Decentralized and Network-Aware UAV Service Deployment for Dependency-Driven Applications

Decentralized and Network-Aware UAV Service Deployment for Dependency-Driven Applications

Talip Tolga Sarı*‡, Christian Quadri†, Gökhan Seçinti‡§, Angelo Trotta*
*Department of Computer Science and Engineering, University of Bologna, Italy
†Department of Computer Science, Università degli Studi di Milano, Italy
‡ Department of Computer Engineering, Istanbul Technical University, Turkey
§ BTS - Digital Twin Application and Research Center, Istanbul Technical University, Turkey
Emails: ttolga.sari2@unibo.it, christian.quadri@unimi.it, secinti@itu.edu.tr, angelo.trotta5@unibo.it

***Abstract*—Unmanned Aerial Vehicle (UAV) swarms enable the rapid deployment of IoT services in dynamic and challenging environments. While these swarms offer flexibility and close proximity to sensing and actuation points, efficiently deploying interdependent services at scale remains a core challenge. Traditional centralized methods struggle to handle the complexity of large UAV networks, leading to increased latency and limited reliability. In this paper, we propose a decentralized approach to service deployment in UAV swarms. Our method relies on local information at each node, allowing UAVs to make their own assignment decisions. Over time, these decisions are iteratively refined as nodes exchange status updates and adapt to network changes. This process avoids the bottlenecks of centralized coordination and enables more responsive resource allocation. Simulation results show that our approach supports the successful deployment of a high number of tasks while maintaining low latency. These findings indicate that decentralized methods with local resource knowledge, improves both scalability and responsiveness in UAV-based IoT systems.**

***Index Terms*—UAV, Internet of Things, decentralized resource allocation, service deployment, edge computing**

I. Introduction

Time plays a critical role in many Internet of Things (IoT) applications, including industrial automation, cooperative robotics, autonomous systems, and immersive human-machine interactions [1]. In these domains, meeting strict latency and timing requirements is essential. Ensuring that sensing, computation, and actuation occur within tight time bounds enables stable control loops, reduces safety risks, and supports real-time decision-making. Unmanned Aerial Vehicle (UAV) swarms offer a flexible and mobile platform for deploying IoT services in hard-to-reach or dynamic environments [2]. By forming distributed airborne networks, UAVs can collaboratively provide sensing and computing capabilities closer to where data is generated and consumed. However, deploying interdependent services across large UAV swarms is challenging. Traditional centralized methods for assigning tasks struggle to scale, as they cannot quickly adapt to changes in network conditions or workloads. This is especially problematic when dealing with services, where they may depend on others, adding complexity to the placement and scheduling of tasks. In time-critical scenarios, any delay in selecting suitable nodes or in distributing tasks can degrade performance and even compromise safety. Centralized solutions, which rely on a single decision-making point, is characterized by long communication delays and may have limited awareness of local conditions. They also create bottlenecks, leading to high latency and reduced reliability [3].

To address these issues, we propose a fully distributed method for deploying interdependent, time-critical IoT services on UAV swarms. By iteratively splitting the services to different UAVs with sufficient computational and communication resources, the system adapts to changing conditions without relying on global knowledge. This distributed strategy reduces latency, improves scalability, and helps ensure that time constraints are met.

Our method employs a two-phase process: topology-aware capacity aggregation and service dependency-aware task distribution. In the first phase, UAVs iteratively refine their topology aware metric based on their available resources and the communication characteristics of their neighbors. This metric, aids in identifying resource-abundant nodes while avoiding bottlenecks that arise from over-utilization. In the second phase, the tasks within a service dependency graph are distributed efficiently by solving a constrained optimization problem that minimizes execution time while satisfying resource and dependency constraints. The algorithm is designed to handle diverse IoT scenarios, including dynamic environmental monitoring, industrial automation, and disaster recovery applications. By leveraging local computations and minimizing dependency on global data, it enables real-time responsiveness and resilience to network disruptions. The main contributions of this work are: (i) a distributed algorithm for distributing aggregated resource information through the UAV network; (ii) an iterative IoT services assignment procedure on UAV swarms that reduces end-to-end latency by avoiding congestion and ensuring that critical tasks run on suitable nodes; (iii) performance evaluation through simulations, showing that the proposed approach achieves lower execution time and better scalability compared to centralized methods.

The rest of the paper is organized as follows. Section II reviews related work. Section III introduces the system model, while Section IV formulates the optimization problem. Section V describes our distributed algorithm. Section VI presents simulation results and analysis, and Section VII concludes the

paper and suggests directions for future research.

