EdgeDrones: Co-Scheduling of Drones for Multi-Location Aerial Computing Missions

Uchechukwu Awada[a], Jiankang Zhang[b], Sheng Chen[c], Shuangzhi Li[a] and Shouyi Yang[a]

[a]School of Information Engineering, Zhengzhou University, Zhengzhou, 450001, China
[b]Department of Computing and Informatics, Bournemouth University, Poole, BH12 5BB, UK
[c]School of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, UK

ARTICLE INFO

Keywords:
Edge computing
Aerial computing
Vehicle routing
Linear regression
Execution time
Resource efficiency
Co-location

ABSTRACT

Low altitude platform (LAP) unmanned aerial vehicles (UAVs), also called *drones*, are currently being exploited by Edge computing (EC) systems to execute complex resource-hungry use cases, such as virtual reality, smart cities, autonomous vehicles, etc., by attaching portable edge devices on them. However, a typical drone has limited flight time, coupled with the resource-constrained attached edge device, which can jeopardize aerial computing missions if they are not holistically taking into consideration. Moreover, the fundamental challenge is how to co-schedule multi-drone among multi-location where EC services are needed, such that drones are scheduled to maximize the utility from the activities while meeting computing resource and flight time constraints. Therefore, for a given fleet of drones and tasks across disjointed target locations in a city, we derive a machine learning (ML) linear regression model that estimates these tasks resource requirement and excution time. Leveraging this estimation values, we jointly consider each drone's flight time availability and its attached edge device resource capacity, and formulate a novel Multi-Location Capacitated Mission Scheduling Problem (MLCMSP) that selects suitable drones and co-schedules their flight routes with the least total distance to visit and execute tasks at the target locations. Then, we show that faster scheduling and execution of complex tasks at each location, while considering the inter-task dependencies is important to achieve effective solution for our MLCMSP. Hence, we further propose *EdgeDrones*, a variant bin-packing optimization approach through *gang-scheduling* of inter-dependent tasks that *co-schedules* and *co-locates* tasks tightly so as to achieve faster execution time, as well as to fully utilize available resources. Extensive experiments on Alibaba cluster trace with information on task dependencies (about 12,207,703 dependencies) show that *EdgeDrones* achieves up to 73% higher resource utilization, up to 17.6 times faster executions, and up to 2.87 times faster flight travel time compared to the baseline approaches.

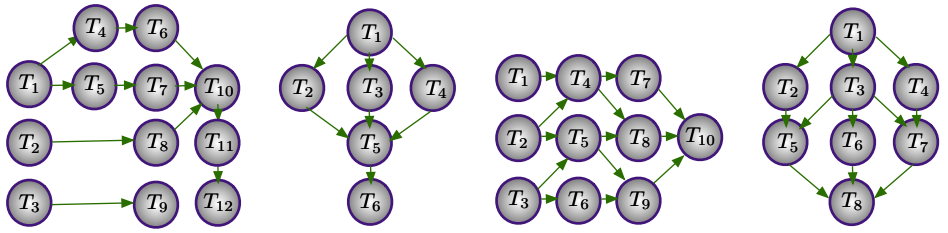
1. Introduction

Edge computing (EC) is a distributed computing model which places cloud computing [1, 2] services closer to data sources so as to achieve faster response times and real-time insights. Many latency-sensitive applications that process data from IoT devices and sensors, rely on heterogeneous edge resources in close proximity for faster response times and to promote rapid development. To this end, several EC devices of various sizes and capacities have emerged. For example, AWS Snowcone[1], Azure Stack Edge mini[2], etc., are portable EC devices that weighs about 2 $\sim$ 3 kg but are inherently resource-constrained relative to their on-premise counterparts, i.e., AWS Snowball[3], etc. Nonetheless, EC systems are currently exploiting attaching these portable edge devices on low altitude platform (LAP) unmanned aerial vehicles (UAVs) or drones to execute complex resource-hungry use cases, such as cyber-physical systems [3], virtual reality [4], smart vehicles [5], face recognition [6], smart cities [7], etc. In addition, an autonomous drone technology called *Drone-in-a-box*[4], is currently being exploited for aerial EC missions [5, 8, 9]. A drone-in-a-box system can be deployed autonomously from a box that serves as a landing pad and charging base (i.e., *depot*) to perform on-demand computation activities in a city. An activity involves visiting points of interest (i.e., *target locations*), hovering and interacting with end devices to execute tasks on its attached edge device(s). After completing the tasks, the results are immediately and deterministically communicated back to the end device, then it returns to its box or depot. However, a typical drone has a limited flight time due to power factor which can jeopardize the entire mission if it is not taking into consideration [10, 11, 12]. Hence, the critical issue is how to assign missions and optimal routes for multiple drones to visit a set of locations, so that they can complete their tasks, subject to the flight time and attached edge resource constraints of each drone, without jeopardizing application performance.

Therefore, for a given fleet of drones and multi-task across disjointed locations in a city, we propose a novel Multi-Location Capacitated Mission Scheduling Problem (MLCMSP) that co-schedules their optimal flight routing among the locations, such that the drones can visit the locations and complete the tasks, within their flight times and computing resource constraints, while maximizing the

*Corresponding author
jzhang3@bournemouth.ac.uk (J. Zhang); ielsz@zzu.edu.cn (S. Li)
ORCID(s):

[1]https://aws.amazon.com/snowcone/
[2]https://azure.microsoft.com/en-us/products/azure-stack/edge/#overview
[3]https://aws.amazon.com/snowball/
[4]https://en.wikipedia.org/wiki/Drone_in_a_Box



(a) Video classification application (b) Face recognition application (c) Image classification application (d) Crowd counting application

Figure 1: DAG of representative applications.

total utilization. Specifically, our MLCMSP combines elements of Vehicle Routing Problem (VRP), which is a variant of the well known Traveling Salesman Problem (TSP) to find optimal routes for a set of vehicles and customers [13, 14]. Existing works have proposed routing strategies for UAV-enabled task offloading in edge-cloud computing systems [15, 16]. However, they have either focused on co-scheduling a fleet of drones or task offloading to meet some specific objectives. It is important to note that a drone routing strategy for EC which do not consider the drones' flight time and resource capacity or charateristics of target tasks, such as dependencies, resource requirement, etc., can lead to loss of job or an incomplete mission [10, 11, 12]. Effective co-scheduling of a given fleet of drones for EC missions across multi-location, requires jointly optimization of the following; (*i*) update information of each drones' flight time availability and its attached edge device(s)[5] resource capacity or availability, (*ii*) locations of end devices requesting EC services interms of flight distance, flight travel time, etc., and (*iii*) their tasks resource requirement and execution time estimations, so as to select drones which can maximize the utility from the activities. A disjointed approach which interacts individually with each drone, would exhibit high computation complexity and is far from trivial to realize [11, 12]. For this reason, we wish to consider an approach which seamlessly integrate all end devices, service entities and edge resources running across multiple drones in a single pool, such that these information can be holistically obtained and monitored from a single control plane (CP), where it can be used for decision making on efficient mission planning and assignment. This approach is called *Edge Federation* (EF) [17, 18]. For example, recently introduced edge computing frameworks, i.e., KubeEdge, MicroK8s, etc, have the capabilities of integrating service entities and edge resources running across multiple drones, run containerized tasks and eliminate provider lock-in situations. EF can enable effective co-scheduling of multiple drones, by selecting a minimum number of drones which can maximize the utility for any given activities. Hence, a drone can be assigned multiple disjointed locations as part of its mission.

[5]A typical drone-based edge deployment can attach one or more or different combination of portable edge devices, depending on the drones' load capacity.

An important challenge is developing an efficient scheduling strategy that can place and execute complex applications on the attached edge devices in a timely manner, while efficiently managing available resources as drones visit their assigned locations. For example, modern applications (i.e., face recognition [6], image classification [19], crowd counting [20], etc.,), as shown in Fig (1) are becoming more complex in nature, structured on microservices architectural style, consisting of a large number of inter-dependent applications and often latency-sensitive [21, 22, 23]. It is naturally important to intelligently schedule such inter-dependent applications in a best possible way, such that they are quickly executed and immediately sent back to the IoT and end devices. Existing scheduling approaches which do not consider such task dependencies, co-location strategy or which randomly deploy tasks to any available resources can easily result in delay, fragmentation and over-allocation of resources, hence jeopardizing the application performance, given the drones' flight time and resource constraints. To address this challenge, we first estimate tasks resource requirement and execution time at target locations, using linear regression machine learning (ML) model. These estimation values, as well as the drones' flight time, flight distance to target locations, and attached edge resource availability are used as inputs to plan missions for a captive set of drones to accomplish any given activities. One drawback of this concept is that inaccurate estimations of tasks resource requirement and execution time at target locations, could also jeopardize the entire missions for selected drones. Similarly, if the tasks are scheduled naively, e.g., in an edge deployment which can only execute one task or job at any time and where each task is scheduled individually [24, 25], the system might not yield an optimal performance. Therefore, we first investigate the accuracy of our trained linear regression ML model for estimating multi-task resource requirement and execution time, using the normalized absolute estimate error (NAEE) method. This serves as the estimation accuracy measure for the trained linear regression ML model. We further propose **EdgeDrones**, which extends the state-of-the-arts by providing an intelligent dependency-aware multi-task scheduling and co-location scheme to achieve high resource utilization and faster execution of tasks.

In particular, we show that EF [17, 18] and ML techniques [26, 27] can help aerial edge systems to achieve effective co-scheduling and optimal route planning for a fleet of autonomous drones to accomplish stochastic service requests from end devices across target locations in a city. With limited edge resources and drones' flight time, it is necessary to consider task dependencies in drone-based EC task offloading, by jointly optimizing the drones' flight time and resource availability, such that all the tasks can be intelligently scheduled and fast executed with minimum resources before the drone returns for recharging or departs for another assigned location. Hence, our aim is to plan optimal flight routes for drones to execute all the tasks by considering dependencies and resource demands, such that the **actual** scheduling and execution time is minimized, and is much less than the drones' flight time. In summary, to achieve our EdgeDrones implementation, we address the following critical areas:

- We propose an integrated system with global information, through the joint optimization of all service entities, and formally define the novel Multi-Location Capacitated Mission Scheduling Problem (MLCMSP) that selects and co-schedules optimal flight routes for a fleet of drones for any given aerial computing missions in a city.
- Specifically, we derive a multi-task machine learning (ML) execution time and resource requirement estimation, to aid MLCMSP to select drones with requisite resource availability which can maximize the utility from the aerial activities.
- To guarantee optimal usage of cluster resources and faster execution of tasks, we further propose a variant bin-packing optimization approach through gang-scheduling of multi-dependent tasks, which co-schedules and co-locate tasks firmly on available nodes, so as to avoid resource wastage.
- We show that EdgeDrones is capable of minimizing the actual completion time of multi-dependent tasks using minimum resources, and we conduct extensive experiments to compare the performance of our EdgeDrones with several existing approaches using real-world data-trace from Alibaba cluster trace[6], which provides information on task dependencies.

The rest of the paper is organized as follows. In Section 2, we discuss the related work. In Section 3, we present some preliminaries on task dependency-awareness and discuss our motivation. In Secton 4, we detail our proposed EdgeDrones for achieving high resource utilization and minimizing the execution times of applications deployed on federated aerial edge resources. In Section 5, we compare the performance of our proposed EdgeDrones against that of several state-of-the-art approaches through extensive experiments. Finally, we conclude the paper in Section 6.

[6]https://github.com/alibaba/clusterdata

2. Related Works

UAV or drone-based edge computing deployments are gaining increasingly popularity due to their autonomous navigation, low cost, mobility, flexibility and adaptive altitude to deliver faster execution closer to data sources. They are currently being exploited for several use cases, i.e., task offloading, data caching, data streaming, etc. For example, the authors of [28] proposed a UAV-enabled edge network to minimize the system-wide computation cost by efficient task offloading and deployment. They have formulated and solved this problem as a stochastic game. The work [3] proposed a joint trajectory, task offloading and caching optimization in a UAV-enabled edge for Cyber-Physical System. They have proposed this to realize energy-efficient performance of the UAV. In [4], the authors explored the UAV-assisted edge and streaming for virtual reality. They formulated this problem as a joint joint UAV placement, edge resource allocation, and 360-degree video content layer assignment. They aim to select the allocation of computing and communications resources, such that the delivered quality of experience (QoE) is maximized.

UAV or drone-enabled multi-task offloading schemes in edge systems can benefit from joint optimization of drones' flight time, resource availability status, multi-task's resource requirement and execution time, such that drone with sufficient resource availability can be deloyed to conduct efficient execution request. A-priori information about task execution time is mostly important for drone-based edge deployments [11, 12]. This is because a typical drone has limited flight time, and could possibly lead to a delayed task execution if it is not taken into consideration [10]. In particular, optimal route can be planned and assinged to ensure that drones can complete their tasks, given their flight time constriants. There exist works that explore routing and trajectory scheduling for drone-enable edge systems. For example, the work [15] proposed a mssion scheduling problem (MSP) that co-schedules the flight routes of drones to visit target locations and record videos. Similarly, the work [16] proposed an online algorithm for UAV swarms to jointly optimize the task offloading and multi-hop routing scheduling. In [3], the authors proposed a joint optimization of drone's 3D trajectory scheduling and the task-cache strategies to minimize its total energy consumption. The work [29] formulated an optimization problem to minimize the total energy consumption of a UAV through joint partitioning and UAV trajectory scheduling. In [30], the authors presented a distributed task offloading and path planning algorithm to provide computational support to large-scale IoT nodes. However, these schemes do not consider drones' flight time and assume a drone can fly for unlimited amount of time, which can lead to delay or loss of job due to drones' limited flight time [10, 11, 12].

For multi-task at target locations, an accurate execution time and resource demand estimation is mostly needed to schedule routes for drone(s) and a-priori to conduct efficient multi-task scheduling. Consequently, existing researches

have proposed a huge number of learning methods to estimate task's resource requirements and execution time, based on collaborative learning (CL) [11, 18, 31], machine learning (ML) [12, 26, 27, 32], incremental learning (IL) [33], scheduling [34, 35, 36, 37] and statistical models [38]. Our previous works [11, 12] focused on of multi-dependent tasks orchestration in autonomous drone-enabled edge computing system, while considering the drones' flight time, so as to avoid loss of jobs [10]. Specifically, in [11], we have proposed a multi-output linear regression model based on CL to estimate multi-dependent task's resource requirement and execution time, to select the closest drone deployment having matching resource availability and flight time to execute ready tasks at a given time. In [12], we have proposed a ML based multi-dependent tasks dispatching over a federated autonomous drone-enabled edge computing platform, using the total estimated value of the multi-dependent tasks' execution time to select a suitable drone.

With limited edge resources, is it also important to avoid any form of resource wastage, i.e., resource under utilization. Efficiently managing edge resources directly dictates service quality and performance [39]. As a result, task co-location has gained attention both in academia and industry as an optimal solution for improving resource utilization and system throughput in distributed systems. However, effective task co-location is a non-trivial task, as it requires an understanding of the computing resource requirement of the co-running tasks, in order to determine how many of them can be co-located. To this end, a tasks co-location mechanism was proposed in [40], where it was showed that by accurately estimating the resource level needed, a co-location scheme can effectively determine how many tasks can be co-located on the same host to improve the system throughput, by taking into consideration the memory and CPU requirements of co-running tasks. With the aim to maximize the resource utilization, the authors of [41] utilized reinforcement learning to co-locate interactive services with batched ML workloads. Our previous works [42, 43] focused on workload co-location in cloud environment, rather than edge systems. To further improve edge resource management, a resource management scheme was proposed in [17, 18] which unifies distributed edge resources, such that they are holistically managed. Our previous work [17] proposed a dependency-aware task scheduling in such unified system. Modern applications are usually structured with inter-task dependencies, whereby a task depends on an input from other task(s). A huge number of existing works, i.e., [21, 22, 23, 44] have tackled the problem scheduling such inter-dependent tasks or multi-dependent tasks, and their common goal is to formulate a scheduling decision that minimizes the average completion time of such tasks.

