_j = \begin{cases} 1, & \text{if } \mathcal{G}_j > 0 \\ 0, & else \end{cases}$
15: $\lambda^{(k+1)} = \lambda^{(k)} + \eta \mathcal{V}$
16: $k = k + 1$
17: **until** $\mathcal{V}_j = 0 \quad \forall j \in \{1, 2, ..., N_p\}$
18: **return** x

Thus, we have a distributed framework to solve an ILP problem in an iterative fashion in which the cost minimisation is performed by UAVs locally, whereas verification of solution feasibility and the update of λ is performed by a central station.

To illustrate the working principle of Algorithm 1, let us consider a simple example as shown in Figure 2 :

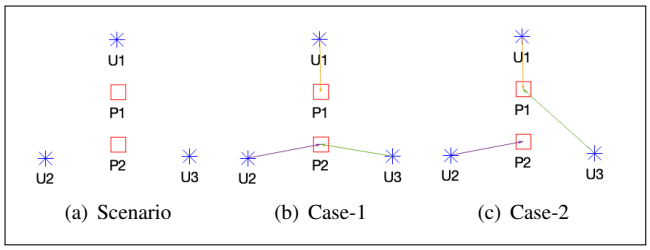


Fig. 2. Example

Here the Distance matrix D, Battery capacities C and α are $\begin{bmatrix} 10 & 20 \\ 17 & 12 \\ 15 & 11 \end{bmatrix}$, $\begin{bmatrix} 100 \\ 100 \\ 100 \end{bmatrix}$ and 0.1 respectively.

Case-1: Work Demand $W = \begin{bmatrix} 80 & 150 \end{bmatrix}^T$

The UAVs execute lines 8-10 of Algorithm 1 as follows: Initially, $\lambda = 0$, so each UAV minimizes its local cost $(d_i)^T x_i$. For the given example, $d_1 = [10 \quad 20]^T, d_2 = [17 \quad 12]^T$ and $d_3 = [15 \quad 11]^T$. The minimum elements of d_1, d_2, d_3 are $10, 12, 11$ respectively, leading to the solution $x_1 = [1 \quad 0]^T, x_2 = [0 \quad 1]^T$ and $x_2 = [0 \quad 1]^T$.
Next, the central station executes the lines 12-17 of the Algorithm 1. It can be verified that we obtain $\mathcal{V} = [0 \quad 0]^T$ which is the stopping criterion of the algorithm. Thus the algorithm gives the optimal solution as shown in Figure 2(b) in just 1 iteration.

Case-2: Work Demand $W = \begin{bmatrix} 150 & 80 \end{bmatrix}^T$

Lines 8-10 of Algorithm 1 are executed by the UAVs just as in the previous case (Case 1), resulting in the solution $x_1 = [1 \quad 0]^T, x_2 = [0 \quad 1]^T$ and $x_2 = [0 \quad 1]^T$.
Next, the central station executes lines 12-17 of the Algorithm 1. It can be verified that $\mathcal{V} = [1 \quad 0]^T$. According to line 15 of the algorithm, $\lambda^{(2)} = \lambda^{(1)} + \eta \mathcal{V} => \lambda = [0 \quad \eta]^T$. In the next iteration each UAV minimizes the local cost $(d_i - \lambda)^T x_i$ where $d_1 = [10 - \eta \quad 20]^T, d_2 = [17 - \eta \quad 12]^T$ and $d_3 = [15 - \eta \quad 11]^T$. Note that the solution will not change in this iteration if $\eta \leq 4$, implying that more iterations are needed for the solution to change. So, if $\eta = 4.1$ then $d_1 = [5.9 \quad 20]^T, d_2 = [12.9 \quad 12]^T$ and $d_3 = [10.9 \quad 11]^T$. The minimum elements of d_1, d_2, d_3 are $5.9, 12, 10.9$ respectively, leading to the solution $x_1 = [1 \quad 0]^T, x_2 = [0 \quad 1]^T$ and $x_2 = [1 \quad 0]^T$. This results in $\mathcal{V} = [0 \quad 0]^T$, which meets the stopping criterion of the algorithm. Thus, the algorithm provides the optimal solution in just two iterations for an appropriate η, as shown in Figure 2(c).

Remark: As illustrated by the previous example in Figure 2, the algorithm provides the optimal solution in just one iteration if the constraints are inactive. However, if the constraints are active, it takes additional iterations based on the value of η. Therefore, selecting the appropriate value of η is crucial. η can be fixed for all iterations or vary for each iteration. This paper does not address this issue, leaving it for future work.