II. Related Works

In this Section, we examine the current state of task and service deployment for various types of networks and use cases. The work in [4] presents RAIN4C, a multi-layer framework combining satellite, aerial, and terrestrial networks to address critical connectivity and resource challenges using drone swarms and CubeSats. Next, [5] explores UAV trajectory planning, DAG task scheduling, and service deployment in UAV-enabled edge computing, utilizing a DRL-based method to enhance task completion and reduce latency. Additionally, [6] introduces a UAV-based Flying Edge Intelligence (FEI) system for smart environmental monitoring to improve data quality and reduce the Age of Information. The work [7], integrates distributed machine learning and edge computing to proactively place UAV base stations and optimize network throughput using user trajectory and content demand predictions. Further, [8] proposes a satisfaction-oriented framework for UAV-enabled MEC networks, leveraging a novel task priority model and genetic algorithms to balance delay and energy constraints. Similarly, [9] presents a two-phase DAG-aware framework for geo-distributed data analytics, reducing data transfer costs without increasing execution time. [10] employs a DQN-based method for DAG scheduling and UAV deployment in multi-UAV edge systems, enhancing task latency and energy efficiency. Moreover, [11] proposes a DRL-based methodology for multi-user edge service orchestration in 5G networks, optimizing resource allocation and service scheduling with a multi-objective reward function. Finally, [12] introduces a distributed service discovery protocol for heterogeneous robotic systems-of-systems, enhancing dynamic, multi-radio environment management. Our work proposes a fully distributed, service DAG-aware deployment scheme considering network topology and graph centrality features.

III. System Model

In this Section, we present the system model used to represent the UAV-based network, the services that must be deployed, the available resources, and how service requests are assigned to devices. Figure 1 illustrates the considered scenario, where a UAV network is connected to ground devices and two different sets of interdependent services are deployed.

A. Network Model

Consider a heterogeneous network composed of two types of devices: *UAVs*: $U = \{u_1, u_2, \ldots, u_{N_U}\}$, and *ground user devices*: $G = \{g_1, g_2, \ldots, g_{N_G}\}$. We define the set of all devices as: $D = U \cup G = \{d_1, d_2, \ldots, d_{N_D}\}$, where $N_D = N_U + N_G$. Each device d_i is placed at a position $p(d_i) = \langle x_i, y_i, z_i \rangle$ in three-dimensional space, where $d_j \in G$ have $p(d_j) = \langle x_j, y_j, 0 \rangle$. We assume that a device d_i can communicate with d_j if there is a link $\langle d_i, d_j \rangle$. The set of all potential links is: $E = \{\langle d_i, d_j \rangle \mid d_i, d_j \in D,\ d_i \neq d_j\}$. Each link $\langle d_i, d_j \rangle \in E$ is associated with a link quality $q(\langle d_i, d_j \rangle)$ measured in bits per second (bps).

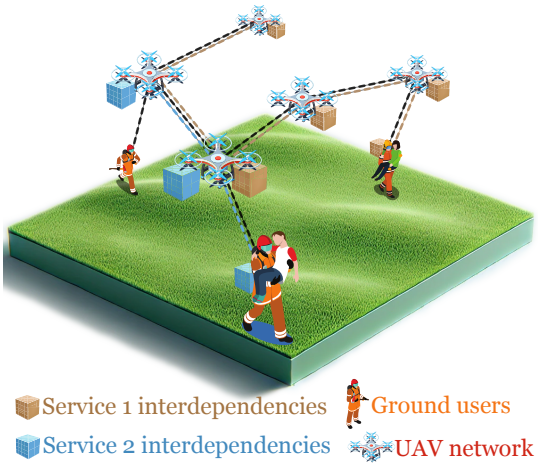


Fig. 1: Scenario overview

The network can be represented by the graph: $N = (D, E)$, and we consider a *communication tree* $CT = (D, E_{CT})$ with $E_{CT} \subseteq E$, representing a spanning tree used to route data flows in a predefined manner. This tree structure ensures that there is a unique path between any pair of devices in the tree.

