IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 9 November 2024, accepted 9 December 2024, date of publication 16 December 2024,
date of current version 26 December 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3518562

RESEARCH ARTICLE

DeepAir: A Multi-Agent Deep Reinforcement Learning-Based Scheme for an Unknown User Location Problem

BARIS YAMANSAVASCILAR, ATAY OZGOVDE, (Senior Member, IEEE), AND CEM ERSOY, (Senior Member, IEEE)
Department of Computer Engineering, Bogazici University, 34342 Istanbul, Türkiye
Corresponding author: Baris Yamansavascilar (baris.yamansavascilar@std.bogazici.edu.tr)

ABSTRACT Unmanned Aerial Vehicles (UAVs) are a major component in next-generation network architecture proposals, playing a critical role in problems like dynamic capacity enhancement, user coverage, and task offloading. When smart utilization of the UAVs is missing, these proposals may require sophisticated approaches, including the deployment of additional edge servers and orchestration efforts. A typical challenge arises from the dynamic nature of real-world problems in which the required capacity should be provided at particular times when fixed infrastructure proves insufficient. One of those existing dynamic problems is the unknown user locations in an infrastructure-less environment in which users cannot connect to any communication device or computation-providing server, which is essential to task offloading in order to achieve the required quality of service (QoS). Therefore, in this study, we investigate this problem thoroughly and propose a novel deep reinforcement learning (DRL) based scheme, DeepAir. DeepAir uses four main phases including sensing, localization, resource allocation, and multi-access edge computing (MEC) to provide the corresponding QoS requirements for the offloaded tasks without violating the maximum tolerable delay. To this end, we use two types of UAVs including detector UAVs, and serving UAVs. We utilize detector UAVs as DRL agents which ensure the sensing, localization, and resource allocation phases. On the other hand, we utilize serving UAVs to provide MEC features. Our experiments show that DeepAir provides higher task success rates by deploying fewer detector UAVs in different scenarios with different numbers of users and user attraction points compared to benchmark methods. Thus, DeepAir achieves 59.65%, 86.06%, and 86.72% task success rates for 2, 4, and 6 detector UAVs, respectively, by using 12 serving UAVs, while the most successful benchmark method provides 28.62%, 41.39%, and 61.09% task success rates for the same configuration, respectively.

INDEX TERMS Deep reinforcement learning, task offloading, UAVs.

I. INTRODUCTION

The widespread utilization of cloud computing after nearly two decades has brought about many opportunities for both companies and end-users that benefit from task offloading, content caching, and resource allocation. Especially throughout its computational advantages, cloud computing has provided computing capacity, reliability, and robustness for the offloaded tasks which otherwise may not be solved on the user devices [1], [2]. However, even though it provides important opportunities, many other computing paradigms including Cloudlet [3], Edge Computing [4], [5], and Fog Computing [6] emerged in the last decade since cloud computing cannot meet the low latency requirements of novel user applications due to the wide area network delay (WAN) [7].

Among those emerged edge solutions, multi-access edge computing (MEC) [8] has become an extensively used

The associate editor coordinating the review of this manuscript and approving it for publication was Zhenbao Liu.

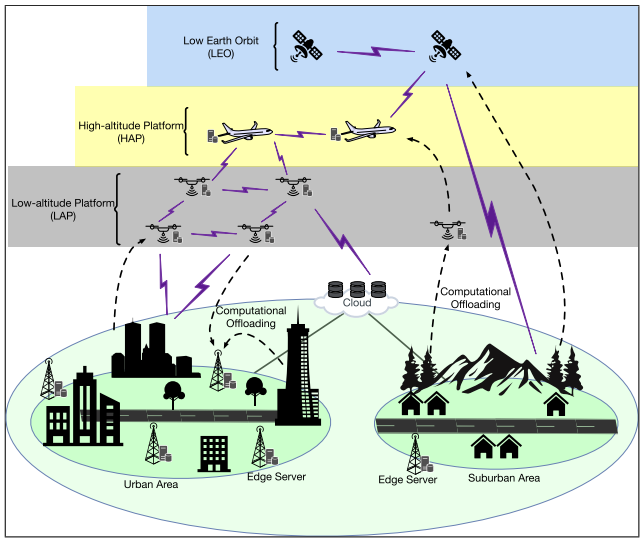


FIGURE 1. An air computing environment with different air platforms.

technology since it provides low latency and computation power for intensive tasks. Therefore, it is deployed for different application types including healthcare, video analytics, smart home, and virtual reality [9], [10]. Nevertheless, the fixed infrastructure of MEC prevents its utilization for dynamic scenarios in which the number of users/requests increases suddenly because of an event or a disaster. For example, a sport or concert event, in which there are thousands of new users, can exceed the capacity of the existing MEC infrastructure and therefore the quality of service (QoS) of users can be heavily affected as the corresponding tasks cannot be executed properly.

To meet the dynamic capacity requirements, UAVs have recently been deployed along with other air vehicles in different air platforms under varied names such as aerial radio access network (ARAN), and air computing [1], [11]. An air computing environment is depicted in Figure 1. The communication between those different air platforms throughout the corresponding air vehicles brings about new research opportunities to meet those requirements considering the application QoS and user Quality of Experience (QoE) [12]. Thus, different application profiles can benefit from various advantages of this new 3D dynamic computing paradigm.

Among those different air vehicles, UAVs are the most studied units since their deployment is easier considering their energy consumption, flying altitude, and configuration [13]. To this end, they are used for dynamic capacity enhancement in environments in which the fixed capacity would not be sufficient to meet the application requirements of an increasing number of users. Therefore, this feature solves variety of problems such as communication in a disaster site, and enhancing services in infrastructure-less environments [14], [15]. Moreover, their deployment provides significant vertical networking opportunities such as high mobility support, coverage, latency, and two-way task offloading [16]. As a result, the requirements of users living in urban, suburban, and rural areas can be met efficiently through these vertical networking opportunities.

There are many studies in which UAVs are used as flying computational units to assist either deployed edge servers or are solely deployed to enhance network capacity for task offloading [15], [17]. Since the battery and CPU capacity of the end users would not be sufficient to process the corresponding tasks, UAVs therefore can be used as a flying edge server. However, since there are many different scenarios, various methods and algorithms are developed to meet the service requirements. Deep Reinforcement Learning (DRL) is one of those methods that is applied in the literature since the traditional heuristic methods and convex optimization cannot solve the corresponding dynamic problems [18]. To this end, DRL solutions would be used for trajectory optimization, energy-efficient offloading, UAV placement, and generic task offloading.

A. MOTIVATIONS

User connectivity is a primary issue in accessing the related resources for task offloading and service differentiation. However, in order to provide a required service, the corresponding technology such as edge or UAV should first detect the user, and then the connection should be established. However, in an environment which is infrastructure-less and user locations are unknown, providing those services is a crucial technical challange.

Even though UAVs are used in many different cases, their utilization in an infrastructure-less environment in which users cannot connect to any cellular operator, edge/cloud server, or satellite has not been investigated properly. That environment can be a disaster site, wilderness, or a natural area that is open to visitors. In such an environment, the detection of user locations, localization, and then the measurement of required capacity for user tasks are major issues to ensure the necessary service. Therefore, in this study, we focus on an environment in which there are users at unknown locations where we locate them through a novel method using DRL. Afterwards, as the second step, we estimate the corresponding requirements of the connected users at detected locations and decide how many UAVs are needed in those corresponding areas.

On the other hand, as detailed by Bai et al. [19], DRL-based UAV studies have four main categories, and most of the studies in the literature focus on a subset of those categories. Therefore, in addition to the challenging environment, providing a solution for each of those categories is our overall goal.