It should be noted that algorithm 1 has some limitations. It fails to arrive at a feasible solution when there are significant differences in the battery capacities of the UAVs., i.e., when the solution depends on these differences. An example of this failure is illustrated in Figure 3 with $C = \begin{bmatrix} 70 & 100 \end{bmatrix}^T$ and $W = \begin{bmatrix} 80 & 50 \end{bmatrix}^T$.

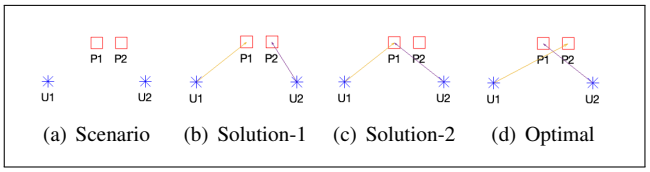


Fig. 3. An example where the algorithm fails

If we apply Algorithm 1 to the example in Figure 3, we observe that the solution oscillates between the solutions shown in Figure 3(b) and Figure 3(c), neither of which are feasible. Consequently, the algorithm never reaches the optimal solution shown in Figure 3(d).

Note that although Algorithm 1 fails in certain cases, the problem formulation in equation (1) is valid for all these cases and can be solved by any MILP solver to obtain the optimal solution. Determining whether it can be solved in a distributed manner is part of our future work.

The next section provides simulation results supporting the formulated optimization problem and the algorithm.

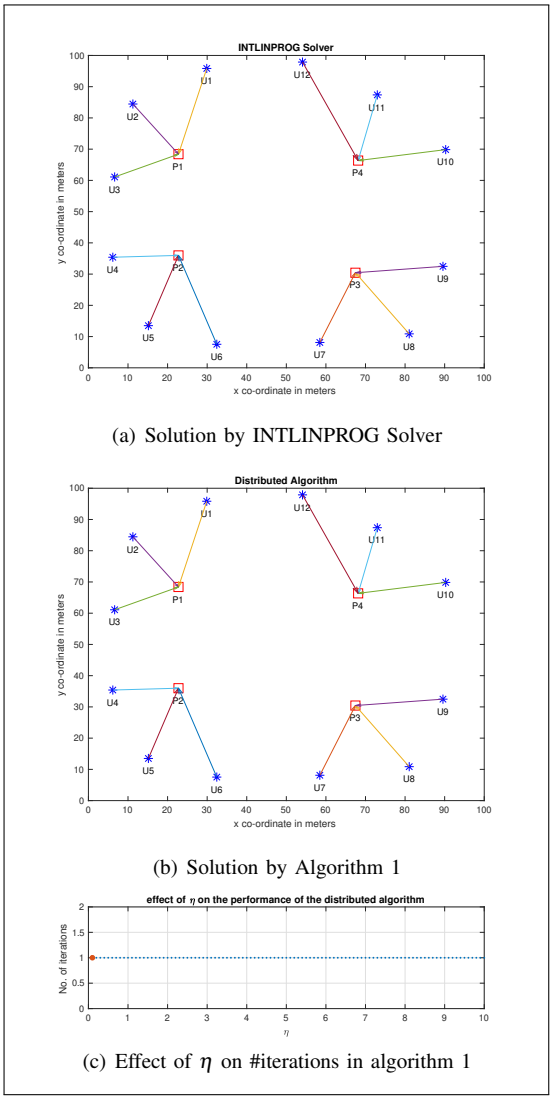
III. SIMULATION RESULTS

To demonstrate how the formulated problem works, we consider a scenario with the following specifications:

$$N_u = 12, \quad N_p = 4, \quad \alpha = 0.1 \quad \text{and } M = 10$$

Note that the work demand W_i is a percentage of a UAV's maximum battery capacity. For instance, $W_j = 250$ means that the POI P_j requires at least 250% of a UAV's maximum battery capacity, which can be met by multiple UAVs. Each UAV is assumed to be fully charged to 100% battery capacity. The simulation is conducted in MATLAB for two scenarios, and the results are presented below.