B. Service Model

We have a set of services $S = \{s_1, s_2, \ldots, s_{N_S}\}$ that can be deployed in the network. The services exhibit interdependencies, represented as a Directed Acyclic Graph (DAG), known as the Service Dependency Graph (SDG): $SDG = (S, L)$, where $L = \{\langle s_i \rightarrow s_j \rangle \mid s_i, s_j \in S,\ s_i \neq s_j,$ and s_i depends on $s_j\}$. If a service s_i is requested, all services it depends on must be deployed and available. We define a *Service Sub-DAG* for a service s_i: $S\text{-}SDG(s_i) = (S_{s_i}, L_{s_i})$, where $S_{s_i} = \{s_j \in S \mid \exists$ path from s_i to s_j in $SDG\}$ and $L_{s_i} = \{\langle s_p \rightarrow s_q \rangle \in L \mid s_p, s_q \in S_{s_i}\}$. Each service, when activated, involves data transmission between the requesting user and the deployed service nodes, as well as among interdependent services. Here, $\text{data}(s_i, s_j)[B/s]$ is the data exchanged from service s_i to service s_j.

Let $REQ = \{r_1, r_2, \ldots\}$ be the set of requests for services. Each request r_j is characterized by: $r_j = [s_{r_j}, d_{r_j}]$, where $s_{r_j} \in S$ is the requested service and $d_{r_j} \in D$ is the device serving as the user interface. Deploying a requested service s_{r_j} involves assigning it and all the services in its sub-DAG $S_{s_{r_j}}$ to devices in D.

C. Resource Model

Each device $d_k \in D$ has a computational capacity: C_{d_k}[computations/s]. Similarly, each service $s_i \in S$ requires a certain amount of computational resources c_{s_i} to run. When a service is deployed on a device, it partially consumes available computational resources. The sum of the computational requirements of all services assigned to a device cannot exceed that device's capacity. We define a binary assignment variable:

$$\varphi_{s_i, r_j}^{d_k} = \begin{cases} 1, & \text{if } s_i \text{ of request } r_j \text{ is assigned to } d_k \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

As short notation, we define d_{s_i,r_j} as the device where the service s_i of request r_j is deployed, i.e., $\varphi_{s_i,r_j}^{d_{s_i,r_j}} = 1$. Here, data must flow through the network to support the dependencies. For each dependent pair $\langle s_m \to s_n \rangle \in L_{s_{r_j}}$, if s_m is assigned to d_a and s_n is assigned to d_b, then data must be transmitted along a path: $\text{Path}(d_a, d_b) = \{\langle d_a, d_{k_1} \rangle, \ldots, \langle d_{k_n}, d_b \rangle\} \subseteq E_{CT}$.
The data flow rate for a given request r_j on a link $\langle d_p, d_q \rangle$ is:

$$\text{df}^{r_j}(d_p, d_q) = \sum_{\langle s_m \to s_n \rangle \in L_{s_{r_j}}} \sum_{\substack{d_a, d_b \in D \\ d_a \neq d_b}} \varphi_{s_m,r_j}^{d_a} \varphi_{s_n,r_j}^{d_b} \cdot \cdot \text{ip}_{d_a,d_b}(d_p, d_q) \cdot \text{data}(s_m, s_n) \tag{2}$$

where $\text{ip}_{d_a,d_b}(d_p, d_q)$ is an indicator function that is 1 if link $\langle d_p, d_q \rangle$ is on the path between d_a and d_b, i.e., $\langle d_p, d_q \rangle \in \text{Path}(d_a, d_b)$, and 0 otherwise. The total data flow on a link is:

$$\text{DF}(d_p, d_q) = \sum_{r_j \in REQ} \text{df}^{r_j}(d_p, d_q) \tag{3}$$

D. Execution Time

The execution time of a requested service s_{r_j} depends on both the computational effort required to execute it on the assigned device and the delays introduced by its dependencies. When a service s_i of request r_j is assigned to a device d_{s_i,r_j}, the computation time $T_{\text{comp}}(s_i, d_{s_i,r_j})$ is determined by the ratio of the service's computational requirements to the device's computational capacity:

$$T_{\text{comp}}(s_i, d_{s_i,r_j}) = \frac{\sum_{s \in S, r \in REQ} c_s \cdot \varphi_{s,r}^{d_{s_i,r_j}}}{C_{d_{s_i,r_j}}} \tag{4}$$

In addition, the service may need data from all services it depends on; the time to receive this data is captured by $T_{\text{trx}}(s_i, s_k, r_j)$ for each dependency $\langle s_i \to s_k \rangle$:

$$T_{\text{trx}}(s_i, s_k, r_j) = \sum_{\langle d_a, d_b \rangle \in \text{Path}(d_{s_i,r_j}, d_{s_k,r_j})} T_{\text{trx}}^{\text{dev}}(d_a, d_b) \tag{5}$$

$$T_{\text{trx}}^{\text{dev}}(d_a, d_b) = \frac{\text{DF}(d_a, d_b)}{q(\langle d_a, d_b \rangle)} \tag{6}$$