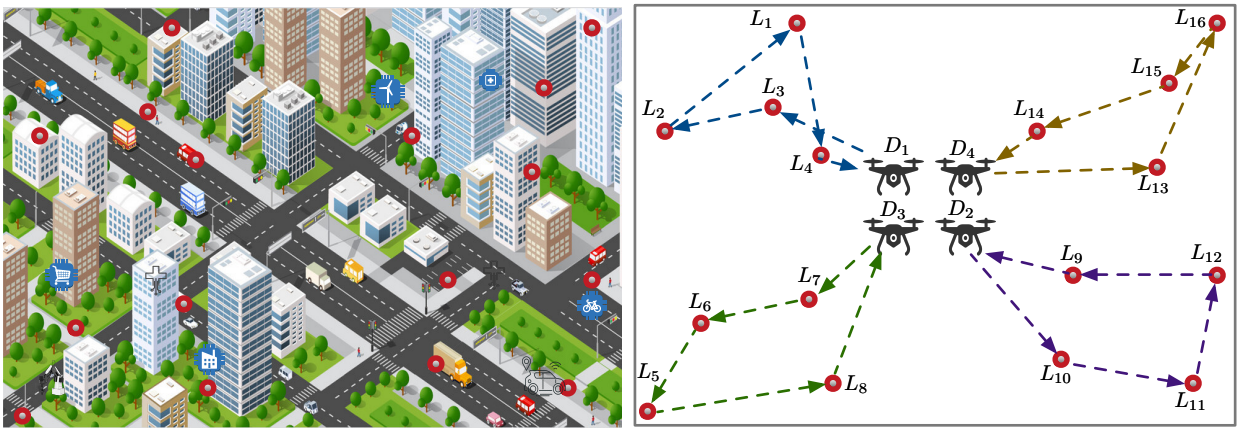
Existing works on UAV-enabled approaches for task offloading and execution in multi-edge deployments do not jointly consider tasks dependencies, do not unify service entities and distributed edge resources, such that they are holistically managed and monitored from a single control plan (CP), where such information can be utilized to co-schedules multi-drone, co-locate multi-task effectively . This motivates our research to extend existing schemes by proposing EdgeDrones. Specifically, we propose a learning-based multi-drone route scheduling through a unified system, which include all service entities and resources running across the multi-drone, location of end devices and their applications. We further propose a variant bin-packing optimization approach through gang-scheduling of multi-dependent tasks, which quickly co-schedules and co-locate tasks firmly on available nodes, so as to avoid delay and resource wastage. We finally show that EdgeDrones is capable of minimizing the actual completion time of multi-dependent tasks using minimum resources through extensive experiments and comparison.

3. A Case Study on a Smart City

We consider a smart city senario, where multiple IoT and other end devices are deployed across the city to improve life standards of its citizens. For example, Toyota Motor Corporation has recently embarked on a new smart city project called Woven City[7], where new technologies such as smart construction and manufacturing, smart homes, robotics, connectivity through AI, autonomy, etc are being deployed. EC provides a promising way of enabling these technologies by offering computing resources with low latency. Smart city solutions are increasingly integrating UAVs or drone-enabled EC for enhanced performance [7]. Suppose at time t, there are updates from devices at multiple locations in the city, as shown in Fig. 2(a), drones equipped with EC devices can fly to these locations to render needed services in a timely manner. However, how to select, assign and route flight paths for drones to accomplish these tasks effectively is a major challenge. To address this, each task's resource requirement in terms of CPU and memory at each location L_i is estimated, i.e., $\langle \widetilde{c_i}, \widetilde{m_i} \rangle$, as well as its execution time $\widetilde{E}_{ex_i}$. These values, as well as the location coordinates are utilized to select suitable drones (i.e., drones with sufficient flight time f_i and resource capacity $\langle c, m \rangle$), such that their optimal routes to visit the locations are scheduled, as shown in Fig. 2(b). Nevertheless, a routing problem with many locations can take a long time to solve. Therefore, for such problems, it is better to set a search limit which terminates the search after a specific length of time or number of solutions is returned.

Most importantly, as the drones embark on their missions, efficient scheduling strategy for complex applications at each location L_i is absolutely necessary to guarantee high application performance and successful completion of each drone's mission. For example, in Fig. 2(b), `Drone 1` is assigned 4 locations (L_1, L_2, L_3 and L_4). Suppose at these locations, we have complex applications in the form of directed ayclic graphs (DAGs), as shown in Figs. 2(a), (b), (c) and (d), respectively, where each job typically consists of several tasks whose dependencies are expressed by DAG, i.e., the job in Fig. 1(a) consists of inter-task dependency depth γ of

[7] https://www.woven-city.global/



(a) Locations of end devices needing EC services in a smart city (b) Drones assignement, their optimal routes and flight paths

Figure 2: (a)Locations of end devices needing EC services, (b) drones assignement together with their optimal routes.

12, i.e., $(T_1, T_2, \ldots, T_{12})$. Such complex inter-task dependencies with multi-dimensional resource demands, i.e., various amounts of CPU and memory resources, and communication requirements, make resource management in such a drone-enabled EC system very challenging. Knowledge about these tasks characteristics, i.e., resource demands and dependencies, is necessary to pack or co-locate them effectively in available resources, ultimately to minimize their collective response times and improve the resource utilization [11, 12]. Clearly, tasks T_1, T_2 and T_3 are independent task, i.e., no dependency, and they can be started without waiting for any other task(s), while tasks T_4 and T_5 depend on the completion of task T_1. Similarly, task T_{10} depends on the completion of tasks T_6, T_7 and T_8. Hence a key objective of our **EdgeDrones** is to reduce the collective execution time of such tasks and improve resource utilization by considering the inter-task dependency and resource demands. Therefore, given the n multi-dependent tasks $T_1, T_2, \cdots, T_n$, as shown in Fig. 2, EdgeDrones adopts the gang-scheduling [11, 12, 45] strategy and a variant bin-packing optimization to efficiently co-schedule and co-locate them in the attached edge nodes. We consider EdgeDrones as a Full Dependency and Full Packing (FDFP) scheduling approach. Thus, the scheduling time can be expressed as:

$$\sum_{z=1}^{m}\sum_{i=1}^{k_z} S_{ch_{z_i}}/k_z, \tag{1}$$

where m is the number of scheduling units, k_z is the number of tasks within the z-th scheduling unit having the tasks $\{T_{z_1}, T_{z_2}, \cdots, T_{z_{k_z}}\}$.

We illustrate the advantage of the scheduling approach in EdgeDrones over three other existing schemes as follows; (i) a scheduling approach which does not consider tasks' dependencies, but schedules 50% of any given multi-dependent tasks by mainly focusing on task co-location, we refer to this scheduling approach as No Dependency and Full Packing (NDFP), and it is similar to the approach in [46]; (ii) a scheduling approach which schedules up to 15% of any given multi-dependent tasks at a time, but does not consider task co-location, we refer to this scheduling approach as Partial Dependency and No Packing (PDNP), and it is similar to to the approach in [47]; (iii) a scheduling approach which schedules up to 40% of any given multi-dependent tasks with task co-location, we consider this scheduling approach as a Partial Dependency and Full Packing (PDFP), and it is similar to to the approach in [48]; and (iv) the Random approach, which does not consider both tasks' dependencies and task co-location, we refer to this scheduling approach as No Dependency and No Packing (NDNP). It is important to note that delay in scheduling inter-dependent tasks directly impacts their collective execution time. For the multi-dependent tasks of Fig. 1 with $n = 12$ tasks, Table 1 lists the scheduling orders and scheduling units for the schemes compared. EdgeDrones only needs one scheduling unit ($m = 1$) which has $k_1 = 12$ tasks, and it also achieves the lowest execution time of $\frac{1}{12}\sum_{i=1}^{12} E_{ex_i}$. By contrast, Random has $m = 12$ scheduling units, each having a single task. Hence it has the highest execution time of $\sum_{i=1}^{12} E_{ex_i}$. Thus, EdgeDrones achieves the lowest scheduling and execution time. PDNP, PDFP and NDFP deploy individual or subsets of the tasks at a time. Generally, delay in scheduling dependent tasks directly impacts job completion time, and utilizing gang scheduling is beneficial for overall performance.

4. System Model, Problem Formulation and Algorithm Framework

In this section, we detail our proposed EdgeDrones for achieving optimal routing scheduling for drones aerial missions, high resource utilization and minimizing the execution times of applications deployed on an integrated edge computing system. The system model is shown in Fig. 3.

Table 1
Scheduling orders and units of various schemes

Scheme	Scheduling Order	Scheduling Units
EdgeDrones	$\{T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}, T_{11}, T_{12}\}$	1
PDNP	$T_3 \rightarrow T_2 \rightarrow \{T_1, T_4\} \rightarrow \{T_6, T_8\} \rightarrow \{T_5, T_7\} \rightarrow \{T_{10}, T_{11}\} \rightarrow \{T_{12}, T_9\}$	7
PDFP	$T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow \{T_4, T_5\} \rightarrow \{T_6, T_7, T_8, T_9\} \rightarrow \{T_{10}, T_{11}, T_{12}\}$	6
NDFP	$\{T_1, T_2, T_3, T_4, T_5, T_6\} \rightarrow \{T_7, T_8, T_9, T_{10}, T_{11}, T_{12}\}$	2
Random	$T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4 \rightarrow T_5 \rightarrow T_6 \rightarrow T_7 \rightarrow T_8 \rightarrow T_9 \rightarrow T_{10} \rightarrow T_{11} \rightarrow T_{12}$	12

4.1. System Model

At the city center, we have a depot L_0 with coordinate $\{x_0, y_0\}$, where a fleet of drones $\mathbb{DR} = \{\mathcal{DR}_1, \cdots, \mathcal{DR}_N\}$ are stationed and ready for aerial computing missions. Each drone has flight time availability f_i^{aval} and resource capacity $\mathcal{RC}_i^{\langle c,m\rangle}$, in terms of CPU and memory resources. Then, at time $t > 0$, we have a set of end devices $\mathbb{D} = \{\mathcal{D}_1, \cdots, \mathcal{D}_M\}$ requesting EC services across multi-location $\mathbb{L} = \{L_1, \cdots, L_M\}$ with their coordinates $L_1 = \{x_1, y_1\}, \cdots, L_M = \{x_M, y_M\}$ within the city. The edge federated system $\mathbb{EF}$ consists of all service entities in the N drones $\mathcal{DR}_i$, $1 \leq i \leq N$, and M end devices $\mathcal{D}_i$, $1 \leq i \leq M$, i.e.,

$$\mathbb{EF} = \mathbb{DR} \bigcup \mathbb{D}. \tag{2}$$

A set of multi-task $\mathbb{T} = \{T_1, \cdots, T_N\}$ from the end device(s) at each location $L_i \in \mathbb{L}$ require an amount of CPU and memory resources $\mathcal{RR}_i^{\langle c,m\rangle}$ for execution. These resource requirements along with execution times are first estimated using linear regression ML model. The multi-task features $\boldsymbol{f}_{\text{mt}}(\omega, \epsilon, \gamma)$, where ω is the number of instances, ϵ is type of tasks , γ is dependency depth, are fed into the model $\Theta^\star$ to estimate the values of the resource requirement and execution times according to

$$\boldsymbol{f}_{\text{mt}} \cdot \Theta^\star = \left[\widetilde{E}_{ex_1} \widetilde{T}_1^{\langle c,m\rangle} \; \widetilde{E}_{ex_2} \widetilde{T}_2^{\langle c,m\rangle} \cdots \widetilde{E}_{ex_N} \widetilde{T}_N^{\langle c,m\rangle}\right], \tag{3}$$

where $\widetilde{T}_i^{\langle c,m\rangle}$ and $\widetilde{E}_{ex_i}$ are the estimated resource requirement (in terms of CPU and memory $\langle c, m\rangle$) and estimated execution time for task i, respectively. We show that with these estimated values, suitable drones can be assigned and multi-dependent tasks can be intelligently scheduled with the aim of minimizing their actual completion time, while maximizing available resources. Assuming that $\boldsymbol{f}_{\text{mt}} \in \mathbb{R}^{1\times d}$ is a d-dimensional vector (tensor), then Θ is a $(d \times \epsilon)$-dimensional parameter matrix. To build this predictor Θ, we train it using historical data from previously executed tasks/jobs based on Keras[8]. Keras is a library which wraps TensorFlow[9] complexity into simple and user-friendly API. The dataset $\mathcal{DS} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$ contain d-dimensional tensors of data features $\boldsymbol{x}_i \in \mathbb{R}^{1\times d}$ and ϵ-dimensional tensors of labels (actual execution times) $\boldsymbol{y}_i \in \mathbb{R}^{1\times\epsilon}$. The learning problem is to solve the following optimization:

$$\Theta^\star = \arg\min_{\Theta\in\mathbb{R}^{d\times\epsilon}} \frac{1}{2n}\sum_{i=1}^{n} \|\boldsymbol{x}_i\Theta - \boldsymbol{y}_i\|_2^2 + \frac{\lambda}{2}\|\Theta\|_F^2, \tag{4}$$

where λ is the regularization parameter and $\|\cdot\|_F$ denotes the Frobenius norm. The optimization (4) is solved using gradient-descent, where the model is updated iteratively until convergence, i.e., $\Theta^{t+1} = \Theta^t - \eta\left(\frac{1}{n}\boldsymbol{g}(\Theta^t) + \lambda\Theta^t\right)$, in which η is the learning rate, $\boldsymbol{g}(\Theta) = \frac{1}{n}\boldsymbol{X}^{\text{T}}\left(\boldsymbol{X}\Theta - \boldsymbol{Y}\right)$ denotes the gradient of the loss function, $\boldsymbol{X} = \left[\boldsymbol{x}_1^{\text{T}} \cdots \boldsymbol{x}_n^{\text{T}}\right]^{\text{T}}$ and $\boldsymbol{Y} = \left[\boldsymbol{y}_1^{\text{T}} \cdots \boldsymbol{y}_n^{\text{T}}\right]^{\text{T}}$ are the feature set and label set, respectively. These estimation values $\widetilde{T}^{\langle c,m\rangle}$ and $\widetilde{E}_{ex}$ are important information for selecting suitable drones from the depot for the missions. The target location coordinates $L_i = \{x_i, y_i\}$ and depot coordinates $L_0 = \{x_0, y_0\}$ are used to compute the distance matrix to also aid with the selection of suitable drones. The distance matrix is an array of distances between these locations. These distances can be obtained using the Manhattan Distance[10], in which the distance between two locations $L_i = \{x_i, y_i\}$ and $L_j = \{x_j, y_j\}$ is given as:

$$d^{L_{i,j}} = |x_i - x_j| + |y_i - y_j|. \tag{5}$$

These distances can also be obtained using Google Distance Matrix API[11]. Hence, the distance matrix is given as;

$$\mathbb{DM} = \begin{bmatrix} d^{L_{0,0}} & d^{L_{0,1}} & d^{L_{0,2}} & \dots & d^{L_{0,n}} \\ d^{L_{1,0}} & d^{L_{1,1}} & d^{L_{1,2}} & \dots & d^{L_{1,n}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d^{L_{m,0}} & d^{L_{m,1}} & d^{L_{m,2}} & \dots & d^{L_{m,n}} \end{bmatrix}. \tag{6}$$

The number of rows of the $\mathbb{DM}$ indicates the number of target locations and the depot inclusive. Autonomous drone

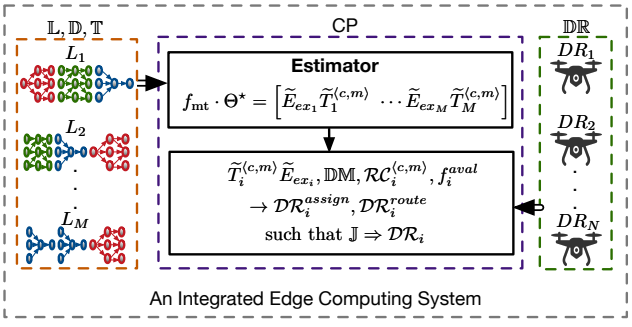


Figure 3: Orchestration overview of EdgeDrones.

[8] https://keras.io/
[9] https://www.tensorflow.org/
[10] https://en.wikipedia.org/wiki/Taxicab_geometry
[11] https://cloud.google.com/blog/products/maps-platform/how-use-distance-matrix-api

systems have tools to help them estimate flight travel time for distances between locations [11, 12]. A drone's power has a capacity P, which is used for the flights (i.e., traveling and hovering) at a constant energy-efficient speed s. Note that the attached edge device(s) does not depend on the drone's power, instead, it is powered by its on-board power supply. For example, AWS Snowcone can be powered by any standard USB-C power bank and can deliver up to 14 trillion operations per seconds (TOPs) with as little as 10W of power. Therefore, the flight travel time for $d^{L_{i,j}}$ is given as $f^{L_{i,j}}$, and the flight hovering time at L_i is given as f^{hover}. As the actual execution time is unknown at this point, we assume that the flight hovering time is equivalent to the multi-task execution time estimation, i.e., $f^{hover} \approx \sum_{i=1}^{N} \widetilde{E}_{ex_i}$. Suppose that a drone travels from L_i to L_j in one step of its route, and that:

- The drone's cumulative flight travel time upon arrival at L_i, given as f^{L_i} is $60s$.
- The drone's cumulative flight travel time upon arrival at L_j, given as f^{L_j} is $130s$.
- The drone's flight travel time $f^{L_{i,j}}$ is $50s$.