***Case-1**: Identical POIs,* $W = \begin{bmatrix}250 & 250 & 250 & 250\end{bmatrix}^T$



(a) Solution by INTLINPROG Solver

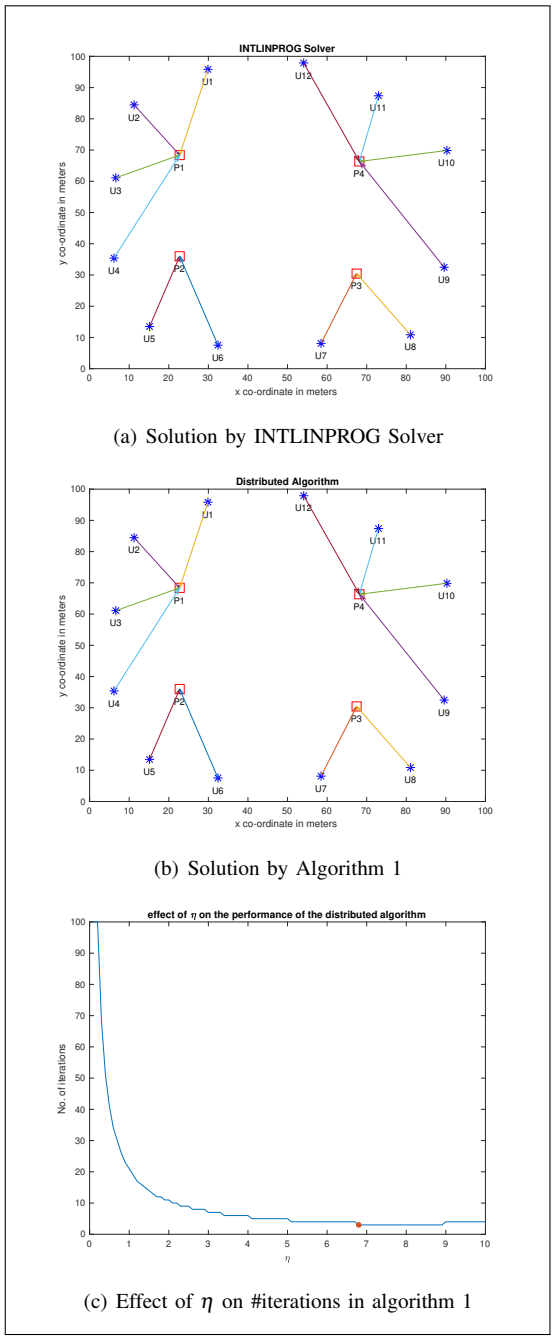
(b) Solution by Algorithm 1

(c) Effect of η on #iterations in algorithm 1

Fig. 4. Simulation for Identical POIs

For the given W, it is evident that each POI requires at least three UAVs. This is confirmed by the simulation results shown in Figures 4(a) and 4(b). Each POI is assigned its three nearest UAVs, resulting in the minimum cost J. Notably, the constraints are inactive in this case, allowing the solution to be obtained in just one iteration. This is confirmed by the simulation result in Figure 4(c).

***Case-2**: Heterogeneous POIs,* $W = \begin{bmatrix}350 & 150 & 150 & 350\end{bmatrix}^T$



(a) Solution by INTLINPROG Solver

(b) Solution by Algorithm 1

(c) Effect of η on #iterations in algorithm 1

Fig. 5. Simulation for heterogeneous POIs

For the given W, it is evident that P_1 and P_4 require at least four UAVs, whereas P_2 and P_3 require at least two UAVs. This is confirmed by the simulation results in Figures 5(a) and 5(b) while minimizing the cost function J. Notably, the constraints are active in this case, necessitating multiple iterations to obtain the solution. The simulation result in Figure 5(c) illustrates how the number of iterations needed to obtain the solution depends on η. Figure 5(c) shows that

$\eta = 6.8$ is an appropriate choice to achieve the solution in three iterations.

Performance Comparison:

To evaluate the performance of Algorithm 1 against MATLAB's built-in solver INTLINPROG, six different cases varying in the number of UAVs and POIs were considered as follows:

$$(N_u, N_p) \in \{(10,5),(20,10),(30,15),(40,20),(50,25),(60,30)\}$$

250 configurations were randomly generated for each case, and the simulations were performed on a workstation with the specifications shown in TABLE I. The results of the simulations are presented in Figure 6:

CPUName :	Intel(R) Xeon(R) Silver 4214R CPU @ 2.40GHz
Clock :	1000.091 MHz
Cache :	16 MB
NumCPUs :	2
TotalCores :	24
TotalMemory :	250 GB
OSType :	Linux
OSVersion :	4.18.0-305.19.1.el8_4.x86_64