We first define the execution time of a service s_i at a specific device d_{s_i,r_j}:

$$\text{ET}(s_i, d_{s_i,r_j}) = T_{\text{comp}}(s_i, d_{s_i,r_j}) + \max_{\langle s_i \to s_k \rangle \in L} T_{\text{trx}}(s_i, s_k, r_j) \tag{7}$$

Since s_i cannot complete before all the services it depends on have finished, we define the overall execution time $ET(s_i)$ by considering both its own execution and the execution times of all its dependencies:

$$\text{ET}(s_i) = \max\left\{ \text{ET}(s_i, d_{s_i,r_j}), \max_{\langle s_i \to s_k \rangle \in L} \text{ET}(s_k, d_{s_k,r_j}) \right\} \tag{8}$$

This ensures that the execution time of s_i accounts for the longest chain of dependent services in the Sub-DAG $S\text{-}SDG(s_{r_j})$, including their computation and transmission delays.

IV. Problem Formulation

We now present the problem of assigning services to devices in a UAV-ground device network, ensuring that each requested service is fully deployed with minimal completion time.
Our problem is to determine an assignment $\{\varphi_{s_i,r_j}^{d_k}\}$ that minimizes the Maximum Completion Time (MCT) among all requested services, respecting both the physical and logical constraints imposed by the network resources and the SDG. The optimization problem can be formally described as:

$$\text{find} \quad \varphi_{s_i,r_j}^{d_k} \in \{0,1\},\ \forall s_i \in S,\ r_j \in REQ,\ d_k \in U$$

$$\text{minimize} \quad \text{MCT} = \max_{r_j \in REQ} \text{ET}(s_{r_j}) \tag{9}$$

$$\text{s.t.} \quad \sum_{\substack{s_i \in S \\ r_j \in REQ}} \varphi_{s_i,r_j}^{d_k} \cdot c_{s_i} \leq C_{d_k}, \quad \forall d_k \in D \tag{10}$$

$$\text{DF}(d_i, d_j) \leq q(\langle d_i, d_j \rangle),\ \forall \langle d_i, d_j \rangle \in E \tag{11}$$

$$\sum_{d_x \in D} \varphi_{s_i,r_j}^{d_x} \leq \sum_{d_y \in D} \varphi_{s_k,r_j}^{d_y}, \quad \begin{matrix} \forall r_j \in REQ \\ \forall \langle s_i \to s_k \rangle \in L \end{matrix} \tag{12}$$

$$\sum_{d_x \in D} \varphi_{s_i,r_j}^{d_x} \leq 1,\ \forall r_j \in \text{REQ},\ \forall s_i \in S_{r_j} \tag{13}$$

where Equation 10 is about the computational capacity: the first set of constraints ensures that the total computational load of all services assigned to a device does not exceed its available computational capacity C_{d_k}. Equation 11 is about communication and it enforces that the total data flow $\text{DF}(d_i, d_j)$ passing through each link does not surpass its bandwidth limit $q(\langle d_i, d_j \rangle)$. The set of constraints of Equation 12 ensures that no service is deployed unless all the services it depends on have also been assigned, i.e., for each dependency $\langle s_i \to s_k \rangle \in L$, service s_i can only be deployed if s_k is deployed. This maintains the logical order imposed by the Service Dependency Graph (SDG). Constraint 13 ensure that each service is deployed at most once. Based on the successful deployed requests, we define the Request Delivery Ratio (RDR) index as the ratio between deployed services and the request list:

$$\text{RDR} = \frac{\sum_{r_j \in \text{REQ}} \sum_{d_k \in D} \varphi_{s_{r_j},r_j}^{d_k}}{|\text{REQ}|} \tag{14}$$

V. Proposed Distributed Algorithm

The system model considers a single service request at a time, assuming steady-state resource allocations without explicit timing. In contrast, our distributed algorithm operates iteratively, introducing the notion of time through discrete rounds. At each time step $t \in T = \{t_0, t_1, t_2, \ldots\}$, nodes exchange information, update local metrics, and make offloading decisions. This iterative, time-stepped process captures the algorithm's dynamic behavior, even though the underlying problem focuses on a single request. We structure our algorithm into two parts: *Topology Aware Aggregated Capacity*, where each node refines a metric reflecting its computational capabilities and communication links, and *DAG Splitting and Service Assignment*, which uses these metrics to decompose the service DAG and assign its parts to suitable nodes.