Obviously, the drone can not depart location L_i immediately upon arrival, otherwise its cumulative flight travel time f^{L_j} upon arrival at L_j would be $110s$. Instead, the drone must hover and execute the tasks at L_i for $20s$ before departing for L_j. In other words, the execution time also constitues a drone's flight travel time. On the other hand, note that the a drone's resource capacity $\mathcal{RC}^{\langle c,m\rangle}$ does not accumulate as it travels along its route. This is because, after executing its tasks at L_i, the results are immediately and deterministically communicated back to the end device at L_i, and its resources becomes available for its next task execution at location L_j. Also, a drone can be assigned multiple disjointed locations as part of its mission, giving that it has sufficient resource availability, i.e., compute resources and flight time. Hence, a drone's total distance and flight travel time for its entire mission is given as;

$$d^{total} = \sum_{i=0}^{n} \sum_{j=i+1}^{m+1} d^{L_{i,j}} \tag{7}$$

and

$$f^{total} = \sum_{i=0}^{n} \sum_{j=i+1}^{m+1} f^{L_{i,j}} + f_j^{hover}, \tag{8}$$

respectively. Since, a drone can be assigned multiple disjointed m locations, it must start and end its missions at the depot L_0. For uniformity, we denote the starting and the ending depot location in its route as L_0 and L_{m+1}, respectively. Therefore, given a federated system $\mathbb{EF}$ consisting of drone-enabled EC deployments $\mathbb{DR}$, where each participating drone $\mathcal{DR}_i$ is attached with container-optimized nodes, and a set end devices $\mathbb{D}$ requesting EC services across multiple locations in a city, an update state information from the CP which include each drones' fight time availability f_i^{aval}, its total resource capacity $\mathcal{RC}_i^{\langle c,m\rangle}$, each end device $\mathcal{D}_i$ inter-dependent tasks execution and resource demand estimation $\widetilde{E}_{ex_i}\widetilde{T}_i^{\langle c,m\rangle}$, location coordinates $L_i = \{x_i, y_i\}$ and distance matrix $\mathbb{DM}$, is needed to select, assign and schedule optimal routes $\mathcal{DR}_i^{route}$ for drones to visit these locations and execute the tasks, such that

$$\mathcal{DR}_i^{route} = \arg\min_{\mathcal{DR}_i \in \mathbb{DR}} \left\{ d_i^{total} : f_i^{total} < f_i^{aval}, \mathcal{RC}_i^{\langle c,m\rangle} \text{ sufficient} \right\}. \tag{9}$$

A drone must execute a set of inter-dependent tasks $\mathbb{T}$ at each of its assinged location before it returns to its depot. *EdgeDrones* utilizes the gang scheduling [45] strategy to co-schedule all tasks $T_i \in \mathbb{T}$ at a time, i.e.,

$$\mathbb{T} \Rightarrow \mathcal{DR}_i. \tag{10}$$

For a task $T_i \in \mathbb{T}$, its actual starting time and completion time are denoted as E_{st} and E_{cp}, respectively. Thus, its actual execution time is given as:

$$E_{ex} = E_{cp} - E_{st}. \tag{11}$$

Hence the collective actual execution time of a multi (n)-task $\mathbb{T}$ is given as $\sum_{i=1}^{n} \frac{E_{ex_i}}{n}$. Given a cluster of container-instances or nodes I_p in the edge device(s) attached to $\mathcal{DR}_i$, let $I_p^{\langle c,m\rangle}$ denote the p-th node's resource capacity. The estimated resource demands of k-dependent tasks to be orchestrated $\sum_{q=1}^{k} \widetilde{T}_q^{\langle c,m\rangle}$ and the resource capacity of each node is important information needed in order to make an efficient scheduling and co-location decision on I_p at time t. Our system extends to handle bulk requests from multiple end devices at the same location. Suppose at t, there are n service requests from multiple end devices at the same location L_i, where each device $\mathcal{D}_i$ is offloading $\mathbb{T}$. The collective n requests from the end devices can be scheduled as a multi-Job $\mathbb{J}$, where $\mathbb{J} = \sum_{i=1}^{n} \mathbb{T}_i$, with collective resource demand estimation of each job denoted as $\sum_{q=1}^{k} \widetilde{T}_q^{\langle c,m\rangle} = \widetilde{T}^{\langle c,m\rangle\prime}$, and the aggregate execution time estimation of each job as $\sum_{q=1}^{k} \widetilde{E}_{ex_q} = \widetilde{E}_{ex\prime}$. We can gang-schedule and co-locate $\mathbb{J}$ effectively on $\mathcal{DR}_i$:

$$\mathbb{J} \Rightarrow \mathcal{DR}_i, \tag{12}$$

by considering the estimated total resource demand of $\mathbb{J}$:

$$\sum\nolimits_{J \in \mathbb{J}} \widetilde{T}^{\langle c,m\rangle\prime} = \widetilde{T}^{\langle c,m\rangle\prime}_{\text{total}}, \tag{13}$$

and $\mathcal{DR}_i$ resource capability $\mathcal{RC}_i^{\langle c,m\rangle}$. Hence, the total estimated execution time of $\mathbb{J}$ at L_i is given as:

$$\sum\nolimits_{J \in \mathbb{J}} \widetilde{E}_{ex\prime} = \widetilde{E}^{\text{total}}_{ex\prime}. \tag{14}$$

Therefore, estimated resource utilization of $\mathcal{DR}_i$ for multi-job $\mathbb{J}$ at L_i is given by

$$\widetilde{\rho}_{\mathcal{DR}_i}^{\langle c,m\rangle} = \frac{\widetilde{T}_{\text{total}}^{\langle c,m\rangle\prime}}{\mathcal{RC}_i^{\langle c,m\rangle}}. \tag{15}$$

For a drone $\mathcal{DR}_i$, let the aggregate of the actual execution time of multi-job $\mathbb{J}$ at L_i be

$$\sum\nolimits_{J\in\mathbb{J}} \sum\nolimits_{q=1}^{k} \frac{E_{ex_q}}{k} = \sum\nolimits_{J\in\mathbb{J}} E_{ex\prime} = E_{ex\prime}^{\text{total}}, \tag{16}$$

and the total resources actually assigned for multi-job $\mathbb{J}$ at L_i be $\mathcal{RA}_{\mathcal{DR}_iU}^{\langle c,m\rangle}$. Under the condition that estimated total resource demand $\widetilde{T}_{\text{total}}^{\langle c,m\rangle\prime}$ is accurate, i.e., $\widetilde{T}_{\text{total}}^{\langle c,m\rangle\prime} \approx \mathcal{RA}_{\mathcal{DR}_iU}^{\langle c,m\rangle}$, then $\mathcal{RA}_{\mathcal{DR}_iU}^{\langle c,m\rangle}$ will not exceed $\mathcal{RC}_i^{\langle c,m\rangle}$. Similarly, under the condition that estimated total estimated execution time $\widetilde{E}_{ex\prime}^{\text{total}}$ is accurate, i.e., $\widetilde{E}_{ex\prime}^{\text{total}} \approx E_{ex\prime}^{\text{total}}$, then the drone $\mathcal{DR}_i$ will have sufficient flight time availability f_i^{aval} for its entire mission.

Our learning-based approach has significant advantages over non-learning based counterparts. By accurately estimating the resource requirement and execution times of multi-tasks/multi-jobs, our scheme can intelligently select suitable drones having requisite resource and flight time availability for the missions, and co-locate multi-dependent tasks in their attached edge nodes, such that the dependent tasks can communicate and execute faster, ultimately to minimize the response times and improve resource utilization, hence guarantees the entire mission completion. The accuracy of the estimated resource requirement and execution times can be ensured by constructing multiple training datasets for different classes of multi-tasks/multi-jobs from historical data to learn multiple models, one for a class of multi-tasks/multi-jobs. Given the multi-tasks/multi-jobs to be deployed, the model that is most similar to them is employed to estimate the resource requirement and execution times. Since the estimated total resource demand $\widetilde{T}_{\text{total}}^{\langle c,m\rangle\prime}$ and execution time $\widetilde{E}_{ex\prime}^{\text{total}}$ are accurate estimates of the actual total resource need to be allocated $\mathcal{RA}_{\mathcal{DR}_iU}^{\langle c,m\rangle}$ and actual execution time $E_{ex\prime}^{\text{total}}$, it is unlikely that the selected drone edge $\mathcal{DR}_i$ will not have sufficient resources. In other words, it is very unlikely that

$$f_i^{\text{total}} > f_i^{aval} \text{ and/or } \mathcal{RA}_{\mathcal{DR}_iU}^{\langle c,m\rangle} > \mathcal{RC}_i^{\langle c,m\rangle}, \tag{17}$$

which would lead to loss of job and jeopadize the entire mission. By contrast, standard non-learning based schemes have no means to intelligently select appropriate drones for ensuring that they will have sufficient resources, and the probability of (17) occurring can be much higher than our intelligent learning approach. There also exists simple and effective measure to guard against estimation error. It is obvious that loss of job may only occur in under estimation scenario. Instead of using the estimates of resource demand and execution time for selecting drones, we can add the two standard deviations of the estimation to the corresponding estimates and use these 'modified' or 'overly' estimated values to select the drones. This will reduce the probability of (17) occurring to almost zero. It is straightforward to provide both the estimate and estimation standard deviation by dividing the training data into multiple subsets and running the estimation procedure multiple times to provide the average estimate and estimation standard deviation.

Table 2
Common notations

Notation	Description
$\mathbb{EF}$	Federated edge deployments
$\mathbb{DR}$	A fleet of autonomous drones
$\mathbb{D}$	A set of end devices
$\mathbb{L}$	A set of disjointed locations
$\mathbb{T}$	A set of containerized inter-dependent applications
$\mathbb{DM}$	Distance matrix among the depot and target locations
T	Individual application or task
$\widetilde{T}^{\langle c,m\rangle}$	Task resource requirements estimation
$\widetilde{T}_{\text{total}}^{\langle c,m\rangle\prime}$	Estimated total resource requirements for jobs
I_i	Container-instance or node attached to a drone
$I_i^{\langle c,m\rangle}$	Resource capacity or availability of a node
$\mathcal{DR}_i$	Individual drone-enabled EC deployment
$\mathcal{RC}_i^{\langle c,m\rangle}$	Resource capacity in a drone attached edge devices
$\mathcal{RA}_{\mathcal{DR}_iU}^{\langle c,m\rangle}$	The total resources actually assigned for jobs
$\mathcal{DR}_i^{assign}$	A drone's set of assigned locations
$\mathcal{DR}_i^{route}$	The route for a drone's mission
$\mathcal{DR}_{iU}^{\langle c,m\rangle}$	Actual resources used for execution of jobs
$\mathcal{DR}_{iARU}^{\langle c,m\rangle}$	The actual resource usage of a cluster
$\rho_{\mathcal{DR}_i}^{\langle c,m\rangle}$	Actual resource utilization of jobs
$\widetilde{\rho}_{\mathcal{DR}_i}^{\langle c,m\rangle}$	Estimated resource utilization of jobs
$\rho_{\mathcal{DR}_i}^{\langle c\rangle}$, $\rho_{\mathcal{DR}_i}^{\langle m\rangle}$	Actual cluster CPU, memory resource utilization
E_st, E_cp	Application/task starting, completion time
E_{ex}	Application or task execution time
$E_{ex\prime}^{\text{total}}$	Actual total execution time for jobs
$\widetilde{E}_{ex}$	Application or task execution time estimation
$\widetilde{E}_{ex\prime}^{\text{total}}$	Estimated total execution time for jobs
L_0, L_i	Drone's depot and destination location
f_i^{aval}	Drone's flight time availability
f_i^{hover}	Drone's hovering time at location
d_i^{total}	Total distance of a drone's mission/trip
f_i^{total}	Total flight travel time of a drone's mission/trip
$d^{L_{i,j}}$	Travel distance of a drone from location i to j
$f^{L_{i,j}}$	Flight travel time of a drone from location i to j
ω_J	Number of instances of a Job
ϵ_J	The type of job
γ_J	Dependency depth of a job
$\boldsymbol{f}_{\text{mt}}$	Set of multi-task runtime parameters
Θ	Multi-output linear regression model
J, $\mathbb{J}$	A Job, A set of Jobs

4.2. Problem Formulation

The notations adopted are listed in Table 2. Our Multi-Location Capacitated Mission Scheduling Problem (ML-CMSP) is summarized as: *Given a federated system* $\mathbb{EF}$ *consisting of a fleet of autonomous drones* $\mathcal{DR}_i$, $1 \le i \le N$,

and a set of end devices $\mathcal{D}_i, 1 \leq i \leq M$ with their computing activities to be performed across multi-location $\mathbb{L} = \{L_1, \cdots, L_M\}$ within a city, the objectives are to co-schedule suitable drones, i.e., drones with sufficient flight time and computing resource availability, onto optimal mission routes among the locations, then co-locate the corresponding inter-dependent tasks on each drone's attached edge nodes at each location, such that all the activities are successful. Particularly, we present EdgeDrones, which intelligently co-schedules and co-locates multi-dependent tasks firmly on nodes, while considering task dependencies in oder to minimize the overall actual execution time and maximize the actual resource utilization, subject to certain constraints.

4.2.1. Constraints

At time $t > 0$, a set of end devices at different locations in the city require EC services, then a set of suitable drones are selected and co-schedule for EC missions. A drone can be assigned multiple disjointed locations as part of its mission, i.e., $\mathcal{DR}_i^{assign} = \{L_1, \cdots, L_n\} \subseteq \mathbb{L}$. For a drone $\mathcal{DR}_i$, its trip route is given as:

$$\mathcal{DR}_i^{route} = \Big(L_0 \rightarrow L_1 \rightarrow, \cdots, \rightarrow L_n, \rightarrow L_0\Big), \quad (18)$$

and no location is assigned twice within a mission, i.e.,

$$\vartheta\left[\mathcal{DR}_i^{assign}, L_i\right] = \begin{cases} 1, & \text{if } L_i \text{ is assigned to } \mathcal{DR}_i, \\ 0, & \text{otherwise,} \end{cases} \quad (19)$$

such that;

$$\mathcal{DR}_i^{assign} \cap \mathcal{DR}_k^{assign} = \emptyset, \forall i, k. \quad (20)$$

This is to ensure that a location is mapped to just one drone, and that no location is assigned twice within the missions, where the indicator $\vartheta\left[\mathcal{DR}_i^{assign}, L_i\right] = 1$ indicates that location L_i is assigned to the drone $\mathcal{DR}_i$; otherwise $\vartheta\left[\mathcal{DR}_i^{assign}, L_i\right] = 0$, all selected drones must start and end their missions at the depot L_0, i.e.,

$$\sum_{i=0}^{n}\sum_{j=i+1}^{m+1} d^{L_{i,j}} - \sum_{j=0}^{m}\sum_{i=j+1}^{n+1} d^{L_{j,i}} = 0, \forall \mathcal{DR}_i \in \mathbb{DR}, \quad (21)$$

and they must complete all their tasks at any location L_i before departing for another location L_j, i.e.,

$$\forall \mathcal{DR}_i \in \mathbb{DR},\ \varphi\left(\mathcal{DR}_i^{L_i \rightarrow L_j}\right) = \begin{cases} 1, \text{ if } \mathbb{J} \in [E_{cp}], \\ 0, \text{ if } \mathbb{J} \notin [E_{cp}], \end{cases} \quad (22)$$

where the indicator $\varphi\left(\mathcal{DR}_i^{L_i \rightarrow L_j}\right) = 1$ indicates that the drone $\mathcal{DR}_i$ has completed the execution of $\mathbb{J}$, and sent the results back to the devices at L_i. Hence it can depart L_i for Lj; otherwise $\varphi\left(\mathcal{DR}_i^{L_i \rightarrow L_j}\right) = 0$. The collective resource demand estimation of $\mathbb{J}$ at any of the locations assigned to each drone, cannot exceed its resource capacity. Recall that a drone's resource capacity $\mathcal{RC}^{\langle c,m\rangle}$ does not accumulate as it travels along its route. This is because, after executing its tasks at any location L_i, the results are sent back to the device(s) at L_i, and its resources becomes available again for the jobs at its next location L_j. Since the actual total resources that needs to be assigned to the multi-job at L_i is unknown at the scheduling stage, we use the estimated total resource demand $\widetilde{T}_{\text{total}}^{\langle c,m\rangle\prime}$ to replace it:

$$\widetilde{T}_{\text{total}}^{\langle c,m\rangle\prime} \leq \mathcal{RC}_i^{\langle c,m\rangle}, \quad \forall \mathcal{DR}_i \in \mathbb{DR}. \quad (23)$$

Durng the multi-task scheduling, unused or inactive attached nodes $I_i \in \mathcal{DR}_i$ in a selected drone would be shut down. All the nodes can be expressed in one of these two states: *Active* and *Inactive*. An *Active* node is a node that is running and is currently considered for allocation or has at least a job being started, executing or completing. An *Inactive* node is a node that is not running and and is not currently considered for allocation and not having at least a job that is being started, executing or completing. These two states can be expressed as follows:

$$\forall c, m\ \beta\left(I_i\right) = \begin{cases} 1, & \textit{Active} \text{ if } J_i \in [E_{st}, E_{cp}, E_{ex}], \\ 0, & \textit{Inactive} \text{ if } J_i \notin [E_{st}, E_{cp}, E_{ex}], \end{cases} \quad (24)$$

where the indicator $\beta\left(I_i\right) = 1$ indicates that the node I_i is ready to accept new jobs, and at least a job J_i is being started, executing or completing, i.e., $J_i \in [E_{st}, E_{cp}, E_{ex}]$, on I_i; otherwise $\beta\left(I_i\right) = 0$.