B. CONTRIBUTIONS

In this study, we develop a DRL-based scheme, DeepAir, which takes unconnected users' emitted RSSI signals into account as a reward and then finds the corresponding user

TABLE 1. List of abbreviations.

Notation	Description
CF	Community Flying
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
MDP	Markov Decision Process
MEC	Multi-Access Edge Computing
QoS	Quality of Service
QoE	Quality of Experience
UAV	Unmanned Aerial Vehicles
SLA	Service Level Agreement
WAN	Wide Area Network

attraction points in the environment. Since the number of user attraction points could not be known, DeepAir iteratively detects those points. Afterwards, based on the quality of those detected locations, users that are in the coverage can connect to agents, which we also call detector UAVs. After these phases, the corresponding user task profiles are extracted through the requests of connected users. Next, the required capacity based on those task profiles is computed, and then the necessary number of serving UAVs, which have MEC features, is measured for the detected user-concentrated areas. Thus, we increase the task success rate based on their Service Level Agreement (SLA) requirements. The main contributions of our paper are as follows.

- In DRL-based UAV studies, a subset of four categories, including (1) sensing, (2) localization, (3) resource allocation, and (4) UAV-assisted MEC, are mainly considered in most of the cases [19]. In this study, we take all of these categories into account as phases of our novel approach.
- In order to provide such a system, we develop a DRL-based multi-agent scheme, DeepAir, which is a novel approach that iteratively detects user locations in the environment using unconnected users' additive RSSI as the reward. DeepAir performs this operation iteratively, sending a single agent (detector UAV) to the environment for each iteration, since the number of user concentrated areas cannot be known.
- We examined the relationship between the detector and serving UAVs by conducting various experiments with different scenarios. As a result, the performance of coordinated utilization of sensing, localization, resource allocation, and UAV-assisted MEC phases on the overall task success rate is investigated.

The rest of this paper is organized as follows. In Section II, we elaborate on the related works including task offloading, UAVs, and DRL. We provide the system model and problem formulation in Section III. In Section IV, we introduce DeepAir providing technical and theoretic discussions. We show the experimental results in Section V. We discussed our observations through experiments in Section VI. Finally, we conclude our study in Section VII. We list the abbreviations used throughout the paper in Table 1.

II. RELATED WORKS

We surveyed the related research papers considering our DeepAir implementation in which user locations are not known and UAVs are used for sensing, localization, resource allocation, and UAV-assisted MEC phases. We conducted our research considering various high-impact journals and conferences.

A. FLIGHT TRAJECTORY STUDIES

As UAVs are flying units in a networking environment, their corresponding flight path is crucial for the system performance considering energy efficiency, and overall task offloading success. Therefore, some studies focus on finding an optimal trajectory efficiently. In [20], Wang et al. focused on the fairness-related optimization of user equipment considering geographical fairness, load, and overall energy consumption. To perform this, they developed a multi-agent deep reinforcement learning-based trajectory control algorithm based on the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm. They used the average fairness index and overall energy consumption as their success metric. Chang et al. proposed a trajectory design and resource allocation scheme based on multi-UAVs [21]. To this end, they considered the joint user association, power allocation, and trajectory design to maximize the system utility. They used a multi-agent DRL method which performs centralized learning and decentralized execution. In [25], Ning et al. proposed a UAV trajectory optimization scheme considering user-differentiated services. Their goal was to minimize both the computational costs of users and UAVs. They solved the problems, including unknown information about the corresponding services, and UAV trajectory, by using a multi-agent DRL approach. In the end, they measured the overall cost of UAVs and users, respectively, as the performance metric. Oubbati et al. proposed a DRL-based multi-UAV optimization method called DISCOUNT in order to utilize UAVs as relays in vehicular ad hoc networks (VANETs) [29]. Accordingly, they considered energy consumption, coverage, and routing performance to cover sparse areas in the network.

B. UAV-ASSISTED MEC STUDIES

Since UAVs have computational units on them, they can be used as flying edge servers. Therefore, deploying UAVs efficiently for dynamic capacity enhancement would be essential in particular locations. In [26], Hao et al. developed a DRL-based multi-UAV solution for the task offloading problem. Therefore, they investigated a UAV-assisted MEC system considering energy consumption and task delay. Shi et al. focused on the optimization of the UAV trajectories and offloading strategies of users jointly [27]. Hence, they investigated multi-UAV collaboration considering UAV-assisted MEC. To this end, they used a multi-agent DRL approach since the corresponding joint optimization requires non-convex operation. In [28], Zhang et al. investigated

TABLE 2. Comparison between related studies.

Study	Multi-UAV	Unknown User Location	Sensing	Resource Allocation	Localization	UAV-assisted MEC
[20]	✓				✓	✓
[21]	✓			✓	✓	
[22]	✓	✓	✓		✓	
[23]	✓	✓			✓	
[24]	✓	✓	✓		✓	
[25]	✓				✓	
[26]	✓			✓	✓	✓
[27]	✓				✓	✓
[28]				✓	✓	✓
[29]	✓			✓	✓	
[30]	✓			✓	✓	✓
[31]	✓			✓	✓	✓
Our study	✓	✓	✓	✓	✓	✓

UAV-assisted communications using a single UAV and multiple users. To this end, they proposed a proximal policy optimization (PPO) based DRL algorithm in order to adjust the direction, altitude, and speed of the UAV. In addition to these adjustments, they took the QoS requirements of the users into account regarding service time minimization. In [30], Yan et al. focused on an efficient task offloading strategy considering a UAV-assisted vehicular edge computing environment. Therefore, their primary objective was to reduce system delay by deploying a deep deterministic policy gradient algorithm with a long short-term memory (LSTM) network and an attention mechanism using UAVs. They optimized their approach considering UAV battery power, maximum flight speed, and communication bandwidth constraints. Gao et al. investigated the joint trajectory control and task offloading problem in UAV-assisted MEC in [31]. Moreover, they took task latency minimization, energy consumption of UAVs, and number of processed tasks into account as an optimization goal. Furthermore, they developed a DRL-based approach to solve this optimization problem by considering a fast adaptive method in which, when a trained scheme is faced with a new scenario, the corresponding DRL model could perform a rapid adaptation.

C. UAV-ASSISTED DISASTER SUPPORT STUDIES

UAVs are widely deployed in the case of natural disasters in which communication infrastructure has been destroyed. Therefore, they play a critical role in rescue operations and also in finding the location of missing people. In [22], Zhang et al. investigated user localization through UAV swarms in the case of a disaster-affected ground that has no base station. Their goals were to increase the efficiency of the task and to minimize the energy consumption of the UAVs. They proposed a multi-agent DRL approach whose initial route is based on the probability distribution map of the users. In [23], Zhang et al. investigated a multi-UAV cooperative reconnaissance and search (MCRS) scheme for the localization of static targets. Therefore, they designed a belief probability map model based on Dempster-Shafer (DS) evidence theory and then proposed a new DRL algorithm called Double Critic Deep Deterministic Policy Gradient (DCDDPG). DCDDPG takes the belief probability map into account and uses the MADDPG approach by utilizing two critic networks. They evaluated their system performance based on decreasing uncertainty, and increasing number of targets found. Chent et al. focused on energy-efficient and dynamic multi-UAV coverage control for disaster areas [24]. They developed a trace pheromone-based mechanism through the MADDPG algorithm in order to provide non-overlapping coverage. Based on reduced overlapping UAVs, they could achieve energy efficiency. They evaluated the performance of their system using average coverage rate, normalized average energy consumption, and coverage efficiency.