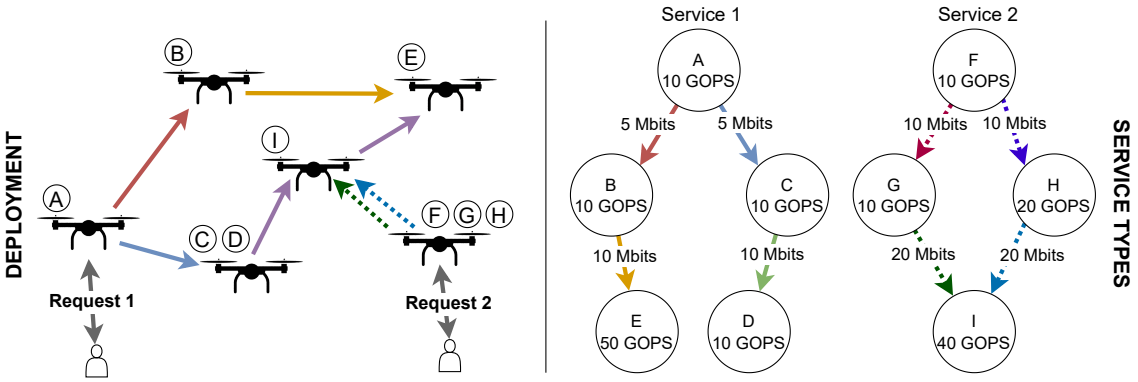


Fig. 2: Illustration of the service deployment framework for different service types.

Figure 2 provides a visual representation of the service request management in a UAV-ground device network. It highlights two service requests with their respective DAGs and showcases how the distributed deployment algorithm iteratively assigns tasks to nodes based on computational resources and communication links. The UAVs and ground nodes are annotated with their processing capacities and the bandwidth of their connecting links, emphasizing the resource constraints and the need for efficient task offloading.

A. Topology Aware Aggregated Capacity

Central nodes, due to their proximity to others, are often favored for selection. However, overuse of these nodes can cause inefficiencies and bottlenecks [13]. To address this, we introduce the topology-aware aggregated capacity (ϕ), which is iteratively updated through local interactions. This metric incorporates a node's service rate, its neighbors' service rates, and communication delays. The available service rate from node d_k to node d_i is defined as:

$$\frac{1}{\widetilde{T}^{\text{dev}}_{\text{trx}}(d_i, d_k) + \frac{1}{\phi_{k,t}}} \tag{15}$$

Here, $\frac{1}{\phi_{k,t}}$ represents the expected delay to process 1 unit of data due to the service capacity of node d_k at iteration t and, $\widetilde{T}^{\text{dev}}_{\text{trx}}(d_i, d_k)$ is a proxy of the communication delay in Equation 6 and is calculated as:

$$\widetilde{T}^{\text{dev}}_{\text{trx}}(d_i, d_k) = \frac{1}{q(\langle d_i, d_k \rangle)} \tag{16}$$

Then, the sum in the denominator gives the total delay for a service from node d_k to d_i. Thus, its reciprocal quantifies the effective service rate contribution from d_k to d_i. With this knowledge, the iterative update equation for the aggregated service rate of node d_i is given as:

$$\phi_{i,t+1} = F_{d_k} + \sum_{d_i \in M_{d_k}(t)} \frac{1}{\widetilde{T}^{\text{dev}}_{\text{trx}}(d_i, d_k) + \frac{1}{\phi_{k,t}}} \tag{17}$$

In this equation, $M_{d_k}(t)$ denotes the set of neighbors of node d_i at time t and, F_{d_k} represents the available service rate of node d_k in time t_0, i.e., at the arrival of request r_j, and is calculated as:

$$F_{d_k} = C_{d_k} - \sum_{\substack{s_i \in S \\ r_j \in REQ}} \varphi^{d_k}_{s_i, r_j} \cdot c_{s_i} \tag{18}$$

The iterative process updates ϕ values over time by node's exchanging their current ϕ values periodically. This propagation mechanism ensures that nodes with resource-rich neighbors achieve higher ϕ values, even if their own intrinsic service capacities are limited. Moreover, after each request assignment, F_{d_k} values decrease for nodes that have been utilized. This reduction naturally promotes load balancing by making heavily used nodes less likely to be selected for future assignments.