The aggregate actual execution time of $\mathbb{J}$ at all locations assinged to each drone $\sum_{L=i}^{n} E_{ex_i\prime}^{\text{total}}$ and the total flight travel time $\sum_{i=0}^{n}\sum_{j=i+1}^{m+1} f^{L_{i,j}}$ cannot exceed its flight time availability f_i^{aval}. Since the aggregate actual execution time $\sum_{L=i}^{n} E_{ex_i\prime}^{\text{total}}$ is unavailable at the scheduling stage, we replace it with the estimate $\sum_{L=i}^{n} \widetilde{E}_{ex_i\prime}^{\text{total}} \approx f^{hover\prime}$:

$$f_i^{total} = \sum_{i=0}^{n}\sum_{j=i+1}^{m+1} f^{L_{i,j}} + f_j^{hover\prime}, \quad (25)$$

therefore,

$$f_i^{total} \leq f_i^{aval}, \quad \forall f_i^{aval} \in \mathbb{DR}. \quad (26)$$

4.2.2. Optimization formulation

Suitable drones are selected and co-schedule with the least total distance to visit and execute tasks at target locations:

$$\textbf{Minimize} \quad d_i^{total} \quad \forall \mathcal{DR}_i \in \mathbb{DR}, \quad (27)$$

$$\textit{subject to} \quad \vartheta\left[\mathcal{DR}_i^{assign}, L_i\right] \in \{0, 1\}, \exists, \quad (28)$$

$$\mathcal{DR}_i^{assign} \cap \mathcal{DR}_k^{assign} = \emptyset, \forall i, k, \quad (29)$$

$$\sum_{i=0}^{n}\sum_{j=i+1}^{m+1} d^{L_{i,j}} - \sum_{j=0}^{m}\sum_{i=j+1}^{n+1} d^{L_{j,i}} = 0, \quad (30)$$

$$\varphi\left(\mathcal{DR}_i^{L_i \to L_j}\right) \in \{0,1\}, \forall \mathcal{DR}_i \in \mathbb{DR},\ \exists, \quad (31)$$

$$\widetilde{T}_{\text{total}}^{\langle c,m\rangle\prime} \leq \mathcal{RC}_i^{\langle c,m\rangle},\ \forall \mathcal{DR}_i \in \mathbb{DR}, \quad (32)$$

$$f_i^{\text{total}} \leq f_i^{aval},\ \forall f_i^{aval} \in \mathbb{DR}. \quad (33)$$

The objective function (27) is to minimize the total distance $d^{total} = \sum_{i=0}^{n}\sum_{j=i+1}^{m+1} d^{L_{i,j}}$ of each drone's route to visit its assigned locations. Constraints (28) to (30) and condition (31) ensure that no location is assigned twice within the missions, all drones must start and finish their trip at the depot, and each drone must execute all its task at any location before it departs for another location. Constraint (32) guarantees that $\widetilde{T}_{\text{total}}^{\langle c,m\rangle\prime}$ of $\mathbb{J}$ at any assigned location for $\mathcal{DR}_i$ would not exceed its resource capacity $\mathcal{RC}_i^{\langle c,m\rangle}$. Constraint (33) also guarantees that each drones' total flight travel time f_i^{total} for any mission would not exceed its flight time availability f_i^{aval}. The details of our optimal mission and route planning is given in Section 4.3 and in Algorithm 2.

As the actual resource utilization of a cluster/edge is unknown, we maximize the estimated resource utilization:

$$\textbf{Maximize} \quad \widetilde{\rho}_{\mathcal{DR}_i}^{\langle c,m\rangle}, \quad (34)$$

$$\textit{subject to} \quad \mathbb{J} \Rightarrow \mathcal{DR}_i,\ \exists, \quad (35)$$

$$\widetilde{T}_{\text{total}}^{\langle c,m\rangle\prime} \leq \mathcal{RC}_i^{\langle c,m\rangle},\ \forall \mathcal{DR}_i \in \mathbb{DR}, \quad (36)$$

$$f_i^{\text{total}} \leq f_i^{aval},\ \forall f_i^{aval} \in \mathbb{DR}, \quad (37)$$

$$\varphi\left(\mathcal{DR}_i^{L_i \to L_j}\right) \in \{0,1\}, \forall \mathcal{DR}_i \in \mathbb{DR},\ \exists, \quad (38)$$

$$\beta\left(I_i\right) \in \{0,1\},\ \forall c, m,\ \exists. \quad (39)$$

Provided that the estimated resource utilization $\widetilde{\rho}_{\mathcal{DR}_i}^{\langle c,m\rangle}$ is accurate, little optimality will be lost. The constraints (35) to (37) indicate the dispatching of multi-job $\mathbb{J}$ at each assigned location to $\mathcal{DR}_i$, given that its resources and flight time availability is sufficient. More specifically, (35) is the multi-job $\mathbb{J}$ deployment constraint, guaranteeing that $\mathbb{J}$ is gang-scheduled onto $\mathcal{DR}_i$ attached resources, such that dependent tasks within each $J \in \mathbb{J}$ can communicate and execute faster. The constraint (36) guarantees that $\widetilde{T}_{\text{total}}^{\langle c,m\rangle\prime}$ of $\mathbb{J}$ would not exceed $\mathcal{RC}_i^{\langle c,m\rangle}$ of $\mathcal{DR}_i$, and constraint (37) guarantees that f_i^{total} would not exceed f_i^{aval} of any $\mathcal{DR}_i \in \mathbb{DR}$. The condition (38) guarantees that active nodes $\left(\beta\left(I_i\right)=1\right)$ would be used for execution, and inactive nodes $\left(\beta\left(I_i\right)=0\right)$ would be shut down. Hence, our aim is to minimize the number of active nodes used for execution by co-locating jobs tightly on each node in order to maximize resource utilization. We shall discuss the details of our multi-job co-location principle in Section 4.3 and Algorithm 3.

Then again, $\widetilde{E}_{ex\prime}^{\text{total}}$ of $\mathbb{J}$ can be minimized depending on scheduling:

$$\textbf{Minimize} \quad \widetilde{E}_{ex\prime}^{\text{total}}, \quad (40)$$

$$\textit{subject to} \quad \mathbb{J} \Rightarrow \mathcal{DR}_{i^\star},\ \exists, \quad (41)$$

$$\widetilde{T}_{\text{total}}^{\langle c,m\rangle\prime} \leq \mathcal{RC}_i^{\langle c,m\rangle},\ \forall \mathcal{DR}_i \in \mathbb{DR}, \quad (42)$$

$$f_i^{\text{total}} \leq f_i^{aval},\ \forall f_i^{aval} \in \mathbb{DR}, \quad (43)$$

$$\varphi\left(\mathcal{DR}_i^{L_i \to L_j}\right) \in \{0,1\}, \forall \mathcal{DR}_i \in \mathbb{DR},\ \exists. \quad (44)$$

Note that the actual overall execution time $E_{ex\prime}^{\text{total}}$ is unknown at this stage, and we use the estimated overall execution $\widetilde{E}_{ex\prime}^{\text{total}}$ to replace it in the optimization. Again, provided that the estimate $\widetilde{E}_{ex\prime}^{\text{total}}$ is accurate, little optimality will be lost. The constraint (41) guarantees that $\mathbb{J}$ at any location is dispatched to $\mathcal{DR}_i$, such that dependent tasks within each $J \in \mathbb{J}$ can communicate and execute faster. The constraint (42) guarantees that $\widetilde{T}_{\text{total}}^{\langle c,m\rangle\prime}$ of $\mathbb{J}$ would not exceed $\mathcal{RC}_i^{\langle c,m\rangle}$ of $\mathcal{DR}_i$, and constraint (43) guarantees that f_i^{total} would not exceed f_i^{aval} of any $\mathcal{DR}_i \in \mathbb{DR}$. The condition (44) ensures that each drone must execute all its task at any location before it departs for another location.

4.3. EdgeDrones Algorithm Framework

Our EdgeDrones approach consists of linear regression estimation, optimal route planning for multi-drone, and gang scheduling of tasks. These three components aim at providing optimal solution for our Multi-Location Capacitated Mission Scheduling Problem (MLCMSP). Particularly, the optimization (27), (34) and (40) aim at ensuring least travel distance for drones to visit their assigned locations, and every task at each location is fast executed given the available resources, such that the missions are accomplished. The values of the linear regression estimations are required by the router, as well as the update state of the drones' flight time availabity for effective route planning and assignment, while our gang-scheduling approach involves co-scheduling and co-locating tasks firmly on available resources. We detail the procedures of the three components of EdgeDrones as follows:

4.3.1. Resource and execution time estimation

Algorithm 1 describes the resource and execution time estimations for multi-job. As $\mathbb{J}$ are released, their collective resource requirement $\widetilde{T}_{\text{total}}^{\langle c,m\rangle\prime}$ and execution time $\widetilde{E}_{ex\prime}^{\text{total}}$ are estimated. The set of runtime parameters $\boldsymbol{f}_{\text{mt}}(\omega, \epsilon, \gamma)$, where ω is the number of instances, ϵ is type of tasks , γ is dependency depth, are fed into the model $\Theta^\star$ to to produce the estimation values (line $2 \sim 9$). Once the estimation values are produced, they are used in the assignment and route planning.

4.3.2. Mission planning

Given the update state from the CP, i.e., all ready drones $\mathcal{DR}_i \in \mathbb{DR}$ at the depot L_0 (which include each drones'

Algorithm 1 Linear Regression Estimation

Input: At $t > 0$, $\mathbb{J}$ at L_i; $\boldsymbol{f}_{\text{mt}}$ are fed into $\Theta^{\star}$
Output: $\widetilde{T}_{\text{total}}^{\langle c,m\rangle\prime}$ and $\widetilde{E}_{ex\prime}^{\text{total}}$

for $J_i \in \mathbb{J}$ **do**
  Type of Job $J = \epsilon_J$
  Number of instances of Job $J = \omega_J$
  Dependency depth of Job $J = \gamma_J$
  for $T_i \in J_i$ **do**
    $\boldsymbol{f}_{\text{mt}}(\omega, \epsilon, \gamma) \cdot \Theta^{\star} = \left[\widetilde{T}_i^{\langle c,m\rangle}\ \widetilde{E}_{ex_i}\right]$
  end for
  $\widetilde{T}_i^{\langle c,m\rangle\prime} = \widetilde{T}_i^{\langle c,m\rangle\prime} + \widetilde{T}_i^{\langle c,m\rangle}$
  $\widetilde{E}_{ex_i\prime} = \widetilde{E}_{ex_i\prime} + \widetilde{E}_{ex_i}$
end for

Algorithm 2 Mission Planning

Input: At $t > 0$; $L_i \in \mathbb{L}$; $\widetilde{T}_{\text{total}}^{\langle c,m\rangle\prime} \in L_i$; $\widetilde{E}_{ex\prime}^{\text{total}} \in L_i$; $\mathcal{DR}_i \in \mathbb{DR}$; $f^{aval} \in \mathcal{DR}_i$; $\mathcal{RC}^{\langle c,m\rangle} \in \mathcal{DR}_i$; and $\mathbb{DM}$
Output: $\mathcal{DR}_i^{assign}$ and $\mathcal{DR}_i^{route}$

for $\mathcal{DR}_i \in \mathbb{DR}$ **do**
  Initialize total flight travel time f_i^{total} for $\mathcal{DR}_i$
  Initialize total travel distance d_i^{total} for $\mathcal{DR}_i$
  f_i^{aval} = Flight travel time availability of $\mathcal{DR}_i$
  $\mathcal{RC}_i^{\langle c,m\rangle}$ = Resource capacity of $\mathcal{DR}_i$
  $\mathbb{L}$ = Target locations
  L_i = route start for $\mathcal{DR}_i$
  while $L_i \neq$ end of route **do**
    $\widetilde{T}_{\text{total}}^{\langle c,m\rangle\prime}$ = Resource demand at L_i
    $\widetilde{E}_{ex\prime}^{\text{total}}$ = Execution time estimation at L_i
    $f_i^{total} = d_i^{total} + \widetilde{E}_{ex\prime}^{\text{total}}$
    if $f_i^{total} \leq f_i^{aval}$; $\vartheta\left[\mathcal{DR}_i^{assign}, L_i\right] = 0$; and $\widetilde{T}_{\text{total}}^{\langle c,m\rangle\prime} \leq \mathcal{RC}_i^{\langle c,m\rangle}$ **then**
      $\vartheta\left[\mathcal{DR}_i^{assign}, L_i\right] = 1$
      L_j = next feasible neighbor
      $L_i = L_j$
    else
      L_j = next feasible neighbor
      $L_i = L_j$
    end if
    $d_i^{total} = d_i^{total} + d^{L_{i,j}}$
  end while
end for

flight time availability f_i^{aval} and resource capacity $\mathcal{RC}_i^{\langle c,m\rangle}$), all end devices at target locations $L_i \in \mathbb{L}$ (which include each device set of inter-dependent tasks resource requirement $\widetilde{T}_{\text{total}}^{\langle c,m\rangle\prime}$ and execution time $\widetilde{E}_{ex\prime}^{\text{total}}$ estimation), and the distance matrix $\mathbb{DM}$, such that starting from L_0, a route is iteratively built and assigned to a drone $\mathcal{DR}_i$, by selecting from among the nearest locations which meet the constraints, i.e., L_1 whose its end device(s) $\widetilde{T}_{\text{total}}^{\langle c,m\rangle\prime}$ and $\widetilde{E}_{ex\prime}^{\text{total}}$ does not exceed the drone's $\mathcal{RC}_i^{\langle c,m\rangle}$ and f_i^{aval}, respectively, and

Algorithm 3 Multi-job Co-location

Input: $\mathbb{J}$ gang-scheduled onto $\mathcal{DR}_{i_\star}$, resource demand estimation $\sum_{J\in\mathbb{J}} \widetilde{T}_J^{\langle c,m\rangle\prime}$, resource availability $I_i^{\langle c,m\rangle}$ of all nodes $I_i \in \mathcal{DR}_{i_\star}$
Output: $\mathbb{J}$ is co-located, such that **Minimize** $\sum_{I_i\in\mathcal{DR}_{i_\star}} I_i$

for $I_i \in \mathcal{DR}_{i_\star}$ **do**
  if $\beta\left(I_i\right) = 1$ **then**
    $I_i^{\langle c,m\rangle} = \langle c, m\rangle$, i.e., initial resource available
    for $J \in \mathbb{J}$ **do**
      if $\Gamma\left[J, I_i\right] = 0$ and $\widetilde{T}_J^{\langle c,m\rangle\prime} \leq I_i^{\langle c,m\rangle}$ **then**
        $J \Rightarrow I_i$
        $\Gamma\left[J, I_i\right] = 1$
        $I_i^{\langle c,m\rangle} = I_i^{\langle c,m\rangle} - \widetilde{T}_J^{\langle c,m\rangle\prime}$
      end if
      if $I_i^{\langle c,m\rangle}$ close to zero **then**
        break
      end if
    end for
  end if
end for

whose flight travel time f_1^{total} does not exceed the drone's f_i^{aval}. This process resumes from L_1 to find L_2, and so on until there is no feasible neighbor (line $6 \sim 17$). L_0 is finally added to conclude the route. This procedure is repeated to find other routes until all the possible target locations are chosen. This mission planning also aims to optimized and minimize the distance travelled by each drone, which lets us find an optimal solution from the given Algorithm 2. To solve this mission scheduling problem, we have adopted the CP-SAT solver and the MPSolver wrapper.