To the best of our knowledge, the unknown user location case has not been deeply investigated by the related studies that utilize UAVs to provide required services in an environment. Thus, we provide a novel DRL-based approach, DeepAir, which locates the users through their RSSI using UAVs iteratively. Afterwards, we compute the QoS requirements of connected users in the detected areas and provide the corresponding serving UAVs. On the other hand, our novel scheme provides sensing, resource allocation, localization, and UAV-assisted MEC phases that the related studies ensure only a subset of them. Our main differences between the related studies are given in Table 2.

III. SYSTEM MODEL AND PROBLEM FORMULATION

We consider an environment as a set of users denoted as $\mathcal{M} = \{1, 2, \ldots, M\}$, a set of serving UAVs represented as $\mathcal{N}_s = \{1, 2, \ldots, N_s\}$, and a set of detector UAVs specified as $\mathcal{N}_d = \{1, 2, \ldots, N_d\}$. Detector UAVs are used for sensing and localization of users in the environment whose locations are unknown. Serving UAVs, on the other hand, are used as a computational resource for offloaded tasks of users. In other words, serving UAVs are flying edge servers in the environment. Each UAV type has a horizontal radius, r, for communicational or computational coverage. Each user $m \in \mathcal{M}$ randomly produces a computation-intensive task $W_m = (D_m, C_m, \lambda_m, T_m^{max})$, where D_m is the size of the

TABLE 3. List of main notations.

Symbol	Definition
M	Number of users
N_s	Number of serving UAVs
N_d	Number of detector UAVs
r_{n_d}	Horizontal radius of a detector UAV
r_{n_s}	Horizontal radius of a serving UAV
W_m	Task of user m
D_m	Size of task W_m
C_m	Required CPU cycle for task W_m
λ_m	Arrival rate of task W_m
T_m^{max}	Maximum tolerable delay of task W_m
$d_n(t)$	Flight distance of UAV n at time t
$\vartheta_n(t)$	Flight angle of UAV n at time t
f_m	Computational capacity of user m
f_{n_s}	Computational capacity of serving UAV n_s
T_m^{loc}	Computation delay of user m
$T_{n_s}^{UAV}$	Computation delay of serving UAV n_s
T_t^{UAV}	Total delay including queueing and computation at serving UAV n_s
T_q	Queueing delay at serving UAV n_s
T_{m,n_s}	Transmission delay between user m and serving UAV n_s
$h_{m,n_d}(t)$	Channel gain between user m and detector UAV n_s after path loss
g_{m,n_d}	Channel power gain between user m and detector UAV n_s
d_{m,n_d}	Distance between user m and detector UAV n_d
v_{m,n_s}	Data rate (bit/sec) between user m and serving UAV n_s
$H(t)$	Cumulative signal strength on detector UAV n_s at time t
t_{w_m}	Total delay for task W_m
$\beta m, n_s$	Offloading decision variable between user m and serving UAV n_s
α_{w_m}	Task completion variable for user m

task as bits, C_m is the required CPU cycle for processing as cycle/bit, λ_m is the arrival rate as task/sec, and T_m^{max} is the maximum tolerable latency as seconds. For convenience to read, the list of main notations used in formulations is given in Table 3.

In the environment, the location of a UAV $n \in \mathcal{N}_s$ or $n \in \mathcal{N}_d$ can be denoted as $u_n(t) = [x_n(t), y_n(t), z_n(t)]$, where $x_n(t)$, $y_n(t)$, and $z_n(t)$ are the X, Y, Z coordinates at time time t. Therefore, the position of UAV n at the next time step for a horizontal flight can be expressed as

$$x_n(t+1) = x_n(t) + d_n(t) \times cos(\vartheta_n(t)) \quad (1)$$

$$y_n(t+1) = y_n(t) + d_n(t) \times sin(\vartheta_n(t)) \quad (2)$$

where $d_n(t)$ is the flight distance, and $\vartheta_n(t) \in [0, 2\pi]$ is the flight angle. Moreover, the following movement constraints should be satisfied during the horizontal flight considering the area of the environment

$$0 \leq x_n(t) \leq X_{max} \quad (3)$$

$$0 \leq y_n(t) \leq Y_{max} \quad (4)$$

where X_{max} and Y_{max} are the maximum lengths of the environment. Similarly, to avoid collision between any two UAVs, including serving and detector UAVs, minimum distance should be satisfied as follows

$$||u_i(t) - u_j(t)|| \geq d_{min} \quad \forall i, j, i \neq j \quad (5)$$

where d_{min} denotes the minimum distance. On the other hand, location of a user $m \in \mathcal{M}$ at time t is represented as $L_m(t) = [x_m(t), y_m(t), 0]$ where $x_n(t)$, and $y_n(t)$ are the X, and Y coordinates, respectively. Since users are in the ground, the Z coordinates are zero.

If a user $m \in \mathcal{M}$ processes its task locally, the corresponding local computation delay is measured as follows

$$T_m^{loc} = \frac{D_m \times C_m}{f_m} \quad (6)$$

where f_m is the computational capacity of user m as CPU cycle per second. On the other hand, if the task W_m is offloaded to a serving UAV $n_s \in \mathcal{N}_s$, the computation delay at n_s is measured as

$$T_{n_s}^{UAV} = \frac{D_m \times C_m}{f_{n_s}} \quad (7)$$

where f_{n_s} is the computational capacity of UAV n_s as CPU cycles per second. Since multiple users can be connected and therefore offload their tasks to a serving UAV, $M/M/1$ queueing model is used for the overall delay measurement at the corresponding serving UAV as

$$T_t = \frac{\sum_m^{\mathcal{M}_{sub}} D_m \times C_m}{f_{n_s} - \sum_m^{\mathcal{M}_{sub}} (\lambda_m \times D_m \times C_m)} \quad (8)$$

where $\mathcal{M}_{sub}$ denotes a subset of users concentrated in the corresponding area. Therefore, queueing delay at the serving UAV is measured as

$$T_q = T_t - T_{n_s}^{UAV} \quad (9)$$

Considering the task offloading case for task W_m, transmission delay between a user $m \in \mathcal{M}$ and a serving UAV $n_s \in \mathcal{N}_s$ is measured as

$$T_{m,n_s} = \frac{D_m}{v_{m,n_s}} \quad (10)$$

where v_{m,n_s} is the data rate between m and n_s as bit/sec. It is important to note that users connect and communicate multiple serving UAVs via orthogonal frequency-division multiple access (OFDMA). Therefore, the transmission interference between different users can be ignored.

Channel gain, which indicates to the measurement of the strength of the signal between the transmitter and receiver in wireless communication, is computed between a user m and a detector UAV $n_d \in \mathcal{N}_d$ using free-space path loss model as

$$h_{m,n_d}(t) = \frac{g_{m,n_d}}{|d_{m,n_d}(t)|^2} \quad (11)$$

where g_{m,n_d} denotes the channel power gain between the user m and detector UAV n_d, and $d_{m,n_d}(t)$ is the distance at time

t. Considering the cumulative signal strength of users at a detector UAV n_d, it is measured as

$$H(t) = \sum_{m}^{\mathcal{M}} h_{m,n_d}(t) \tag{12}$$

Since serving UAVs are sent to the locations that detector UAVs has already provided, we assume that the channel quality between users and serving UAVs is already above an acceptable threshold and therefore is not included in the formulation.