B. DAG splitting and service assignment

Upon receiving the request r_j on the device $d_k \in D$ at time t, the device initiates the execution of an algorithm to distribute the services, exploiting the computational capabilities of its neighbors (see eq. (17)). The device d_k has to decide which services to host locally and which to offload to its neighbors, using only local information available at time t. The receiving device solves a local optimization problem that minimizes the overall aggregated service rate of the selected devices. Formally, we define the set $\overline{D^t_{d_k}} = M_{d_k}(t) \cup \{d_k\}$, $\overline{D^t_{d_k}} \subseteq D$ that includes the receiving device d_k and its one-hop neighbors available for offloading. The optimization problem for the distributed service assignment is formulated as follows:

$$\min \sum_{d_p \in \overline{D^t_{d_k}}} \left(1/\phi_{p,t} \sum_{s_i \in S_{sr_j}} \varphi^{d_p}_{s_i, r_j} \cdot c_{s_i} \right) \tag{19}$$

$$s.t. \quad \varphi^{d_a}_{s_p, r_j} \cdot \varphi^{d_b}_{s_q, r_j} \leq 1 \qquad \forall \langle d_a, d_b \rangle \in E, \forall \langle s_p \to s_q \rangle \in L_{s_i} \tag{20}$$

$$\varphi^{d_a}_{s_p, r_j} \cdot \varphi^{d_b}_{s_q, r_j} = 0 \qquad \forall \langle d_a, d_b \rangle \notin E, \forall \langle s_p \to s_q \rangle \in L_{s_i} \tag{21}$$

Constraints 10-13 are also included. The goal (19) is the minimization of the consumed aggregated data rate percentage. Constraints (20) and (21) forbid the offloading to different neighbors of sub-DAGs that join later in the service computation graph. For example, service 1 in Fig. 2 service B and

chain C-D, can not be offloaded to different neighbors. This prevents the computation graph from diverging and requiring longer and more complex paths.

A feasible solution of the problem (19)-(21) provides the device d_k with the indication of which services to deploy locally and those to offload to the selected neighbors. Services selected for local deployment are instantiated, consuming the local resources. The remaining services are offloaded to the selected neighbors. The local device sends a deployment request to each selected neighbor containing (i) the sub-DAG induced by the services offloaded that specific neighbor and (ii) the list of devices that have been selected, including the local device d_k, to prevent reuse of the same devices. At the reception of the deployment request, the neighbor device solves the same optimization problem (19)-(21) on the sub-DAG and the subset of neighbors that have not been selected by any previous optimization steps performed by other devices. The distributed process terminates when all the services of the original DAG are deployed and no more offloading requests are sent. To prevent the possibility of reusing a node across multiple hops, we assume that devices send control messages to probe the availability of the neighbors by exploiting some unique request identifier. To handle simultaneous probing messages a simple first-come-first-served policy is adopted.

VI. Evaluation

Our simulations model UAV communication within a specified area using a two-ray ground reflection model. The communication ability between any two nodes d_i and d_j is determined by their signal-to-noise ratio (SNR), and we utilize Shannon's capacity formula to calculate data rates $q(\langle d_i, d_j \rangle)$ [14]. We use the service types as illustrated in Figure 2. We offload one of them randomly to randomly selected UAVs. We offload in total 20 requests and record the average maximum completion time and request delivery ratio at their steady state. These metrics are analyzed for different numbers of workers and varying area sizes, with results presented including 95% confidence intervals.

TABLE I: Simulation Parameters

Parameter	Value
Number of Service Requests	20
Number of UAVs	10-30
Simulation Area	1 - 25 km^2
Computation Capability Mean	50 $GOPS$
Computation Capability Deviation	5 $GOPS$
UAV Altitude	500 m
Transmit Power	30 dBm
Background Noise Power	-95 dBm
Communication Bandwidth	5 MHz
Minimum SNR	3 dB

Table I lists the simulation parameters used. We compare three service deployment methods: *Random Deployment*, *Computation-Oriented Deployment*, and *Distributed Deployment*, which is our proposed method. *Random Deployment* is a distributed algorithm that performs random DAG splitting and random neighbor selection at each step. *Computation-Oriented Deployment* is an alternative version of our proposed deployment algorithm, that aims to select neighbors with larger capacity, by solving the following optimization problem:

$$\max \sum_{d_k \in \overline{D^t_{d_k}}} \sum_{s_i \in S_{s_{r_j}}} \varphi^{d_k}_{s_i, r_j} \cdot C_{d_k} \tag{22}$$