4.3.3. Co-location

In the edge resources attached to each drone $\mathcal{DR}_i$, our co-location algorithm uses the $I_i^{\langle c,m\rangle}$ and $\widetilde{T}_i^{\langle c,m\rangle\prime}$ of each $J_i \in \mathbb{J}$ to provide efficient co-location, such that fewer nodes are used for execution at each location. Specifically, the gang scheduling approach is adopted alongside our bin-packing optimization to co-schedule and co-locate $\mathbb{J}$ at a time. Bin-packing is one the of most popular packing problems. The goal is to minimize the number of nodes used as given in optimization (45). Unlike other approaches, such as first fit bin packing problem (FFBPP) [49], it requires the next J_i to be placed on the active node, otherwise, it is placed on a new node. Our approach scans all $J \in \mathbb{J}$ and maps J_i to active nodes in full utilization. All $J \in \mathbb{J}$ are co-located firmly on active nodes, so that resource wastage is avoided and fewer nodes are used to execute all jobs concurrently. Hence our co-location strategy is to find the solution to the problem:

$$\textbf{Minimize} \sum_{I_i\in\mathcal{DR}_{i_\star}} I_i, \tag{45}$$

$$\textit{subject to } \mathbb{J} \Rightarrow \mathcal{DR}_{i_\star}, \exists, \tag{46}$$

Table 3
Multi-job across multiple disjointed locations in a city, where the actual resource consumed for multi-job execution $T_{\text{total}}^{\langle c,m\rangle\prime}$ and the actual execution time $E_{ex\prime}^{\text{total}}$ are taken from the original Alibaba data, while the estimated resource demand $\widetilde{T}_{\text{total}}^{\langle c,m\rangle\prime}$ and execution time $\widetilde{E}_{ex\prime}^{\text{total}}$ are calculated by Algorithm 1

$\mathbb{L}$	$\mathbb{J}$	$\mathbb{T}$	$\widetilde{T}_{\text{total}}^{\langle c,m\rangle\prime}$	$T_{\text{total}}^{\langle c,m\rangle\prime}$	**NAEE**	$\widetilde{E}_{ex\prime}^{\text{total}}$ (s)	$E_{ex\prime}^{\text{total}}$ (s)	**NAEE**
L_1	2	10	$\langle 570.18, 1.98\rangle$	$\langle 595, 1.92\rangle$	$\langle 0.04, 0.03\rangle$	173.46	148	0.17
L_2	3	12	$\langle 625.06, 2.37\rangle$	$\langle 540, 1.85\rangle$	$\langle 0.15, 0.28\rangle$	189.03	164	0.15
L_3	2	9	$\langle 478.02, 1.73\rangle$	$\langle 340, 0.96\rangle$	$\langle 0.4, 0.8\rangle$	167.8	142	0.18
L_4	2	8	$\langle 398.42, 1.71\rangle$	$\langle 445, 1.42\rangle$	$\langle 0.1, 0.2\rangle$	56.69	44	0.28
L_5	5	21	$\langle 1135.11, 4.13\rangle$	$\langle 1035, 3.38\rangle$	$\langle 0.09, 0.22\rangle$	355.68	309	0.15
L_6	5	23	$\langle 1228.72, 4.56\rangle$	$\langle 1080, 3.4\rangle$	$\langle 0.13, 0.3\rangle$	370.27	311	0.19
L_7	5	19	$\langle 1005.85, 3.89\rangle$	$\langle 1070, 3.39\rangle$	$\langle 0.05, 0.14\rangle$	236.95	198	0.19
L_8	3	15	$\langle 773.7, 3.02\rangle$	$\langle 655, 2.13\rangle$	$\langle 0.18, 0.4\rangle$	211.06	178	0.18
L_9	2	10	$\langle 570.18, 1.98\rangle$	$\langle 595, 1.92\rangle$	$\langle 0.04, 0.03\rangle$	173.46	148	0.17
L_{10}	3	14	$\langle 727.81, 2.8\rangle$	$\langle 670, 2.1\rangle$	$\langle 0.08, 0.3\rangle$	202.45	172	0.17
L_{11}	3	15	$\langle 773.7, 3.02\rangle$	$\langle 655, 2.13\rangle$	$\langle 0.18, 0.4\rangle$	211.06	178	0.18
L_{12}	4	17	$\langle 876.45, 3.44\rangle$	$\langle 785, 2.38\rangle$	$\langle 0.11, 0.4\rangle$	224.49	188	0.19
L_{13}	5	19	$\langle 1025.99, 3.7\rangle$	$\langle 885, 2.88\rangle$	$\langle 0.15, 0.28\rangle$	341.79	298	0.14
L_{14}	3	17	$\langle 925.32, 3.48\rangle$	$\langle 990, 3.14\rangle$	$\langle 0.06, 0.1\rangle$	341.79	182	0.87
L_{15}	3	14	$\langle 727.81, 2.8\rangle$	$\langle 670, 2.1\rangle$	$\langle 0.08, 0.33\rangle$	202.45	172	0.17
L_{16}	4	15	$\langle 773.7, 3.02\rangle$	$\langle 655, 2.13\rangle$	$\langle 0.18, 0.4\rangle$	202.45	178	0.13

$$\sum_{J\in\mathbb{J}} \Gamma\left[J,\ I_i\right]\cdot \widetilde{T}_J^{\langle c,m\rangle\prime} \le I_i^{\langle c,m\rangle},\quad \forall c,m, \tag{47}$$

where

$$\Gamma\left[J,\ I_i\right] = \begin{cases} 1, & \text{if } J \Rightarrow I_i,\\ 0, & \text{otherwise.}\end{cases} \tag{48}$$

The constraint (46) is the multi-job $\mathbb{J}$ deployment constraint, guaranteeing that $\mathbb{J}$ is gang-scheduled to $\mathcal{DR}_{i_\star}$, such that dependent tasks within each $J\in\mathbb{J}$ can communicate and execute faster. The constraint (47) indicates that the total estimated resource requirements of co-located jobs $\sum_{i=1}^{N}\widetilde{T}_i^{\langle c,m\rangle\prime}$ cannot exceed $I_i^{\langle c,m\rangle}$, the node resource availability. The condition (48) means that if job J_i is placed on the node I_i, then $\Gamma\left[J_i,\ I_i\right]=1$; otherwise, $\Gamma\left[J_i,\ I_i\right]=0$. This is to guarantee that each $J\in\mathbb{J}$ is placed in exactly one node. To solve this multi-job packing problem, we have adopted the solving Constraint Integer Programs (SCIP) solver, which is currently one of the fastest mathematical programming (MP) solvers for this problem [12]. Algorithm 3 describes the co-location strategy which co-locates multi-dependent tasks firmly on nodes, such that for any given jobs, resource wastage is avoided and fewer nodes are used for execution. It takes the resource demand estimation of multi-task/job and resource availability of nodes as input, then scans all $J\in\mathbb{J}$ and maps them to active nodes in full utilization (line $2\sim 7$).

4.3.4. Connection with optimization objectives

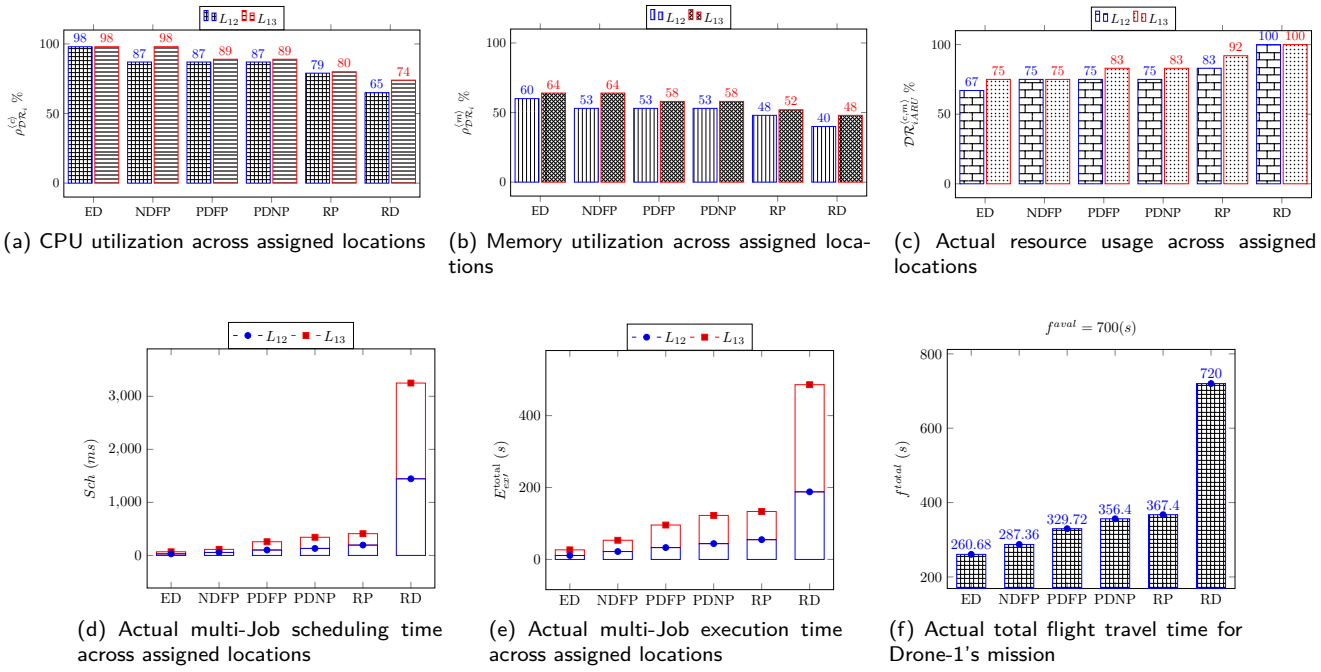
As stated previously, our objectives are to minimize the number of selected drones and total distance travelled by each drone, maximize the actual edge cluster resource utilization, and to minimize the overall actual execution time of the task-dependent multi-jobs. Algorithms 1, 2 and 3 together achieve these objectives. By gang-dispatching the task-dependent multi-jobs to an edge having the sufficient resource for the jobs and flight time availability, Algorithm 2 ensures that drones assigned for missions allocates the sufficient actual resources needed for jobs execution $\mathcal{DR}_{iU}^{\langle c,m\rangle}$, such that the dependent tasks can be executed faster, ultimately leading to a smaller actual aggregate execution time $E_{ex\prime}^{\text{total}}$ and better actual cluster resource utilization. By intelligently packing dependent tasks tightly on nodes, Algorithm 3 is capable of fully utilizing available resources at edge clusters, ultimately leading to the actual resource assigned to the execution of jobs $\mathcal{DR}_{iU}^{\langle c,m\rangle}$ as small as possible while guaranteeing it is sufficient for the multi-jobs. More specifically, the actual resource usage (ARU) of the cluster for multi-job $\mathbb{J}$ deployment is given by

$$\mathcal{DR}_{iARU}^{\langle c,m\rangle} = \frac{\mathcal{DR}_{iU}^{\langle c,m\rangle}}{\mathcal{RC}_i^{\langle c,m\rangle}}. \tag{49}$$

It can be seen that solving the optimization (45) is directly linked to minimize the ARU (49). Let the actual CPU resource and the actual memory resource assigned for $\mathbb{J}$ be $\mathcal{DR}_{iU}^{\langle c\rangle}$ and $\mathcal{DR}_{iU}^{\langle m\rangle}$, respectively. Further denote the actual CPU consumed and the actual memory consumed in executing $\mathbb{J}$ as $\sum_{J\in\mathbb{J}}T^{\langle c\rangle\prime}$ and $\sum_{J\in\mathbb{J}}T^{\langle m\rangle\prime}$, respectively. Then the actual CPU utilization $\rho_{\mathcal{DR}_i}^{\langle c\rangle}$ and the actual memory utilization $\rho_{\mathcal{DR}_i}^{\langle m\rangle}$ are defined respectively by

$$\rho_{\mathcal{DR}_i}^{\langle c\rangle} = \frac{\sum_{J\in\mathbb{J}}T^{\langle c\rangle\prime}}{\mathcal{DR}_{iU}^{\langle c\rangle}}, \tag{50}$$

$$\rho_{\mathcal{DR}_i}^{\langle m\rangle} = \frac{\sum_{J\in\mathbb{J}}T^{\langle m\rangle\prime}}{\mathcal{DR}_{iU}^{\langle m\rangle}}. \tag{51}$$



(a) CPU utilization across assigned locations

(b) Memory utilization across assigned locations

(c) Actual resource usage across assigned locations

(d) Actual multi-Job scheduling time across assigned locations

(e) Actual multi-Job execution time across assigned locations

(f) Actual total flight travel time for Drone-1's mission

Figure 4: Activities and utilities of Drone-1's mission.

Algorithms 2 and 3 are directly connected with minimizing d^{total}, minimizing $E^{\text{total}}_{ex'}$ as well as maximizing $\rho^{\langle c \rangle}_{DR}$ and maximizing $\rho^{\langle m \rangle}_{DR}$.

5. Performance Evaluation

In this section, we described our experimental setup including cluster resource configuration, the Alibaba cluster data traces used, and the comparison baselines. We perform extensive experiments to compare EdgeDrones against some existing schemes. We will also compare the performance of EdgeDrones against exisiting schemes in individual drones. We show that EdgeDrones can achieve minimized actual execution time of multi-dependent tasks, achieve high resource utilization, achieve load balancing, use fewer cluster resources and avoid loss of job in an aerial edge computing system.

5.1. Experimental Setup

Drone's Mission Scheduling and Resources: Our ML-CMSP is implemented using Google OR Tools[12]. It uses MPSolver wrapper for solving LP and MIP problems. We perform experiments for a set of drones with computing resource and flight time constraints, among a set of 16 target locations in a city. At each of these locations are end devices, with multi-dependent tasks/jobs needed to be executed. Our linear regression ML model, as given in Algorithm 1, estimates the resource demands and execution time of tasks at each location (as shown in Table 3).

Each of the drones has different payload capacity. The payload is the *weight* a drone can carry in the air. The total weight of the payload has a great impact on the flight time of the drone. For example, the Aurelia X8 Standard drone[13] with a payload of 8 kg, has a net flight time of 25 min. However, with a maximum payload capacity, the maximum flight time availability of the drone will be 12 min. It has a maximum flight speed s of 15 m/s and a maximum wind resistance speed of 9 m/s. Also, for the Aurelia X6 Pro drone, its net flight time is 55 min, however, with its full payload of 5 kg, it will have a 30 min flight time availability. Hence, the flight time availability of the selected drones, as well as their resource capacities is given in Table 4.

The optimal routes of the selected drones is also given in Table 4. These optimal routes are derived using Algorithm 2. The entire missions covered a distance of 6552 meters, with total flight travel time of 3659 seconds, given that each drone is traveling at a constant flight speed s of 13 m/s.

Multi-dependent Tasks: We employ the v-2018 version of Alibaba cluster trace[14], which records the activities of about 4000 machines in a perids of 8 days. The entire trace contains more than 14 million tasks with more than 12 million dependencies, and more than 4 million jobs. Among which we have deployed 54 jobs with total of 238 tasks (including dependencies) for our experiments. The number of tasks within each job ranges from (1, 5], while the task dependency depth among the jobs ranges from (1, 4]. The multi-task dependencies in in the data trace is valuable for our investigation. Researchers have thoroughly investigated

[12] https://developers.google.com/optimization

[13] https://aurelia-aerospace.com/our-drones/

[14] https://github.com/alibaba/clusterdata

Table 4
Drones assigned locations, optimal routes and resources

$\mathcal{DR}_i$	$\mathcal{DR}_i^{route}$	$d_i^{total}(m)$	$f_i^{hover\prime}(s)$	**Attached Edge Devices and total weight**	$\mathcal{RC}_i^{\langle c \rangle}$	$\mathcal{RC}_i^{\langle m \rangle}$	$f_i^{aval}(s)$
$\mathcal{DR}_1$	$\{L_0 \rightarrow L_{12} \rightarrow L_{13} \rightarrow L_0\}$	936	566	`Huawei AR502H Series x3` = 3.3kg	12 Cores	6 GiB	600
$\mathcal{DR}_2$	$\{L_0 \rightarrow L_9 \rightarrow L_{14} \rightarrow L_{16} \rightarrow L_8 \rightarrow L_7 \rightarrow L_0\}$	1712	1165	`HIVECELL x2` = 2.72kg	12 Cores	16 GiB	1200
$\mathcal{DR}_3$	$\{L_0 \rightarrow L_1 \rightarrow L_4 \rightarrow L_3 \rightarrow L_{15} \rightarrow L_{11} \rightarrow L_0\}$	2192	811	`HIVECELL + Huawei AR502H Series` = 2.46kg	10 Cores	8 GiB	900
$\mathcal{DR}_4$	$\{L_0 \rightarrow L_{10} \rightarrow L_2 \rightarrow L_6 \rightarrow L_5 \rightarrow L_0\}$	1712	1117	`Azure Stack Edge mini` = 3.17kg	16 Cores	48 GiB	1200

v-2018 version of Alibaba cluster trace and used it for various task scheduling problems [11, 12, 50, 51].

Comparison Baselines: We compare the scheduling approach of EdgeDrones **(ED)** with the following three existing schemes and the random approach, fixing each drone's routes to that of EdgeDrones, as follows:

1. An approach which does not consider tasks' dependencies, but schedules 50% of any given multi-dependent tasks by mainly focusing on task co-location. We refer to this approach as No Dependency and Full Packing **(NDFP)**, and it is similar to the approach in [46].
2. An approach which schedules up to 40% of any given multi-dependent tasks with task co-location. We consider this approach as a Partial Dependency and Full Packing **(PDFP)**, and it is similar to to the approach in [48].
3. An approach which schedules up to 15% of any given multi-dependent tasks at a time, but does not consider task co-location. We refer to this approach as Partial Dependency and No Packing **(PDNP)**, and it is similar to to the approach in [47].
4. An approach which schedules tasks according to their resource requests for execution, i.e., the more resource demand, the higher priority for the task to be scheduled and allocate resources. We refer to this strategy as Resource Priority **(RP)**, and it is similar to the approach in [52].
5. Random **(RD)** approach schedules a single task individually and assumes a node can only execute a task at a time.