A. PROBLEM FORMULATION

In the environment, the total delay for a task of user m is computed as

$$t_{w_m} = t_{network} + t_{service} \tag{13}$$

where $t_{network}$ is the network delay, and $t_{service}$ is the service delay. The network delay is computed as

$$t_{network} = \begin{cases} T_{m,n_s}, & \text{if} \quad \beta_{m,n_s} = 1 \\ 0, & \text{if} \quad \beta_{m,n_s} = 0 \end{cases} \tag{14}$$

where β_{m,n_s} is a binary variable that indicates whether the task is offloaded to a serving UAV or not. While calculating the network delay, the propagation delay is ignored. The service delay on the other hand is computed as

$$t_{service} = \begin{cases} T_{n_s}^{UAV} + T_q, & \text{if} \quad \beta_{m,n_s} = 1 \\ T_{n_s}^{loc}, & \text{if} \quad \beta_{m,n_s} = 0 \end{cases} \tag{15}$$

In our environment, a task of user m, W_m, is successfully completed if the total delay is lower than or equal to the maximum tolerable delay of the task. Hence, we define α_{w_m} as a success variable for task completion as

$$\alpha_{w_m} = \begin{cases} 1, & \text{if} \quad t_{w_m} \leq T_m^{max} \\ 0, & otherwise \end{cases} \tag{16}$$

Thus, in this study, our main goal is to maximize the overall task success in the environment. To this end, our objective function is defined as

$$\max \sum_{m}^{\mathcal{M}} \sum_{w_m}^{W} \alpha_{w_m} \tag{17}$$

subject to

$$\sum_{n_s}^{\mathcal{N}_s} \beta_{m,n_s} \leq 1 \quad \forall m \in \mathcal{M} \qquad \text{(Constraint 1)}$$

$$d_{m,n_s} \leq r_{n_s} \quad \forall m \in \mathcal{M}, \quad \forall n_s \in \mathcal{N}_s \qquad \text{(Constraint 2)}$$

$$f_{n_s} - (\lambda_m \times D_m \times C_m) > 0 \quad \forall m, n_s \qquad \text{(Constraint 3)}$$

$$\text{Equations (3)} - \text{(5)} \qquad \text{(Constraint 4)}$$

where d_{m,n_s} is the distance between user m and serving UAV n_s. Constraint 1 represents that a task can be offloaded to only a single serving UAV even though the user can connect to multiple serving UAVs. Constraint 2 denotes that a user should be in the coverage of a serving UAV in order to connect it and then offload a task to it. Constraint 3 ensures that offloaded tasks cannot exceed the capacity of a serving UAV. Finally, Constraint 4 describes the movement constraints.

IV. DEEPAIR

Our DeepAir operation has four main phases that should be considered to provide the required QoS for user tasks. As formulated in Section III, a user task is successfully completed if the total delay in the system is smaller than or equal to its maximum tolerable delay. Therefore, each phase, including sensing, localization, resource allocation, and MEC, is significant for the efficient operation in the environment. To this end, we use detector UAVs for the sensing, localization, and resource allocation phases. Note that a detector UAV is also used as a DRL agent in the environment to find the user-concentrated areas. On the other hand, based on the reporting of detector UAVs, we deploy serving UAVs for the offloading and processing of tasks as part of the MEC resources. The whole operation including four phases is depicted in Figure 2.

Each type of UAV in the environment can communicate with each other via a separate channel. Thus, they know their existing location. Moreover, they can also communicate with the base, which is at (0, 0) coordinates, so that they can notify the existing situation in the corresponding areas in their horizontal coverage. As a result, the system can react to the events in those corresponding areas in real-time.

A. SENSING

Due to the nature of the infrastructure-less deployment, initially, users in the environment are not connected to any system component, emitting only signals for a possible connection. Therefore, as shown in Figure 3a, a detector UAV can sense signal strength at some point in the environment at time t. Based on the location of users and the detector UAV, that signal strength can change in different areas of the environment considering the cumulative signal strength measurements in Equations 11 and 12.

In this study, we assume that there are different user attraction points in the environment whose locations are also unknown. Based on those user attraction points as shown in Figure 3a, users are gathered in certain areas. Therefore, the corresponding additive signal strength would be higher when the detector UAV is close to that area. However, considering the fact that there would be many user attraction points whose locations could be in different parts of the environment, and some of those parts may have similar user densities, sensing levels can turn out to have identical or similar values for different points in the environment. Thus, this fact should be taken into account in the localization phase in which we use DRL.

B. LOCALIZATION

In the localization phase, information gathered in sensing is initially used for the movement of detector UAVs (agents) and

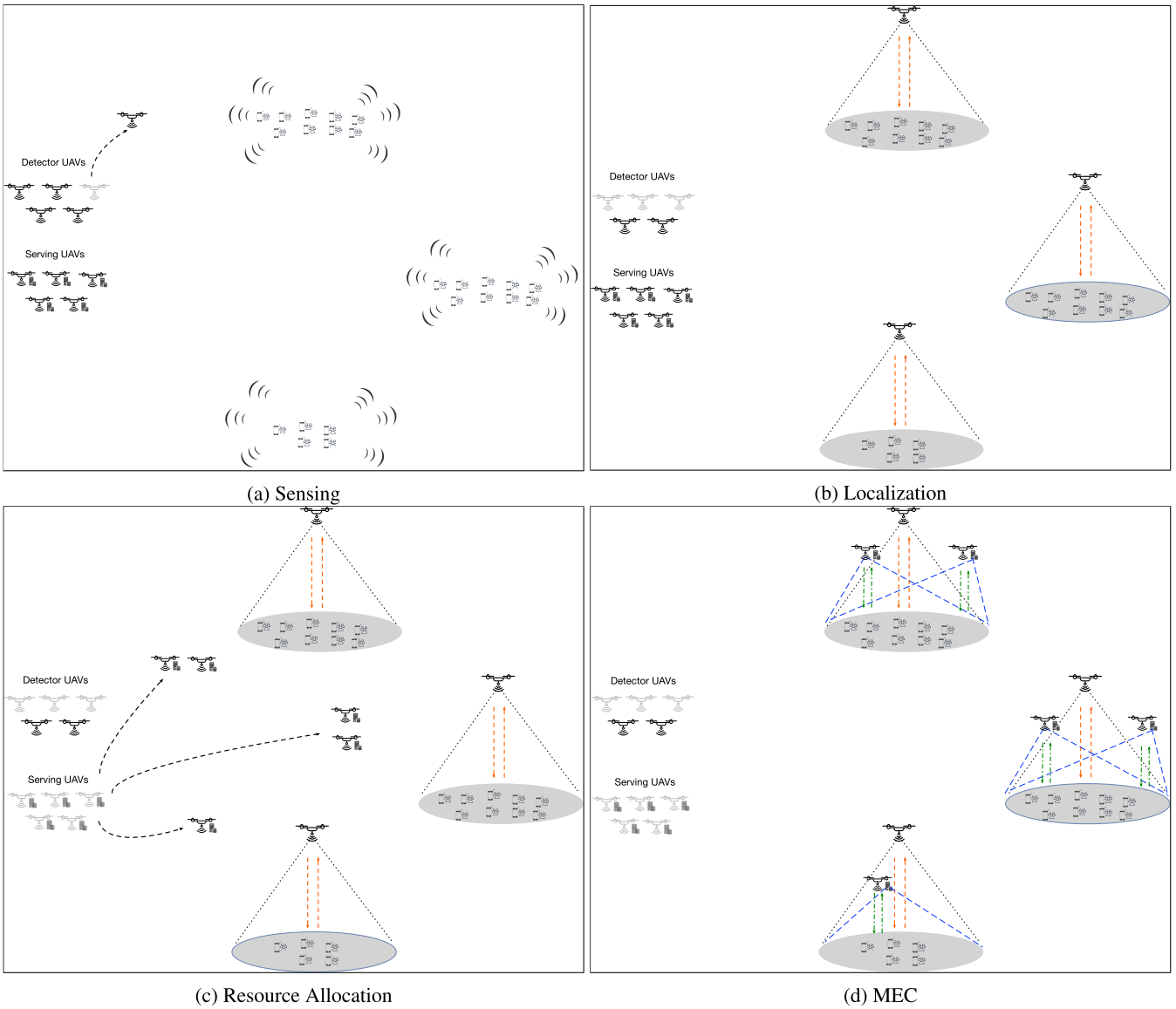


FIGURE 2. **Phases of DeepAir.**

then to locate the corresponding user attraction points. One of the crucial factors here is that the number of user attraction points in the environment is also unknown along with the number of users, and user locations. Therefore, we cannot apply multiple agents simultaneously in the environment since if the number of deployed agents is less than the number of user attraction points then users at some of the attraction points cannot be detected and served. Thus, we apply an iterative approach using DRL as shown in Algorithm 1.