$$s.t.\ (10) - (13), (20) - (21) \tag{23}$$

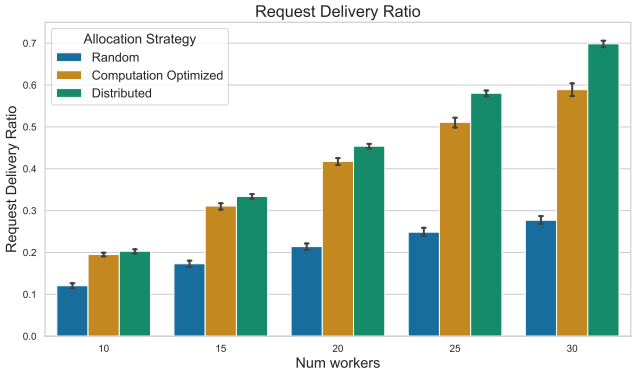
Figure 3a shows the request delivery ratio as a function of the number of workers. The results demonstrate that as the number of workers increases, the request delivery ratio improves across all strategies. However, random allocation consistently underperforms due to its lack of topology-awareness, leading to inefficient task assignments. The computation-optimized approach performs better but tends to overburden high-capacity nodes, creating bottlenecks. In contrast, our proposed distributed deployment strategy leverages the ϕ metric to balance resource utilization, achieving the highest completion rates by avoiding excessive reliance on central nodes.

Figure 3b highlights the impact of worker count on maximum completion times, showing a slight increase in delays as more workers are added. This is caused by the longer communication paths required to utilize additional workers. While random and computation-optimized methods struggle with growing delays, our distributed approach minimizes this effect by dynamically redistributing tasks to balance computational and communication overheads, resulting in consistently lower delays.

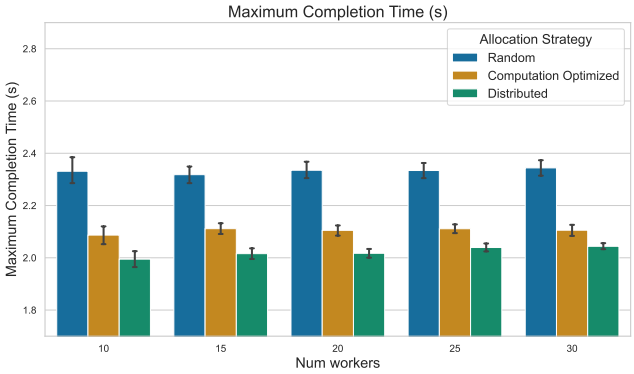
Figures 3c and 3d analyze performance across different simulation area sizes. Expanding area reduces network connectivity, leading to reduced request delivery ratios and higher delays for all methods. Random allocation is most affected, while the computation-optimized approach shows moderate improvement but suffers from uneven task distribution. The proposed method adapts effectively by identifying resource-rich nodes and minimizing long communication paths, maintaining higher completion ratios and significantly lower delays in larger areas. Thus, it achieves slightly better maximum completion time than computation optimization approach while having substantially more request delivery ratio. These results validate the scalability and robustness of the proposed strategy in diverse UAV network scenarios.

VII. Conclusions

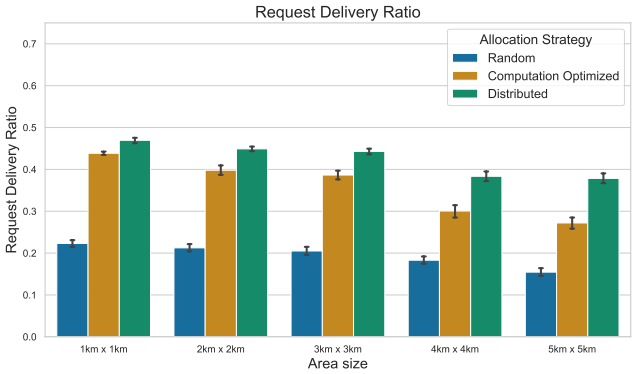
In conclusion, this paper proposes a decentralized approach for task deployment in UAV swarms, effectively addressing scalability and latency issues of centralized methods. By incorporating graph centrality measures and iterative optimization, the method efficiently utilizes resource-rich nodes without overloading them. Simulation results demonstrate superior performance in task deployment and latency reduction compared to existing baselines, highlighting the potential of decentralized algorithms for scalable, dependency-driven applications in UAV networks. Future work will focus on integrating dynamic network conditions, such as varying communication delays and



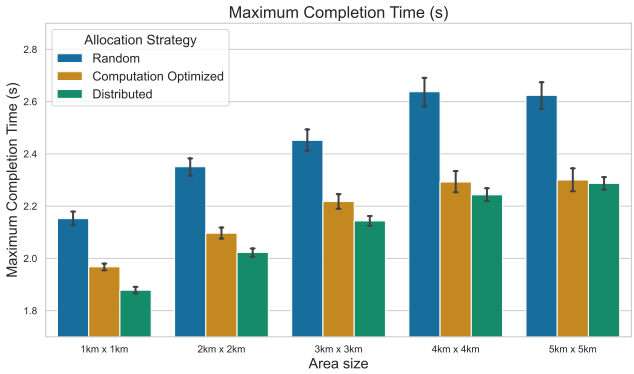
(a) Request delivery ratio as the number of workers increases.