5.2. Deployment Results and Performance Comparison

Our investigation focuses on CPU and memory usage/utilization, task deployment, scheduling time, execution time and successful mission completion. The results obtained by ED, NDFP, PDFP, PDNP, RP and RD are compared.

5.2.1. Resource and execution time estimation accuracy

As detailed in the previous section, to implement the proposed learning based intelligent drone routing and multi-task co-location strategy, we train a linear regression model from a training dataset. In the real-time application experiments, the trained model is used to estimate the resource requirement and execution time (Algorithm 1). The estimated resource requirement and execution time are then employed to aid our optimal route planning and intelligent multi-task scheduling strategy (Algorithms 2 and 3). Clearly, the accuracy of Algorithm 1 impacts the achievable performance of our EdgeDrones. Therefore, we first investigate the accuracy of our trained linear regression model.

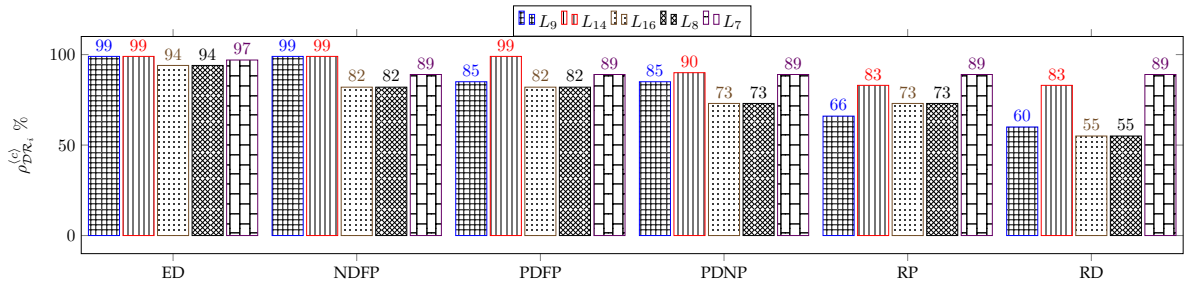
The multi-job execution information across the federated deployments, obtained according to Alibaba data, are listed in Table 3, where the estimated resource demand $\widetilde{T}_{\text{total}}^{\langle c,m \rangle\prime}$ and the estimated execution time $\widetilde{E}_{ex\prime}^{\text{total}}$ are calculated using Algorithm 1, while the actual resource consumed for the multi-job execution $T_{\text{total}}^{\langle c,m \rangle\prime}$ and the actual execution time $E_{ex\prime}^{\text{total}}$ are taken from the original data. The normalized absolute estimate error (NAEE), defined as

$$\text{NAEE} = \frac{\left|\text{estimated value} - \text{actual value}\right|}{\text{actual value}}, \tag{52}$$

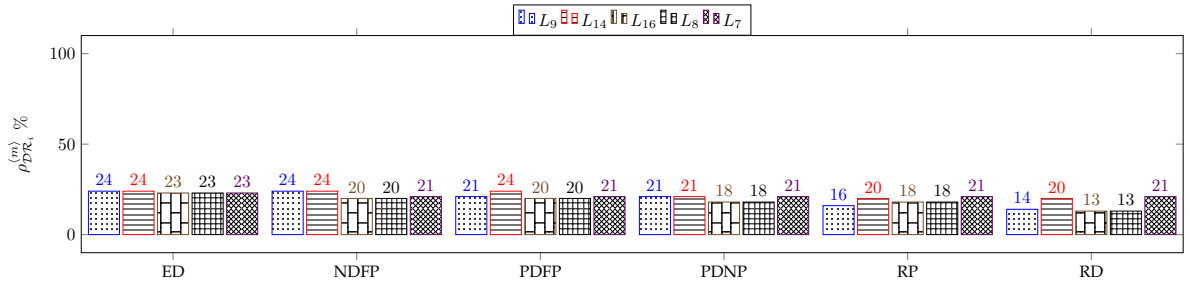
is also listed in Table 4 for both resource consumed and execution time, which serves as the estimation accuracy measure for the trained multi-output linear regression model. The average NAEE across 16 locations is 0.13 for CPU resource, 0.28 for memory resource, and 0.22 for execution time. From Tables 3 and 4, it can be seen that $\widetilde{T}_{\text{total}}^{\langle c,m \rangle\prime} < \mathcal{DR}^{\langle c,m \rangle}$, $T_{\text{total}}^{\langle c,m \rangle\prime} < \mathcal{DR}_i^{\langle c,m \rangle}$ and $f^{total} < f^{aval}\ \forall \mathcal{DR}_i$, given that each drone is traveling at a constant flight speed s of 13 m/s. In other words, each drone has sufficient resource to execute its multi-jobs assigned to it. This further indicates the suitability or accuracy of the trained ML model to provide the necessary information for our intelligent co-location strategy.

5.2.2. Performance comparison across integrated edge resources

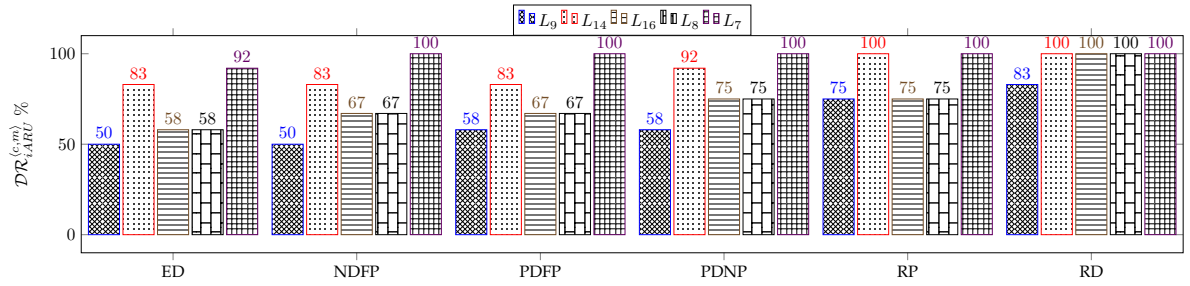
After completing the optimal route planning, as shown in Table 4, we are now ready to co-schedule the drones for their missions, apply our EdgeDrones to orchestrate 54 jobs with 238 tasks among the four drones and compare its performance with those of the benchmark schemes. We first investigate the *CPU utilization* across the assigned locations of the four drones, depicted in Figs. 4(a), 5(a), 6(a) and 7(a). It can be observed that both EdgeDrones (ED) achieved the highest CPU utilization across the entire missions. Specifically, in Drone-1's mission, as shown in Fig. 4(a), EdgeDrones acheieves an average of 98% CPU utilization across Drones-1's assigned locations. This is followed by the NDFP, PDFP and PDNP schemes, which achieves the same average of 92.5%. The remaining two schemes, RP and RD achieves the lowest CPU utilization across the same assigned locations of Drone-1. RP achieves



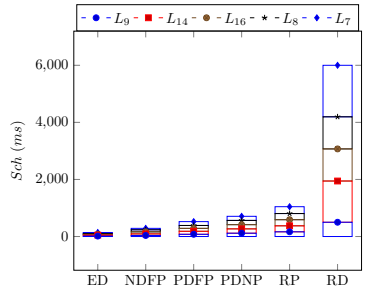
(a) CPU utilization across assigned locations



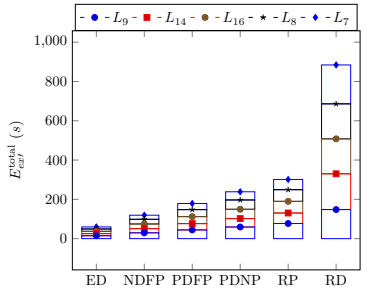
(b) Memory utilization across assigned locations



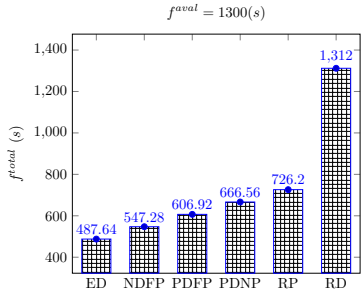
(c) Actual resource usage across assigned locations



(d) Actual multi-Job scheduling time across assigned locations



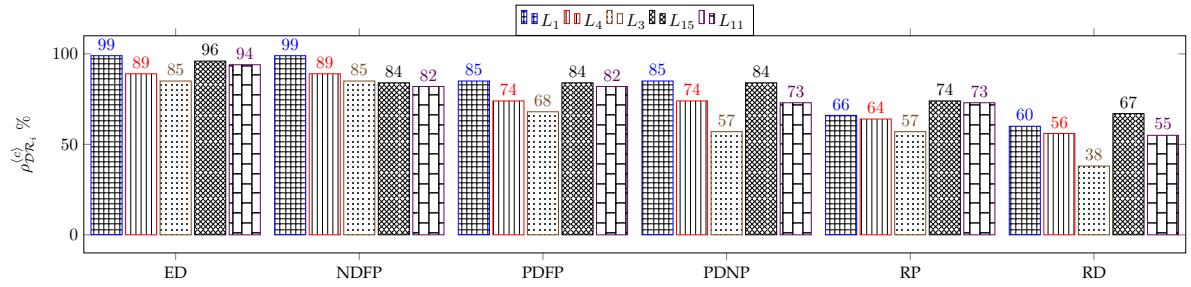
(e) Actual multi-Job execution time across assigned locations



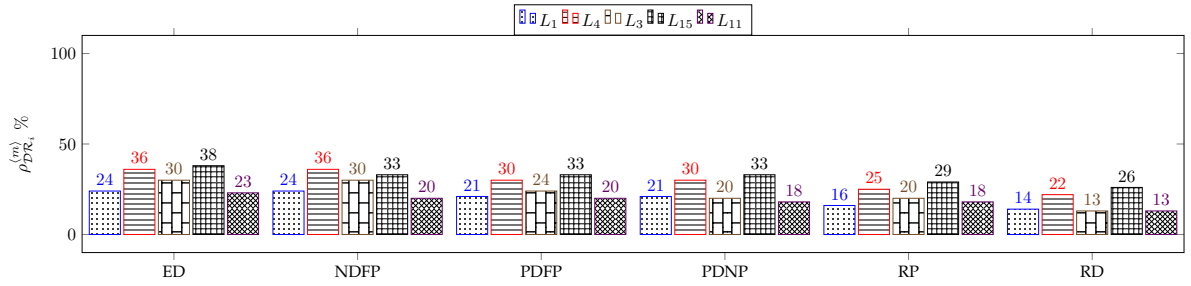
(f) Actual total flight travel time for Drone-2's mission

Figure 5: Activities and utilities of Drone-2's mission.

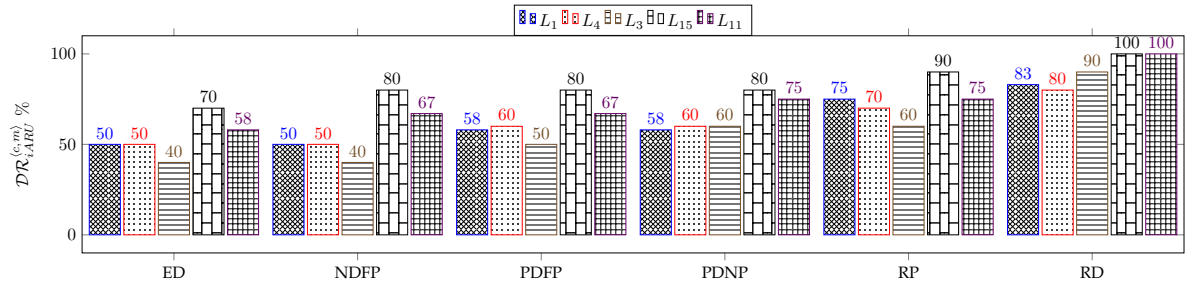
an average of 79.5%, while RD obviously achieves an average of 69.5%. It can also be seen that Edgedrones achieves the highest CPU utilization, according to Figs. 5(a), 6(a) and 7(a) for Drone-2, Drone-3 and Drone-4, respectively. Edgedrones is able to intelligently gang-schedule and co-locate all task tightly on nodes, resulting in higher resource utilization. In Figs. 5(a), 6(a) and 7(a), EdgeDrones achieves the highest average CPU utilization of 96.6, 92.6 and 94.5, respectively, compared to other schemes. In particular, NDFP achieves the second highest average CPU utilization across the assigned locations of the drones, i.e., it achieves an average CPU utilization of 6.4%, 4.8% and 3% less than EdgeDrones across Drone-2. Drone-3 and Drone-4 missions, respectively. PDFP and PDNP schemes performed averagely in terms of CPU utilization compared to EdgeDrones and NDFP, i.e., PDFP and PDNP achieve 9.2%, 14.6%; 14%, 18%; and 5%, 8.25%



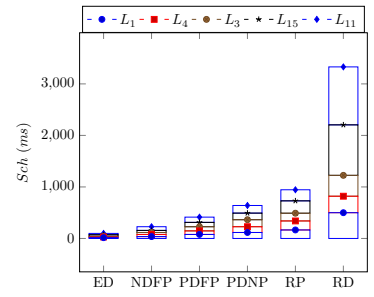
(a) CPU utilization across assigned locations



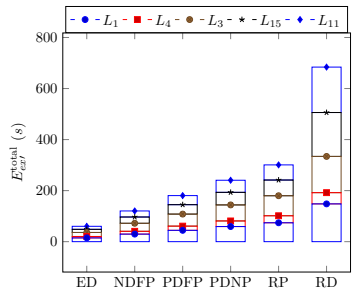
(b) Memory utilization across assigned locations



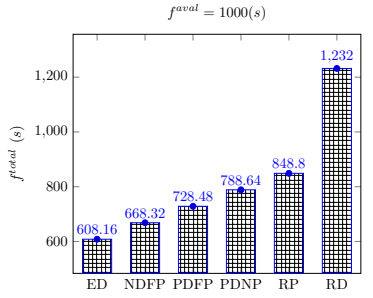
(c) Actual resource usage across assigned locations



(d) Actual multi-Job scheduling time across assigned locations



(e) Actual multi-Job execution time across assigned locations



(f) Actual total flight travel time for Drone-3's mission

Figure 6: Activities and utilities of Drone-3's mission.

less than EdgeDrones across Drone-2, Drone-3 and Drone-4 missions, respectively. However, RP and RD schemes performed poorly mainly due to their resource under-utilization, i.e., both RP and RD schieve an average CPU utilization of 19.8%, 28.2%; 25.8%, 37.4%; and 18.25%, 44.25% less than EdgeDrones across Drone-2, Drone3 and Drone4 missions, respectively.

Figs. 4(b), 5(b), 6(b) and 7(b) compares the *Memory Utilization* of EdgeDrones with those of the four baseline schemes and the random approach. Note that all the tasks executed across the assigned locations of the four drones are CPU intensive tasks, hence, the memory utilization across the locations are lower compared to the CPU utilizations. Nevertheless, EdgeDrones outperformes all the benchmark schemes, including the random approach. For example, across Drone-1's activities at its assigned locations, EdgeDrones is superior in achieving higher memory utilization with 3.5%, 6.5%, 6.5%, 12% and 18% more

memory utilizations than NDFP, PDFP, PDNP, RP and RD, respectively. It achieves 23.4% average memory utilization across Drone-2's activities, which surpasses NDFP, PDFP, PDNP, RP and RD by 1%, 2.2%, 3.6%, 4.8% and 7.2%, respectively. EdgDrones also outperforms all the baseline schemes and random approach across Drone-3 and Drone-4 assigned locations. It achieves an average of 30.2% and 17% memeory utilization at Drone-3 and Drone-4 activities, respectively. NDFP came second at both drones memory utilization (with 1.6% and 1.25% less compared to EdgeDrones at Drone-3 and Drone-4, respectively). PDFP and PDNP came third and forth, respectively. PDFP achieves 4.6% and 1.5% less in memory utilization compared to EgeDrones at Drone-3 and Drone-4 activities, respectively, while PDNP achieves 5.8% and 1.75% less compared to EgeDrones at Drone-3 and Drone-4 activities, respectively. RP and RD performance are the worst compared to all other schemes. In particular, the random approach (RD) achieves an average of 17.6% and 11.25% memory utilization across the assigned locations of Drone-3 and Drone-4, respectively, which is 12.6% and 5.75% less than EdgeDrones' achievement across the two drones activitites. RP on the other hand, achieves an average of 8.6% less compared to EdgeDrones across Drone-3 assigned locations, and an average of 3.5% less than EdgeDrones across Drone-4 assigned locations.