In Algorithm 1, we find the user attraction points in the environment. To perform this, we initially set a threshold for new connected users in an iteration. For each iteration we send a DRL agent to the environment flying from the base at (0, 0) coordinates, and after the convergence it returns the corresponding location information along with how many new users are connected to it. If the number of connections is smaller than the threshold, the execution of the algorithm is terminated. Otherwise, we continue to send an agent to the environment. Note that when a user device connects to a detector UAV, it stops emitting the connection signal. As a result, the additive signal power would be less in a particular place in the next iteration for the agent. This situation is depicted in Figure 3.

1) MDP DEFINITION

The DRL is based on MDP which is formally defined as a 4-tuple $< S, A, P, R >$ where

- S is the set of states where $s \in S$
- A is the set of actions where $a \in A$
- $P : S \times A \rightarrow P(S)$ is the state transition probability function that provides the probability of

Algorithm 1 Finding Locations

```
Input: Threshold and the environment
Output: Found Locations
isThresholdMet = True
locations = [ ]
while isThresholdMet do
    singleLocation, connectionCount = DDQN()
    if connectionCount is smaller than Threshold then
        isThresholdMet = False
    else
        Add singleLocation to locations
    return locations
```

$P(s_t, s_{t+1}) = P(s_{t+1} = s' \mid s_t = s, a_t = a)$. This function denotes that the current state s_t at time t changes to the state s_{t+1} by taking the action a.

- $R : S \times A \times S \rightarrow R$ is the reward function. It defines the corresponding reward $\mathcal{R}$ at time t by taking an action a in a state s_t. Therefore, it can be also defined as $\mathcal{R}_t = R(s_t, a_t, s_{t+1})$.

Our environment provides the corresponding MDP definition through the state representation, action space, reward mechanism, and state transition. Therefore, an applied DRL algorithm in the environment using a detector UAV can create a policy π based on a given state as $\pi(a \mid s) = P(a_t = a \mid s_t = s)$. Through this policy, the agent can learn the dynamics in the environment for a given state and therefore takes the most effective action.

2) STATE SPACE S

Based on the Algorithm 1, there is only a single agent in the environment for each iteration. Therefore, the state at time t consists of the current location of the agent as

$$s(t) = u_n(t) = [x_n(t), y_n(t), z_n(t)] \tag{18}$$

3) ACTION SPACE A

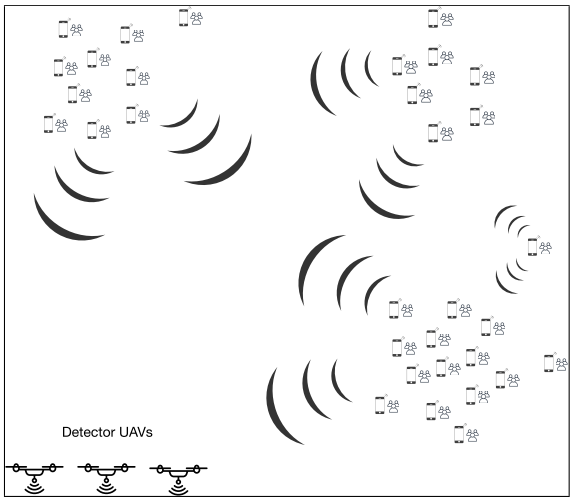
In our environment, the action space A consists of five discrete actions considering the horizontal movements. Therefore, it is defined as

$$a(t) = [Left, Right, Up, Down, NoMove] \tag{19}$$

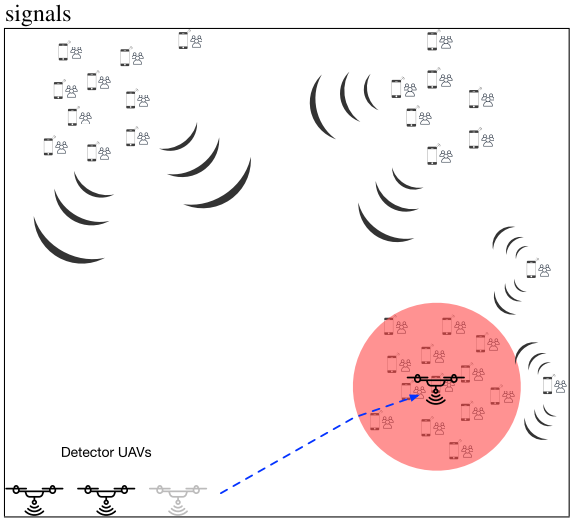
Based on this definition, the horizontal speed of each agent is fixed during their flight. Considering the selected action at time t, they can stay fixed at their horizontal coordinates by *NoMove* action. Otherwise, they can move into four different areas of the environment.

4) REWARD FUNCTION R

Based on the policy π, the agent takes the corresponding actions in the environment to maximize its cumulative reward. To this end, for a given state s_t, the agent maximizes the expected sum of future reward by applying policy $\pi(s_t)$



(a) Environment consisting of users that emits connection signals



(b) Emitted signals of connected users stop after the connection

FIGURE 3. Depiction of the state of environment regarding the emitted signals.

as follows

$$R_t = \sum_{i=t}^{\infty} \gamma * R(s_i, s_{i+1}) \tag{20}$$

where $\gamma \in [0, 1]$ is the discount factor that denotes the importance of the long-term rewards if its value is close to one. Otherwise, its value would be close to zero. Thus, the reward function is defined as follows based on the consideration above

$$R(t) = \begin{cases} H(t), & \text{if satisfying constraints} \\ -1, & \text{if } otherwise \end{cases} \tag{21}$$

5) APPLICATION OF DRL

Since our action space is discrete, applying a value-based DRL algorithm is more convenient in our environment.

Therefore, for each DRL agent, we implement Deep Q-Learning (DQN) algorithm [32], which manifests a high success in many different environments with different state spaces.

In value-based DRL algorithms such as DQN, the agent should select the best state-action pair among different options through its policy by a given state s_t by maximizing Q-function, $Q_\pi(s_t, a_t)$. Therefore, under the policy π, the Q-function is defined as

$$Q_\pi(s_t, a_t) = \mathbb{E}\left[R_t \mid s_i = s, a_i = a\right] \tag{22}$$

which denotes the value of an action a_t in a state s_t. Thus, an optimal policy is defined as selecting the highest valued action in each state

$$\pi(s_t) = arg\max_{a'}(Q(s_t, a')) \tag{23}$$

where a' indicates the set of all possible actions. As a result, the value of a state-action pair is computed as

$$z_t = R_{a_t}(s_t, s_{t+1}) + \gamma * Q(s_{t+1}, arg\max_{a'} Q(s_{t+1}, a'; \theta_t); \theta_t) \tag{24}$$

where θ_t represents the Q-network parameters. On the other hand, considering the convergence through the Q-network, the agent minimize the temporal difference error, δ_t, of $Q_\pi(s_t, a_t)$ regarding z_t:

$$\delta_t = \mid Q(s_t, a_t) - z_t \mid \tag{25}$$

C. RESOURCE ALLOCATION

After the completion of sensing and localization phases through detector UAVs, the next step is the resource allocation for serving UAVs which provide computational serving capabilities. Since a serving UAV has a limited capacity, measuring how many of them should be deployed is the main problem in this phase.