(b) Number of workers against maximum completion time.



(c) Request delivery ratio across different area sizes.



(d) Maximum completion times across varying area sizes.

Fig. 3: Simulation Results.

mobility patterns, to enhance robustness and adaptability. Real-world testing in complex scenarios will further validate its practicality and guide improvements.

Acknowledgment

The work has been supported by the PRIN 2022 project RAIN4C: "Reliable Aerial and satellIte Networks: joint Communication, Computation, Caching for Critical scenarios" (Id: 20227N3SPN, CUP: J53D23007020001) and was partially supported by The Scientific and Technological Research Council of Turkiye (TUBITAK) 1515 Frontier R&D Laboratories Support Program for BTS Advanced AI Hub: BTS Autonomous Networks and Data Innovation Lab. Project 5239903

References

[1] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. M. Assi, "Dynamic task offloading and scheduling for low-latency iot services in multi-access edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, pp. 668–682, 2019.

[2] D. Popescu, F. Stoican, G. Stamatescu, O. Chenaru, and L. Ichim, "A survey of collaborative UAV–WSN systems for efficient monitoring," *Sensors (Basel, Switzerland)*, vol. 19, 2019.

[3] C.-F. Liu, M. Bennis, M. Debbah, and H. V. Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Transactions on Communications*, vol. 67, pp. 4132–4150, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:56475879

[4] A. Trotta, A. Heideker, I. Zyrianoff, G. Interdonato, and S. Pizzi, "Advancing Non-Terrestrial Networks for Critical Scenarios with the RAIN4C Framework," in *2024 20th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, 2024, pp. 780–782.

[5] X. Wei, L. Cai, N. Wei, P. Zou, J. Zhang, and S. Subramaniam, "Joint UAV trajectory planning, DAG task scheduling, and service function deployment based on drl in UAV-empowered edge computing," *IEEE Internet of Things Journal*, vol. 10, pp. 12 826–12 838, 2023.

[6] T. T. Sarı, S. Ahmad, A. Aral, and G. Seçinti, "Collaborative smart environmental monitoring using flying edge intelligence," in *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, 2023, pp. 5336–5341.

[7] Z. Zhao, L. Pacheco, H. Santos, M. W. T. Liu, A. D. Maio, D. do Rosário, E. Cerqueira, T. Braun, and X. Cao, "Predictive UAV base station deployment and service offloading with distributed edge learning," *IEEE Transactions on Network and Service Management*, vol. 18, pp. 3955–3972, 2021.

[8] J. Tian, D. Wang, H. Zhang, and D. Wu, "Service satisfaction-oriented task offloading and UAV scheduling in UAV-enabled mec networks," *IEEE Transactions on Wireless Communications*, vol. 22, pp. 8949–8964, 2023.

[9] Q. Wang, B. Gao, Z. Zhou, F. Xu, and C. Ouyang, "Dag-aware optimization for geo-distributed data analytics," *Proceedings of the 52nd International Conference on Parallel Processing*, 2023.

[10] J. Li, Y. Pan, Y. Xia, Z. Fan, X. Wang, and J. Lv, "Optimizing dag scheduling and deployment for iot data analysis services in the multi-UAV mobile edge computing system," *Wireless Networks*, pp. 1–15, 2023.

[11] C. Quadri, A. Ceselli, and G. P. Rossi, "Multi-user edge service orchestration based on deep reinforcement learning," *Computer Communications*, vol. 203, pp. 30–47, 2023.

[12] L. Montecchiari, A. Trotta, L. Gigli, L. Bellone, M. Di Felice, and E. Natalizio, "Distributed service discovery over heterogeneous robotic systems-of-systems," in *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, 2023, pp. 227–231.

[13] T. T. Sarı and G. Secinti, "Using centrality based topology control for FANET survivability against jamming," *Computer Networks*, vol. 242, p. 110250, 2024.

[14] C. E. Shannon, "A mathematical theory of communication." *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.