Figs. 4(c), 5(c), 6(c) and 7(c) compares the actual resource usage $\mathcal{DR}_{iARU}^{\langle c,m\rangle}$ of EdgeDrones with those of the four baseline schemes and the random approach. It can be seen that solving the optimization (45) is directly linked to minimize the ARU (49), by packing or co-locating tasks firmly on available resources. Hence, it can be seen that EdgeDrones consumes the fewest resources across the integrated drones activities with NPFP as the very close second best, while Random uses all the resources across almost all the drones activities with RP as the second worst. PDFP ranks in the middle, in terms of resource usage across the drones activities. Again, EdgeDrones and NDFP are superior than PDFP, PDNP, RD and Random, and they achieve the highest and close second highest resource utilization across the integrated drones activities, respectively. For example, across Drone-1's assigned locations, EdgeDrone uses the fewest resources amounting to an average of 71% compared to NDFP, PDFP, PDNP and RP, which use 4%, 8%, 8%, 16.5% more than EdgeDrones, respectively. However, RD uses all available resources, i.e., 100%, due to its inability to co-locate tasks on nodes. Across Drone-2, Drone-3 and Drone-4's assgined locations, it can also be seen that EdgeDrones uses fewer resources, up to an average of 37% less compared to other baseline schemes and the random approach.

Three other key metrics are the actual multi-tasks/job scheduling time $\sum_{J\in\mathbb{J}}\sum_{z=1}^{m}\sum_{i=1}^{k_z} S_{ch_{z_i}}/k_z$, where m is the number of scheduling units, k_z is the number of tasks within the z-th scheduling unit, and more importantly, the actual multi-tasks/jobs execution time $E_{ex'}^{\text{total}}$ and total flight travel time f^{total}. Figs. (4(d), 5(d), 6(d), (7(d)); (4(e), 5(e), 6(e), 7(e)); and (4(f), 5(f), 6(f), 7(f)) compares the actual multi-tasks/jobs scheduling time, multi-tasks/job execution time and total flight travel time of EdgeDrones with those of the four benchmarks and random approach, respectively. The results show that EdgeDrones is the best, NDFP is the second best, and PDFP is the third best, PDNP is forth best, while RP and Random is the worst and RP the second worst, in terms of actual task scheduling times, actual task execution times and drone's total flight time. The superior performance of EdgeDrones over the other benchmarks is overwhelmingly clear.

5.2.3. Performance comparison in individual Drones

Figs. 4 ∼ 7 show the performance of the schemes in terms of resource utilization, actual resource usage, actual task scheduling times, actual task execution times and actual flight total travel times across the integrated drones. We now delve into the individual drone to examine the performance of all the schemes.

`Drone-1` is attached with three `Huawei AR502H Series` edge devices, with total resource capacity of 12 Cores and 6 GiB for CPU and memory, respectively. The entire weight of the devices is ≈3.3kg. Its assigned locations $\mathcal{DR}_1^{assign} = \{L_{12}, L_{13}\}$; its route $\mathcal{DR}_1^{route} = L_0 \rightarrow L_{12} \rightarrow L_{13} \rightarrow L_0$; and its flight time availability $f^{aval} = 700(s)$. We deploy 9 jobs with a total of 36 tasks, where the job has a task dependency depth γ (1, 5]. Utilizing the gang scheduling strategy, EdgeDrones co-shedules and co-locates all the 9 jobs at a time in the attached edge devices as possible to minimize the overall used nodes. These jobs are tightly co-located, which enables dependent tasks to communicate and share data effectively. As a result, EdgeDrones achieves the fastest scheduling time and execution time compared to NDFP, PDFP, PDNF, RP and the random approach. In addition, EdgeDrones only uses an average 71% of resources to execute the jobs. Using the same resource capacity, NPFP, PDFP, PDNP and RP utilize an average of 75%, 79%, 79% and 87.5% of the resources, respectively, as shown in Fig 4(c). The random approach uses all available resources. It is observed that EdgeDrones is 1.7 times and 2 times faster than the second best NDFP in both the scheduling time and execution time, respectively. EdgeDrones is more than 3 times and more than 3 times faster than PDFP as well as more than 5 times and more than 4 times faster than PDNP in the scheduling time and execution time, as shown in Figs 4(d) and 4(e), respectively. EdgeDrones is 6 times and 5 times faster than the RP, as well as 47 times and 18 times faster than the random approach in the scheduling and execution times, respectively. The most important is the total flight travel time f^{total} of Drone-1, such that if along the drone's mission f^{total} becomes greater than its flight time availability f^{aval}, then it might lead to loss of job or mission failure. Recall that $f^{total} = \sum_{i=0}^{n}\sum_{j=i+1}^{m} f^{L_{i,j}} + f_j^{hover}$, where $\sum_{j=0}^{m} f_j^{hover} = E_{ex'}^{\text{total}}$. Hence, EdgeDrones is bale to quickly schedule and execute all the 9 jobs at each assigned location, resulting to a successful mission, and the fastest f^{total} (upto 2.7 times faster) compared to the four baseline schems and the random approach, as shown in Fig. 4(f). The random

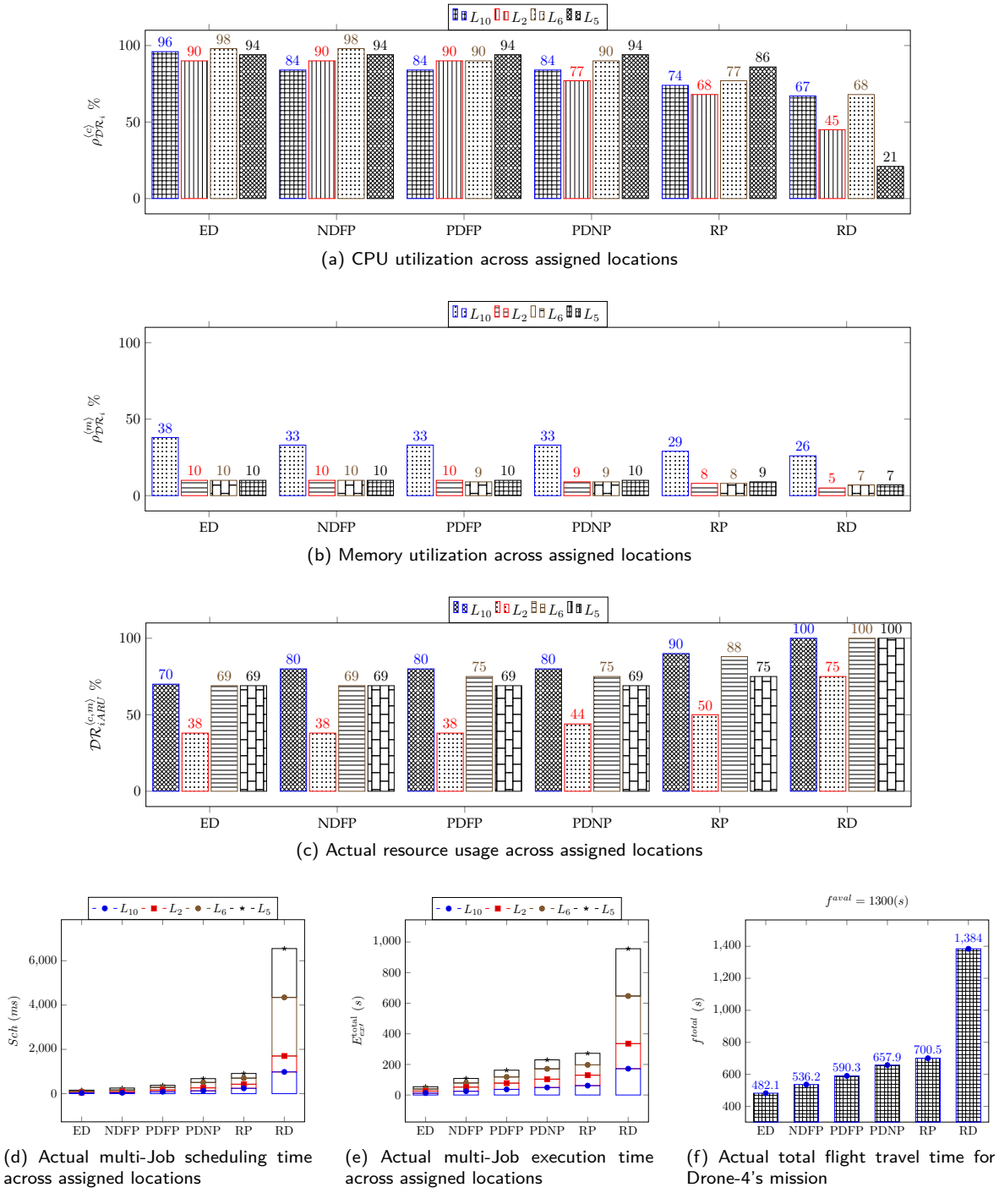


Figure 7: Activities and utilities of Drone-4's mission.

approach could not successfully schedule and execute all the jobs within the drone's f^{aval}, thereby leading to a failed mission.

Like `Drone-1`, `Drone-2` is attached with two `HIVECELL` portable edge devices with total weight of ≈2.72kg, and total resource capacity of 12 Cores and 16 GiB for CPU and memory, respectively. Its assigned locations $\mathcal{DR}_1^{assign} = \{L_7, L_8, L_9, L_{14}, L_{16}\}$; it has to visit and execute tasks on route: $\mathcal{DR}_1^{route} = L_0 \rightarrow L_9 \rightarrow L_{14} \rightarrow L_{16} \rightarrow L_8 \rightarrow L_7 \rightarrow L_0$; and its flight time availability $f^{aval} = 1200(s)$. Here, we deploy a total of $\mathbb{J} = 17$, where each $J \in \mathbb{J}$ has a task dependency in the range of $(1, 4]$. The total number of tasks in $\sum \mathbb{J}$ is 76. We ensure that the attached edge resources are fully utilized by co-locating the jobs tightly on them. As discussed earlier, application container provides isolation to co-located tasks, thereby eliminating interference and resource contentions in the cluster. A single node is capable of running several containerized tasks, given that available

resources are sufficient. In this drone's activities, Edgedrones consume an average of 5.2% fewer resources than NDFP, an average of 6.8%, 11.8%, 16.8% and 28.4% fewer resources than PDFP, PDNP, RP and Random. EdgeDrones, also gain an average of upto 28.2% higher CPU utilization over NDFP, PDFP, PDNP, RP and Random (as shown in Fig. 5(a)), as well as an average of upto 7.2% higher memory utilization than NDFP, PDFP, PDNP, RP and Random, as shown in Fig. 5(b). More significantly, EdgeDrones is 2, 3.7, 5, 7.4 and 42.5 times faster in the scheduling time than NDFP, PDFP, PDNP, RP and the random approach respectively, while it is 2, 3, 4, 5 and 14.8 times faster in the execution time than NDFP, PDFP, PDNP, RP and the random approach, respectively across the assigned locations. Although all the schemes, except for the random approach were able to complete their task within the drone's flight time availability, nonetheless, EdgeDrones achieves the fasters mission completion time, which is much more less than the drone's flight time availability. It can be seen in Fig. 5(f) that EdgeDrones is upto 2.7 times faster than other schemes.

`Drone-3` is attached with one `HIVECELL` and one `Huawei AR502H Series` portable edge devices with total weight of ≈2.46kg. Its total resource capacity is 10 Cores and 8 GiB of CPU and memory, respectively. Its assigned locations $\mathcal{DR}_1^{assign} = \{L_1, L_3, L_4, L_{11}, L_{15}\}$; flight travel route $\mathcal{DR}_1^{route} = \{L_0 \rightarrow L_1 \rightarrow L_4 \rightarrow L_3 \rightarrow L_{15} \rightarrow L_{11} \rightarrow L_0\}$; and flight time availability $f^{aval} = 900(s)$. In this cluster, we deploy $\mathbb{J} = 12$ in total of 56 tasks, where each $J \in \mathbb{J}$ has a task dependency depth γ range (2, 5]. Across this drone's activities, EdgeDrones achieve reduced $\mathcal{DR}_{iARU}^{\langle c,m \rangle}$ by 3.8%, 9.4%, 13%, 20.4% and 37% compared with NDFP, PDFP, PDNP, RD and Random, respectively (as shown in Fig. 6(c)). EdgeDrones achieve 4.8%, 14%, 18%, 25.8% and 37.4% higher CPU utilization as well as 1.6%, 4.6%, 5.8%, 8.6% and 12.6% higher memory utilization compared to NDFP, PDFP, PDNP, RD and Random, respectively. In terms of scheduling, EdgeDrones is about 2.3 times, 4.2 times, 6.5 times, 9.6 times and 34 times faster than NDFP, PDFP, PDNP, RD and Random, respectively (as shown in Fig. 6(d)). It achieves approximately 2 times, 3 times, 4 times, 5 times and 11 times faster execution times than NDFP, PDFP, PDNP, RD and Random, respectively (as shown in Fig. 6(e)). Not surprisingly, Random has the worst scheduling time and execution time performance, resulting to incomplete mission (since its $f^{total} > f^{aval}$). On the other hand, EdgeDrones achieves the fasters mission completion time, which is upto 1.4 times faster than NDFP, PDFP, PDNP and RD, as shown in Fig. 6(f).

`Drone-4` is attached with `Azure Stack Edge mini` memory intensive edge device, with resource capacity of 16 Cores and high memory capacity of 48 GiB. It is four locations $\mathcal{DR}_1^{assign} = \{L_2, L_5, L_6, L_{10}\}$, where there are total of 16 jobs made up of 70 tasks to be executed. Its flight travel route $\mathcal{DR}_1^{route} = \{L_0 \rightarrow L_{10} \rightarrow L_2 \rightarrow L_6 \rightarrow L_5\}$, and flight time availability $f^{aval} = 1200(s)$. It is observed that EdgeDrones consumes the fewest resources at an average of 61.5%, followed by NDFP at 64%. PDFP consumes an average of 65.5%, PDNP consumes an average of 67%, RP consumes an average 75.75% of the resources, while the Random approach uses all the available resources at $L_{10}, L_6 and L_5$ locations, but consumes an average of 93.75% of resources across the assigned locations. EdgeDrones also achieves 3%, 5%, 8.25%, 18.25% and 44.25% higher CPU utilization over NDFP, PDFP, PDNP, RP and Random, respectively (as shown in Fig. 7(a)). Note the edge device attached to this drone is memory intensive, i.e., it has huge memory capacities compared to the memory resource request of jobs at the assigned locations. Therefore, the jobs can only consume few such capacities, as shown in Fig. 7(b). In terms of scheduling time, EdgeDrones is approximately 1.6 times, 1.5 times, 4.3 times, 5.8 times and 41.7 times faster than NDFP, PDFP, PDNP, RP and random respectively (Fig. 7(d)). In terms of execution time, EdgeDrones is about 2 times, 3 times, 4.2 times, 5 times and 17.6 times faster than NDFP, PDFP, PDNP, RP and the random approach respectively (Fig. 7(e)). Importantly, Drone-4 completed its mission with the fastest time under EdgeDrones strategy, with up to 1.5 times faster compared to the baseline schemes, as shown in Fig. 7(f)).

5.3. Discussion

Overall, EdgeDrones has demonstrated better performance in an integrated edge computing system. It has consistently outperform existing schemes (NDFP, PDFP, PDNP, RP and Random) by achieving faster scheduling times and excution times, while using fewer resources. Most importantly, effective multi-tasks scheduling and execution of EdgeDrones across the locations, enables faster tasks response times and mission completion times. EdgeDrones achievements is attributed to its effective orchestration strategy, gang-deployment and co-location of multi-jobs, which allows inter-dependent tasks within each job to communicate and share data faster. Such fast execution is crucial for modern applications to perform better. The existing schemes do not consider task's dependecies or multi-tasks co-location, leading to limited edge resource wastage through under utilization, as well as causing execution delay.

6. Conclusions

This paper has presented a novel Multi-Location Capacitated Mission Scheduling Problem (MLCMSP) that selects suitable drones and co-schedules their flight routes with the least total distance to visit and execute tasks at the target locations. We proposed an intelligent multi-dependent tasks orchestration scheme called EdgeDrones, a variant bin-packing optimization approach through gang-scheduling of multi-dependent tasks, that co-schedules and co-locate tasks firmly on available nodes, so as to avoid resource wastage. Evaluations using real world workloads from Alibaba clusters, shows that EdgeDrones is superiror compared to the baseline schemes. Importantly, EdgeDrones is able to avoid loss of jobs in aerial edge computing missions. In our future research, we can further integrate cost models to MLCMSP by assigning an operational cost per drone's mission, and

convert the MLCMSP into a profit maximization problem. In addition, we can also deploy on-premise (fog-based computing) alongside with the drones (aerial-based computing) to form a hybrid deployment.

CRediT authorship contribution statement

Uchechukwu Awada: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Writing original draft, Writing review & editing, Visualization. **Jiankang Zhang:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Writing original draft, Writing review & editing, Visualization, Supervising, Project administration, Funding acquisition. **Sheng Chen:** Conceptualization, Methodology, Software, Validation, Investigation, Resources, Visualization, Supervising. **Shuangzhi Li:** Conceptualization, Methodology, Formal analysis, Supervising, Funding acquisition. **Shouyi Yang:** Conceptualization, Methodology, Formal analysis, Supervising, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

The financial support of the National Natural Science Foundation of China under grants 61571401 and 61901416 (part of the China Postdoctoral Science Foundation under grant 2021TQ0304), and the Innovative Talent of Colleges and the University of Henan Province under grant 18HASTIT021 are gratefully acknowledged.