TABLE I

WORKSTATION DETAILS

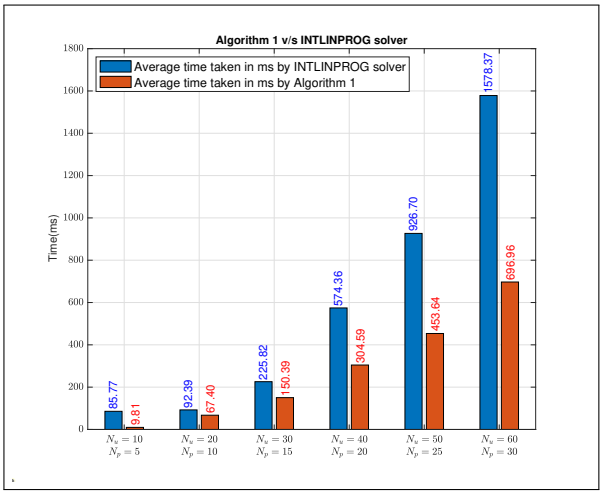


Fig. 6. Comparison of Algorithm 1 with INTLINPROG solver

Figure 6 compares the average time taken by INTLINPROG solver and Algorithm 1 to obtain a solution for each case. It can be seen that Algorithm 1 has the potential to be significantly faster than the solver with an appropriate choice of η. For the above simulations, η was determined through extensive trial and error.

IV. CONCLUSIONS

In this paper, we tackled the problem of assigning UAVs from multiple depots to heterogeneous POIs. We formulated this assignment problem as an ILP problem with the objective of minimizing the total distance traveled by the UAVs. To solve the formulated problem, we proposed a distributed algorithm that iteratively refines the solution, progressively moving towards optimality. The key advantages of this distributed algorithm are that it operates without requiring a solver, distributes computation across multiple UAVs, and has the potential to be much more efficient than the INTLINPROG solver.

However, the algorithm lacks sophistication as the value of η, which influences its performance, is not determined analytically but rather selected through extensive trial and error. The optimal solution appears achievable with an appropriate choice of η. Future work will focus on finding ways to choose the best η for the algorithm and developing a more effective update rule for λ to ensure the improvement of the solution at every iteration. In this paper, the solution obtained by Algorithm 1 is compared with the solution provided by the INTLINPROG solver to confirm the solution's optimality. An analytical proof for the convergence of Algorithm 1 to its optimal solution will be part of our future research.

REFERENCES

[1] S. H. Alsamhi, O. Ma, M. S. Ansari, and F. A. Almalki, "Survey on collaborative smart drones and internet of things for improving smartness of smart cities," *IEEE Access*, vol. 7, pp. 128125–128152, 2019.

[2] M. Krznar, P. Piljek, D. Kotarski, and D. Pavković, "Modeling, control system design and preliminary experimental verification of a hybrid power unit suitable for multirotor UAVs," *Energies*, vol. 14, no. 9, p. 2669, 2021.

[3] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[4] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[5] D. Hernández, J. M. Cecília, C. T. Calafate, J.-C. Cano, and P. Manzoni, "The Kuhn-Munkres algorithm for efficient vertical takeoff of UAV swarms," in *IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, pp. 1–5, IEEE, 2021.

[6] F. Fabra, J. Wubben, C. T. Calafate, J. C. Cano, and P. Manzoni, "Efficient and coordinated vertical takeoff of UAV swarms," in *IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pp. 1–5, IEEE, 2020.

[7] Y. Song, Q. Xi, X. Xing, and B. Yang, "Multi-UAV cooperative multi-target allocation method based on differential evolutionary algorithm," in *39th Chinese Control Conference (CCC)*, pp. 1655–1660, IEEE, 2020.

[8] J. Wubben, D. Hernández, J. M. Cecilia, B. Imberón, C. T. Calafate, J.-C. Cano, P. Manzoni, and C. K. Toh, "Assignment and Take-Off Approaches for Large-Scale Autonomous UAV Swarms," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[9] P. Swoboda, J. Kuske, and B. Savchynskyy, "A Dual Ascent Framework for Lagrangean Decomposition of Combinatorial Problems," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4950–4960, 2017.

[10] O. Bilde and J. Krarup, "Sharp lower bounds and efficient algorithms for the simple plant location problem," in *Annals of Discrete Mathematics*, vol. 1, pp. 79–97, Elsevier, 1977.

[11] R. Vujanic, P. M. Esfahani, P. J. Goulart, S. Mariéthoz, and M. Morari, "A decomposition method for large scale MILPs, with performance guarantees and a power system application," *Automatica*, vol. 67, pp. 144–156, 2016.

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

[13] C. Contardo, A. Lodi, and A. Tramontani, "Cutting planes from the branch-and-bound tree: Challenges and opportunities," *INFORMS Journal on Computing*, vol. 35, no. 1, pp. 2–4, 2023.