As users have already connected to the corresponding detector UAVs, and have started to send their task offloading requests, the system can create a task profile in those user-connected areas by conducting several capacity calculations. To this end, we perform a capacity calculation that includes the task profile of each user as we defined $W_m = (D_m, C_m, \lambda_m, T_m^{max})$. The measurements are essentially based on Equation 8 as the delay at serving UAVs is based on $M/M/1$ queueing model.

After the measurement of the required number of serving UAVs for each detected area, the other important issue is the deployment of available serving UAVs into those areas, each of which may have different task profiles. Note that the available serving UAVs may not meet the total capacity requirements in the environment. In this case, available serving UAVs are first deployed to the areas which have a higher need for serving UAVs regarding task profiles. On the other hand, if the required numbers of serving UAVs are equal for the corresponding areas, then a round-robin approach is applied for the deployment. Note that our primary goal for this computation and then deployment is to maximize the overall task success in the environment as defined in our objective function in Equation 17.

TABLE 4. Simulation parameters.

Parameter	Value
Number of User Attraction Points	[3, 5]
Size of a task (D_m)	500 Kb
Required CPU cycles of a task (C_m)	90 cycles/bit
Arrival rate of a task (λ_m)	0.30 task/sec
Maximum tolerable delay of a task (T_m^{max})	1 sec
Capacity of a serving UAV (f_{n_s})	300 cycles/sec
UAV Radius (r_n)	75 m
Data Rate (v_{m,n_s})	100 Mbps
Channel Power Gain (g_{m,n_d})	1.42×10^{-4}
New Connected User Threshold	1
X_{max}	500 m
Y_{max}	500 m
Altitude of UAVs ($z_n(t)$)	200 m

D. MEC

After the resource allocation, the next and final phase is providing the MEC services to users. To this end, users offload their tasks to serving UAVs and expect a service without violating the task's maximum tolerable delay, T_m^{max}.

A user in a corresponding area can connect to multiple serving UAVs simultaneously in the environment. Therefore, a user should select one of its connected serving UAVs at a particular time to offload the corresponding task. It performs this selection based on the existing queueing condition for each serving UAV. Note that since multiple users can connect to a single serving UAV, the $M/M/1$ queueing model is used for the total delay measurement at the serving UAV as explained in Section III. Here, we assume that a user can receive the recent queueing delay information from each serving UAV to which it is connected via a separate channel. Thus, it selects a serving UAV for task offloading considering the minimum queueing delay. As a result, our objective function defined in Equation 17 can be provided by minimizing the total delay regarding task profiles in the long term.

V. PERFORMANCE EVALUATION

We conducted experiments using a discrete event simulation for the performance evaluation. In these experiments, we have an environment whose size is 500×500 m^2. In this environment, there are various number of user attraction points around which the user densities are higher. Note that the location of those users and attraction points are initially not known by the system. The corresponding simulation parameters are given in Table 4. Throughout the experiments, we used Python 3.10. Moreover, we used PyTorch version 2.2.0 for the training of DRL agents.

In our experiments, we assumed that each user in the environment produces a task with the parameters D_m, C_m, λ_m, and T_m^{max}. Moreover, each produced task should be

TABLE 5. Hyperparameters of DRL algorithm.

Parameter	Value
Learning Rate	0.005
Discount Factor	0.99
Initial Exploration	1
Exploration Factor	0.99
Final Exploration	0.01
Replay Memory Size	1000000
Batch Size	64

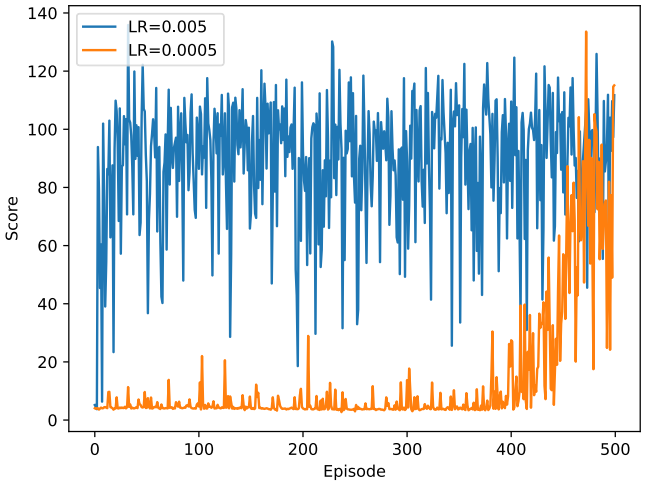


FIGURE 4. Effect of learning rate on scores for each episode using 100 users.

offloaded to one of the serving UAVs since we assumed that the computational capabilities of user devices are not sufficient to meet the task requirements. Similarly, each detector and serving UAV is identical in terms of radius, and altitude. Considering the offloading, we ignored the propagation delay for simplicity. Note that our essential goal was to maximize the overall task success as defined in Equation 17. We repeated our experiments 50 times with different seeds. The duration of each experiment in simulator time was 1000 seconds.

A. TRAINING STEP

Regarding Algorithm 1, we train a DRL agent through detector UAVs for each iteration as long as it provides new connections. To this end, we tested several hyperparameters throughout the training step in order to achieve convergence for different user distributions in the environment considering the final model. Generally, we used random search based on our experience in the domain [33].

The neural network in our final model consists of three layers each of which includes 128 neurons. Rectified Linear Unit (ReLU) is applied for each neuron as the activation function. We used mean squared error (MSE) for the loss, and stochastic gradient descent (SGD) as the optimizer. On the other hand, for the initial exploration, exploration factor, final exploration, and replay memory size, we used well-known values from the literature as given in Table 5.

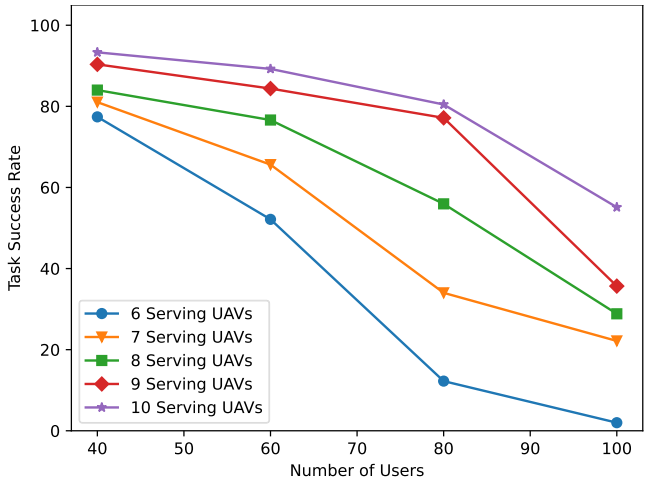


FIGURE 5. Effect of the number of users and serving UAVs on DeepAir.