References

[1] M. U. Bokhari, Q. Makki, Y. K. Tamandani, A survey on cloud computing, in: V. B. Aggarwal, V. Bhatnagar, D. K. Mishra (Eds.), Big Data Analytics, Springer Singapore, Singapore, 2018, pp. 149–164.

[2] S. Li, H. Liu, W. Li, W. Sun, Optimal cross-layer resource allocation in fog computing: A market-based framework, Journal of Network and Computer Applications 209 (2023) 103528. doi:https://doi.org/10.1016/j.jnca.2022.103528.

[3] H. Mei, K. Wang, D. Zhou, K. Yang, Joint trajectory-task-cache optimization in uav-enabled mobile edge networks for cyber-physical system, IEEE Access 7 (2019) 156476–156488. doi:10.1109/ACCESS.2019.2949032.

[4] L. Zhang, J. Chakareski, Uav-assisted edge computing and streaming for wireless virtual reality: Analysis, algorithm design, and performance guarantees, IEEE Transactions on Vehicular Technology 71 (3) (2022) 3267–3275. doi:10.1109/TVT.2022.3142169.

[5] M. Laroui, H. Ibn-Khedher, H. Moungla, H. Afifi, Autonomous uav aided vehicular edge computing for service offering, in: 2021 IEEE Global Communications Conference (GLOBECOM), 2021, pp. 1–6. doi:10.1109/GLOBECOM46510.2021.9685525.

[6] A. Koubaa, A. Ammar, A. Kanhouch, Y. AlHabashi, Cloud versus edge deployment strategies of real-time face recognition inference, IEEE Transactions on Network Science and Engineering 9 (1) (2022) 143–160. doi:10.1109/TNSE.2021.3055835.

[7] L. U. Khan, I. Yaqoob, N. H. Tran, S. M. A. Kazmi, T. N. Dang, C. S. Hong, Edge-computing-enabled smart cities: A comprehensive survey, IEEE Internet of Things Journal 7 (10) (2020) 10200–10232. doi:10.1109/JIOT.2020.2987070.

[8] D. Callegaro, S. Baidya, M. Levorato, A measurement study on edge computing for autonomous uavs, in: Proceedings of the ACM SIGCOMM 2019 Workshop on Mobile AirGround Edge Computing, Systems, Networks, and Applications, MAGESys'19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 29–35. doi:10.1145/3341568.3342109.

[9] Q. Chen, H. Zhu, L. Yang, X. Chen, S. Pollin, E. Vinogradov, Edge computing assisted autonomous flight for uav: Synergies between vision and communications, IEEE Communications Magazine 59 (1) (2021) 28–33. doi:10.1109/MCOM.001.2000501.

[10] G. Faraci, C. Grasso, G. Schembra, Fog in the clouds: Uavs to provide edge computing to iot devices, ACM Trans. Internet Technol. 20 (3) (aug 2020). doi:10.1145/3382756.

[11] U. Awada, J. Zhang, S. Chen, S. Li, Air-to-air collaborative learning: A multi-task orchestration in federated aerial computing, in: 2021 IEEE 14th International Conference on Cloud Computing (CLOUD), 2021, pp. 671–680. doi:10.1109/CLOUD53861.2021.00086.

[12] U. Awada, J. Zhang, S. Chen, S. Li, Airedge: A dependency-aware multi-task orchestration in federated aerial computing, IEEE Transactions on Vehicular Technology (2021) 1–1doi:10.1109/TVT.2021.3127011.

[13] S. Sanyal, K. Roy, Neuro-ising: Accelerating large scale travelling salesman problems via graph neural network guided localized ising solvers, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2022) 1–1doi:10.1109/TCAD.2022.3164330.

[14] J. Shi, J. Sun, Q. Zhang, K. Ye, Homotopic convex transformation: A new landscape smoothing method for the traveling salesman problem, IEEE Transactions on Cybernetics 52 (1) (2022) 495–507. doi:10.1109/TCYB.2020.2981385.

[15] A. Khochare, Y. Simmhan, F. B. Sorbelli, S. K. Das, Heuristic algorithms for co-scheduling of edge analytics and routes for uav fleet missions, in: IEEE INFOCOM 2021 - IEEE Conference on Computer Communications, 2021, pp. 1–10. doi:10.1109/INFOCOM42981.2021.9488740.

[16] B. Liu, W. Zhang, W. Chen, H. Huang, S. Guo, Online computation offloading and traffic routing for uav swarms in edge-cloud computing, IEEE Transactions on Vehicular Technology 69 (8) (2020) 8777–8791. doi:10.1109/TVT.2020.2994541.

[17] U. Awada, J. Zhang, Edge federation: A dependency-aware multi-task dispatching and co-location in federated edge container-instances, in: 2020 IEEE International Conference on Edge Computing (EDGE), 2020, pp. 91–98. doi:10.1109/EDGE50951.2020.00021.

[18] X. Cao, G. Tang, D. Guo, Y. Li, W. Zhang, Edge federation: Towards an integrated service provisioning model, IEEE/ACM Transactions on Networking 28 (3) (2020) 1116–1129. doi:10.1109/TNET.2020.2979361.

[19] C. Cheng, L. Li, J. Wang, F. Gu, The design and implementation of secure distributed image classification reasoning system for heterogeneous edge computing, in: 2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2021, pp. 250–257. doi:10.1109/TrustCom53373.2021.00049.

[20] A. Graziosi, G. Iannello, V. Lapadula, M. Merone, M. Sabatini, L. Vollero, Edge computing optimization method. analyzed task: crowd counting, in: 2021 IEEE International Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0IoT), 2021, pp. 397–401. doi:10.1109/MetroInd4.0IoT51437.2021.9488437.

[21] C. Shu, Z. Zhao, Y. Han, G. Min, H. Duan, Multi-user offloading for edge computing networks: A dependency-aware and latency-optimal approach, IEEE Internet of Things Journal 7 (3) (2020) 1678–1689. doi:10.1109/JIOT.2019.2943373.

[22] J. Liu, H. Shen, Dependency-aware and resource-efficient scheduling for heterogeneous jobs in clouds, in: 2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 2016, pp. 110–117. doi:10.1109/CloudCom.2016.0032.

[23] J. Lee, H. Ko, J. Kim, S. Pack, Data: Dependency-aware task allocation scheme in distributed edge clouds, IEEE Transactions on Industrial Informatics 16 (12) (2020) 7782–7790. doi:10.1109/TII.2020.2990674.

[24] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. Chan, K. K. Leung, Dynamic service migration and workload scheduling in edge-clouds, Performance Evaluation 91 (2015) 205–228, special Issue: Performance 2015. doi:https://doi.org/10.1016/j.peva.2015.06.013.

[25] L. Tong, Y. Li, W. Gao, A hierarchical edge cloud architecture for mobile computing, in: IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications, 2016, pp. 1–9. doi:10.1109/INFOCOM.2016.7524340.

[26] T.-P. Pham, J. J. Durillo, T. Fahringer, Predicting workflow task execution time in the cloud using a two-stage machine learning approach, IEEE Transactions on Cloud Computing 8 (1) (2020) 256–268. doi:10.1109/TCC.2017.2732344.

[27] F. Nadeem, D. Alghazzawi, A. Mashat, K. Faqeeh, A. Almalaise, Using machine learning ensemble methods to predict execution time of e-science workflows in heterogeneous distributed systems, IEEE Access 7 (2019) 25138–25149. doi:10.1109/ACCESS.2019.2899985.

[28] Z. Ning, Y. Yang, X. Wang, L. Guo, X. Gao, S. Guo, G. Wang, Dynamic computation offloading and server deployment for uav-enabled multi-access edge computing, IEEE Transactions on Mobile Computing (2021) 1–1doi:10.1109/TMC.2021.3129785.

[29] D. Wang, J. Tian, H. Zhang, D. Wu, Task offloading and trajectory scheduling for uav-enabled mec networks: An optimal transport theory perspective, IEEE Wireless Communications Letters 11 (1) (2022) 150–154. doi:10.1109/LWC.2021.3122957.

[30] X. Chen, Y. Bi, G. Han, D. Zhang, M. Liu, H. Shi, H. Zhao, F. Li, Distributed computation offloading and trajectory optimization in multi-uav-enabled edge computing, IEEE Internet of Things Journal 9 (20) (2022) 20096–20110. doi:10.1109/JIOT.2022.3175050.

[31] J. Ren, H. Wang, T. Hou, S. Zheng, C. Tang, Federated learning-based computation offloading optimization in edge computing-supported internet of things, IEEE Access 7 (2019) 69194–69201. doi:10.1109/ACCESS.2019.2919736.

[32] T. Khan, W. Tian, G. Zhou, S. Ilager, M. Gong, R. Buyya, Machine learning (ml)-centric resource management in cloud computing: A review and future directions, Journal of Network and Computer Applications 204 (2022) 103405. doi:https://doi.org/10.1016/j.jnca.2022.103405.

[33] M. H. Hilman, M. A. Rodriguez, R. Buyya, Task runtime prediction in scientific workflows using an online incremental learning approach, in: 2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC), 2018, pp. 93–102. doi:10.1109/UCC.2018.00018.

[34] W. Xiao, R. Bhardwaj, R. Ramjee, M. Sivathanu, N. Kwatra, Z. Han, P. Patel, X. Peng, H. Zhao, Q. Zhang, F. Yang, L. Zhou, Gandiva: Introspective cluster scheduling for deep learning, in: Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation, OSDI'18, USENIX Association, USA, 2018, pp. 595–610.

[35] S. Venkataraman, Z. Yang, M. Franklin, B. Recht, I. Stoica, Ernest: Efficient performance prediction for large-scale advanced analytics, in: Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation, NSDI'16, USENIX Association, USA, 2016, pp. 363–378.

[36] Y. Peng, Y. Bao, Y. Chen, C. Wu, C. Guo, Optimus: An efficient dynamic resource scheduler for deep learning clusters, in: Proceedings of the Thirteenth EuroSys Conference, EuroSys '18, Association for Computing Machinery, New York, NY, USA, 2018. doi:10.1145/3190508.3190517.

[37] G. Zhou, R. Wen, W. Tian, R. Buyya, Deep reinforcement learning-based algorithms selectors for the resource scheduling in hierarchical cloud computing, Journal of Network and Computer Applications 208 (2022) 103520. doi:https://doi.org/10.1016/j.jnca.2022.103520.

[38] C. Delimitrou, C. Kozyrakis, Quasar: Resource-efficient and qos-aware cluster management, SIGPLAN Not. 49 (4) (2014) 127–144. doi:10.1145/2644865.2541941.

[39] R.-A. Cherrueau, A. Lebre, D. Pertin, F. Wuhib, J. M. Soares, Edge computing resource management system: a critical building block! initiating the debate via OpenStack, in: USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18), USENIX Association, Boston, MA, 2018.

[40] V. S. Marco, B. Taylor, B. Porter, Z. Wang, Improving spark application throughput via memory aware task co-location: A mixture of experts approach, in: Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference, Middleware'17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 95–108. doi:10.1145/3135974.3135984.

[41] Y. Li, D. Sun, B. C. Lee, Dynamic colocation policies with reinforcement learning, ACM Trans. Archit. Code Optim. 17 (1) (mar 2020). doi:10.1145/3375714.

[42] U. Awada, A. Barker, Resource efficiency in container-instance clusters, in: Proceedings of the Second International Conference on Internet of Things, Data and Cloud Computing, ICC '17, Association for Computing Machinery, New York, NY, USA, 2017. doi:10.1145/3018896.3056798.

[43] U. Awada, A. Barker, Improving resource efficiency of container-instance clusters on clouds, in: Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid '17, IEEE Press, 2017, pp. 929–934. doi:10.1109/CCGRID.2017.113.

[44] Y. Liu, S. Wang, Q. Zhao, S. Du, A. Zhou, X. Ma, F. Yang, Dependency-aware task scheduling in vehicular edge computing, IEEE Internet of Things Journal 7 (6) (2020) 4961–4971. doi:10.1109/JIOT.2020.2972041.

[45] Z. Han, H. Tan, S. H.-C. Jiang, X. Fu, W. Cao, F. C. Lau, Scheduling placement-sensitive bsp jobs with inaccurate execution time estimation, in: IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, 2020, pp. 1053–1062. doi:10.1109/INFOCOM41043.2020.9155445.

[46] R. Grandl, G. Ananthanarayanan, S. Kandula, S. Rao, A. Akella, Multi-resource packing for cluster schedulers, in: Proceedings of the 2014 ACM Conference on SIGCOMM, SIGCOMM'14, Association for Computing Machinery, New York, NY, USA, 2014, pp. 455–466. doi:10.1145/2619239.2626334.

[47] Z. Hu, J. Tu, B. Li, Spear: Optimized dependency-aware task scheduling with deep reinforcement learning, in: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), 2019, pp. 2037–2046. doi:10.1109/ICDCS.2019.00201.

[48] R. Grandl, S. Kandula, S. Rao, A. Akella, J. Kulkarni, Graphene: Packing and dependency-aware scheduling for data-parallel clusters, in: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, OSDI'16, USENIX Association, USA, 2016, pp. 81–97.

[49] S. Rampersaud, D. Grosu, Sharing-aware online virtual machine packing in heterogeneous resource clouds, IEEE Transactions on Parallel and Distributed Systems 28 (7) (2017) 2046–2059. doi:10.1109/TPDS.2016.2641937.

[50] J. Guo, Z. Chang, S. Wang, H. Ding, Y. Feng, L. Mao, Y. Bao, Who limits the resource efficiency of my datacenter: An analysis of alibaba datacenter traces, in: 2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS), 2019, pp. 1–10. doi:10.1145/3326285.3329074.

[51] H. Wu, W. Zhang, Y. Xu, H. Xiang, T. Huang, H. Ding, Z. Zhang, Aladdin: Optimized maximum flow management for shared production clusters, in: 2019 IEEE International Parallel and Distributed

Processing Symposium (IPDPS), 2019, pp. 696–707. doi:10.1109/IPDPS.2019.00078.

[52] H. Liao, X. Li, D. Guo, W. Kang, J. Li, Dependency-aware application assigning and scheduling in edge computing, IEEE Internet of Things Journal 9 (6) (2022) 4451–4463. doi:10.1109/JIOT.2021.3104015.

Uchechukwu Awada is currently working toward the PhD degree in the School of Information Engineering, Zhengzhou University, China. His current research interests include edge computing, cloud computing, distributed systems, IoT and wireless communications. He is a student member of the ACM.

Jiankang Zhang is a Senior Lecturer at Bournemouth University. Prior to joining in Bournemouth University, he was a senior research fellow at University of Southampton, UK. Dr Zhang was a lecturer from 2012 to 2013 and then an associate professor from 2013 to 2014 at Zhengzhou University. His research interests are in the areas of aeronautical communications, aeronautical networks, evolutionary algorithms and edge computing. He serves as an Associate Editor for IEEE ACCESS.

Sheng Chen received his BEng degree from the East China Petroleum Institute, Dongying, China, in 1982, and his PhD degree from the City University, London, in 1986, both in control engineering. In 2005, he was awarded the higher doctoral degree, Doctor of Sciences (DSc), from the University of Southampton, Southampton, UK. From 1986 to 1999, He held research and academic appointments at the Universities of Sheffield, Edinburgh and Portsmouth, all in UK. Since 1999, he has been with the School of Electronics and Computer Science, the University of Southampton, UK, where he holds the post of Professor in Intelligent Systems and Signal Processing. Dr Chen's research interests include adaptive signal processing, wireless communications, modeling and identification of nonlinear systems, neural network and machine learning, intelligent control system design, evolutionary computation methods and optimization. He has published over 600 research papers. Professor Chen has 16,700+ Web of Science citations with h-index 57 and 33,200+ Google Scholar citations with h-index 77. Dr. Chen is a Fellow of the United Kingdom Royal Academy of Engineering, a Fellow of Asia-Pacific Artificial Intelligence Association and a Fellow of IET. He is one of the original ISI highly cited researcher in engineering (March 2004).

Shuangzhi Li received the B.S. and Ph.D. degrees from the School of Information Engineering, Zhengzhou University, Zhengzhou, China, in 2012 and 2018, respectively. From 2015 to 2017, he was a Visiting Student with the Department of Electrical and Computer Engineering, McMaster University, Canada. He is currently a Lecturer with the School of Information Engineering, Zhengzhou University, China. His research interests include noncoherent space-time coding and ultra-reliable low-latency communications.

Shouyi Yang received the Ph.D. degree from the Beijing Institute of Technology, Beijing, China, in 2002. He is currently a Full Professor with the School of Information Engineering, Zhengzhou University, Zhengzhou, China. He has authored or coauthored various articles in the field of signal processing and wireless communication. His current research interests include signal processing in communications systems, wireless communications, and cognitive radio.