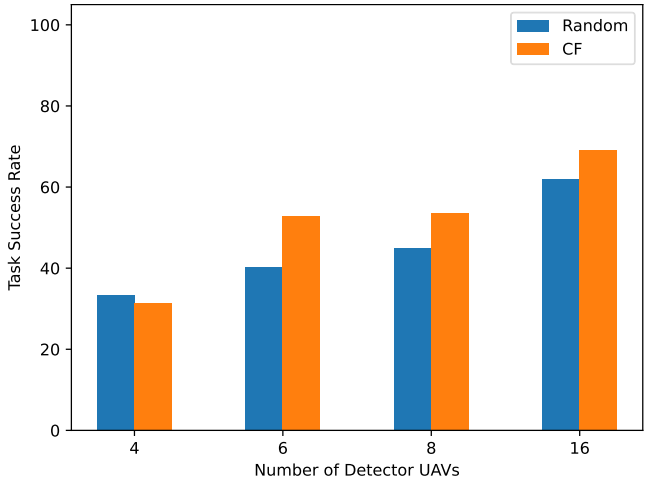
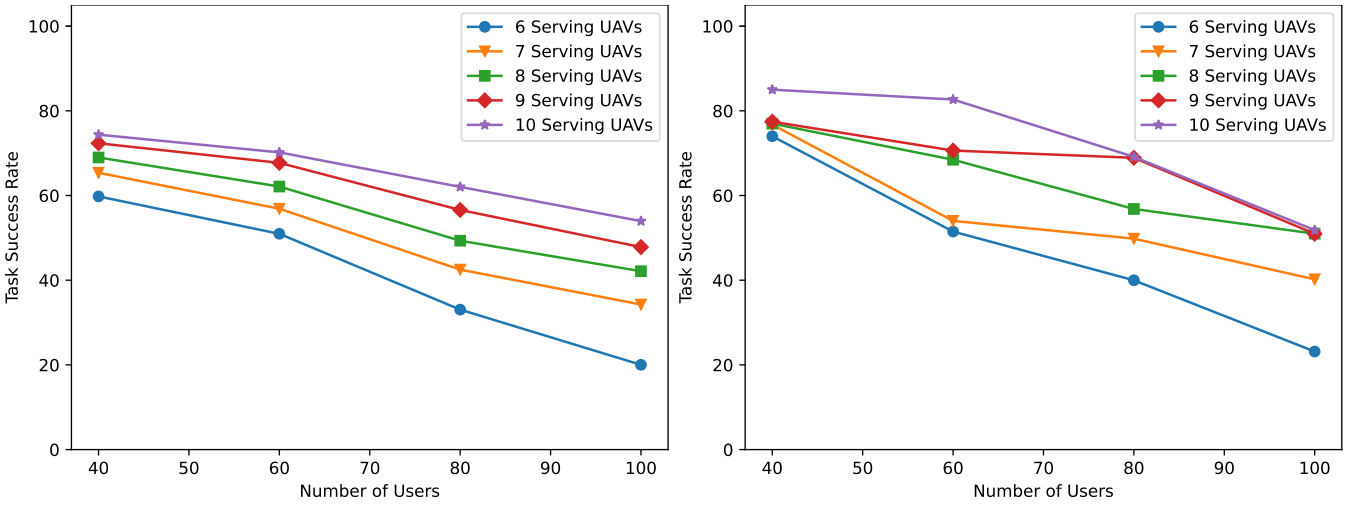


FIGURE 6. Effect of the number of detector UAVs on benchmark methods using 80 users.

Since the learning rate is the most crucial hyperparameter for the performance of the model, we exclusively tested different settings. Thus, we determined 0.005 as the final value for the neural network of the agents. Figure 4 shows the effect of two different learning rates over episodes. As shown in the figure, a lower value of learning rate, such as 0.0005, causes a late convergence in the environment. On the other hand, if we use 0.005 as the learning rate, the model converges earlier which provides time efficiency.

B. COMPETITORS

We used two competitors as benchmark methods namely the Community Flying (CF) and Random methods similar to [25]. In CF, we divided the environment into equal communities, and the center of each community was evaluated as a possible user attraction point. Accordingly, detector UAVs are sent to those centers for possible connections and corresponding QoS measurements. Afterwards, the required number of serving UAVs is deployed based on the needs



(a) Performance of Random-16

(b) Performance of CF-16

FIGURE 7. Results of benchmark methods.

of those areas. On the other hand, in the random method, the possible attraction centers are selected randomly in the environment based on the available number of detector UAVs. We named the random methods based on the available number of detector UAVs as in the case of CF. To this end, for example, if we use 8 detector UAVs, then the method is named as Random-8.

C. PERFORMANCE EXPERIMENTS

We first evaluate the performance of DeepAir considering the effect of varying numbers of users, and different serving UAVs. As shown in Figure 5, the success rate of DeepAir is quite high based on the successfully placed detector UAVs through DRL. Note that based on the configuration in the experiments, at most six detector UAVs are used when we apply DeepAir. On the other hand, when the number of users increases based on the same number of attraction points, the task success rate for each serving UAV case decreases. This is an expected result since the computational capacity of those serving UAVs would not be sufficient to meet the higher number of tasks produced by each user in the environment. Similarly, a higher number of serving UAVs results in a higher task success rate since they provide more computational capacity.

Prior to the evaluation of the performance of benchmark methods, we first conducted experiments to observe their success rate using different numbers of detector UAVs. To this end, as shown in Figure 6, we compared 4, 6, 8, and 16 detector UAV cases using 80 users. As expected, using an increased number of detector UAVs provided a higher task success rate since the probability of covered users in the environment is higher in that case. Therefore, we used CF-16 and Random-16 as the benchmark methods in the experiments.

The performance of Random-16 and CF-16 methods based on different numbers of users and serving UAVs are shown in Figure 7. The results manifest that CF-16 provides a better task success rate since it is a more structured approach. However, even though random decisions are taken in Random-16, when we use 9-10 serving UAVs, the task success rate is above 70% which is quite good. On the other hand, when the number of users increases, the task success rate decreases in both methods for the same reasons that we elaborated on regarding the results of DeepAir.

After the evaluation of benchmark methods and DeepAir, next we compared their most successful deployments in which we used ten serving UAVs. Figure 8 shows the results of this comparison. Based on the results, DeepAir outperforms both benchmark methods for different user counts. Note that since the arrival rate for each task, λ_m, is 0.30 task/sec, each user produces 300 tasks on average for each experiment. Moreover, both benchmark methods deployed 16 detector UAVs in this comparison, while DeepAir utilized at most six detector UAVs. On the other hand, the task success rate is close for each method when there are 100 users in the environment. This is the result of the insufficient computational capacity of serving UAVs deployed in the environment. Therefore, even if the best locations are selected for detector UAVs, the task success rate cannot exceed a certain value under those circumstances. Moreover, since the accuracy of location selection in Random-16 and CF-16 methods is less than that of DeepAir, the less number of connected users are served better using ten serving UAVs. This situation results in close values in terms of the task success rate when there are 100 users in the environment. In summary, although the user locations are successfully detected, if the number of serving UAVs is not sufficient for a certain load, the task success rate improvement requires more UAVs.

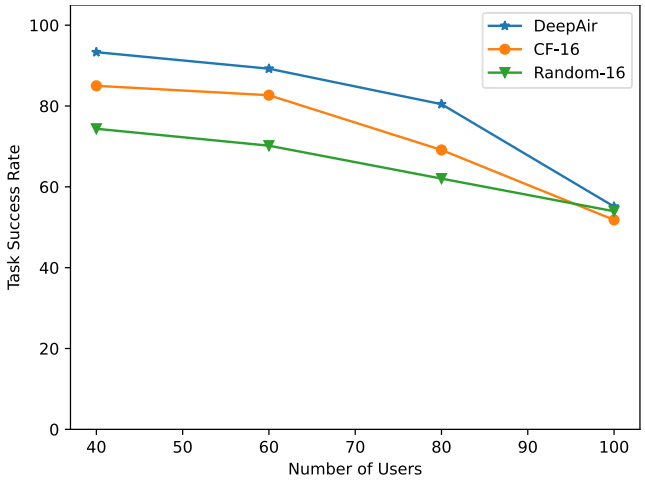


FIGURE 8. Performance comparison between DeepAir and benchmark methods using 10 serving UAVs. While CF-16 and Random-16 use 16 detector UAVs, DeepAir outperform them using at most six detector UAVs.

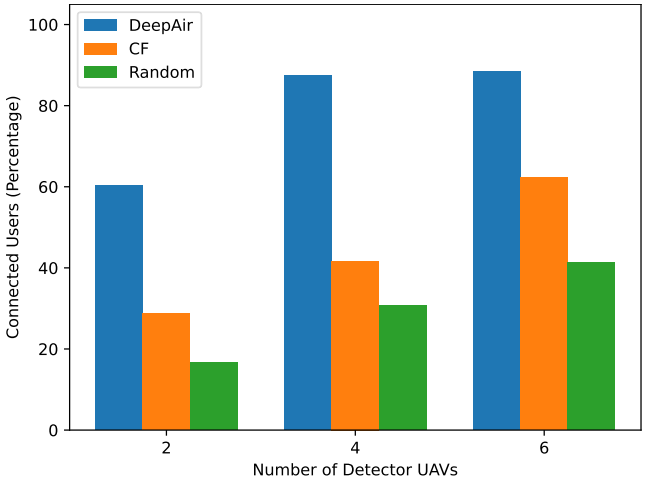


FIGURE 9. The effect of detector UAVs on the connected users.

Since the relationship between the number of detector UAVs and serving UAVs is also significant, and Figure 6 depicts only the performance of benchmark methods, we conducted additional tests. To this end, we configured an experiment based on the same parameters as given in Table 4. However, in this case, to evaluate and show the corresponding relationship more clearly, we fixed the number of user attraction points and users as five, and 50, respectively. To perform a fair evaluation, each user attraction point included 10 users along with different user distributions. We tested the effect of 2, 4, and 6 detector UAVs by deploying 8, 10, and 12 serving UAVs.

We first evaluated the percentage of connected users considering the number of detector UAVs based on each method as shown in Figure 9. The results show that when the number of detector UAVs increases, more users can be connected to the system as expected. As DeepAir uses a smart approach through DRL, it finds user locations

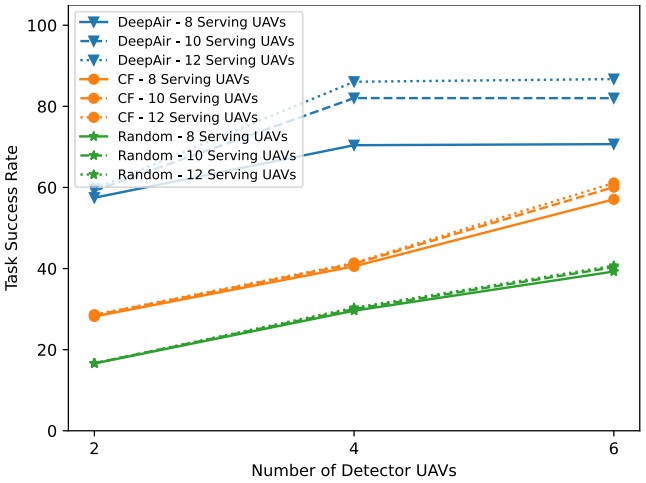


FIGURE 10. The effect of detector and serving UAVs on task success rate using 50 users.

efficiently even using a small numbers of detector UAVs. Therefore, it outperforms benchmark methods. On the other hand, considering the performance of benchmark methods, as detector UAVs are placed more strategically in the CF method, it detects users better than the Random method.

Next, we evaluated the task success rate by deploying 8, 10, and 12 serving UAVs based on the connected users by detector UAVs. Each user produces 300 tasks on average for each simulation, hence approximately 15000 tasks were produced for each experiment in total. The results given in Figure 10 show the relationship between the detector and serving UAVs more clearly. When there is a higher number of serving UAVs in the environment, the task success rate is close to the percentage of connected users. Moreover, the task success rate follows the pattern of the connected users through detector UAVs regarding different numbers of serving UAVs. Therefore, increasing the number of serving UAVs until the total capacity requirements are met in a detected area would increase the task success rate consistently.

VI. DISCUSSION

Considering the fact that the unknown user location problem has not been deeply investigated in the literature, we think that several points should be discussed based on our observations and experiments throughout this study. We first noted about the discrete action space for each type of UAV. Since continuous horizontal actions would complicate the already complex problem regarding DRL, we applied a discrete action space. Moreover, and more importantly, applying a discrete action space alleviates the problem since it turns that into a maze problem in which there are several different prizes (RSSI power) in different sections of the environment. The agent learns to follow those small prizes to reach a bigger prize through episodes. Therefore, the convergence of the agents would be more quickly compared to the case of a continuous action space. As a result we could apply the DQN

algorithm, which is a less complex regarding other DRL algorithms such as PPO, and DDPG.

Throughout our experiments, we also observed that higher RSSI due to a bigger number of users provides more accurate location prediction in DRL. As a result of this, more users can connect to the corresponding detector UAVs. Therefore, if the capacity of serving UAVs is so high, it is not so affected by the number of users, then the task success rate would be larger even though the load is increased. This is an important observation since, otherwise, the corresponding results would be evaluated incorrectly. For this reason, the selection of the capacity of serving UAVs and the required CPU cycles for user tasks are significant for manifesting the accuracy of the experiments.

VII. CONCLUSION

In this study, we investigated the unknown user location problem in a UAV-assisted environment. The corresponding environment can be a disaster site, wilderness, or a rural area in which user devices cannot connect to any communication device and edge servers because of the lack of infrastructure. Moreover, each user device produces tasks that should be completed regarding their maximum tolerable delay which is not met by the computational capabilities of user devices. Therefore, those tasks should be offloaded to the related computational units. In order to achieve this in such an environment, sensing, localization, resource allocation, and MEC capabilities should be provided together, sequentially. Therefore, we proposed DeepAir, a novel approach which uses DRL iteratively via detector UAVs that are responsible for sensing, localization, and resource allocation. Afterwards, MEC features are provided to those connected users by serving UAVs. Conducted experiments show that DeepAir provides a high task success rate by using a small number of detector UAVs in the environment regarding the benchmark methods.

In the future, we plan to take energy consumption into account since energy efficiency is crucial for the movements of the UAVs which would affect the performance. Therefore, we plan to optimize the trade-off between energy consumption and task success rate efficiently.

REFERENCES

[1] B. Yamansavascilar, A. Ozgovde, and C. Ersoy, "Air computing: A survey on a new generation computation paradigm," *Comput. Netw.*, vol. 251, Sep. 2024, Art. no. 110653.

[2] M. Y. Akhlaqi and Z. B. M. Hanapi, "Task offloading paradigm in mobile edge computing-current issues, adopted approaches, and future directions," *J. Netw. Comput. Appl.*, vol. 212, Mar. 2023, Art. no. 103568.

[3] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct. 2009.

[4] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[5] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How can edge computing benefit from software-defined networking: A survey, use cases, and future directions," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2359–2391, 4th Quart., 2017.

[6] *Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are*, Cisco, San Jose, CA, USA, 2017.

[7] M. Laroui, B. Nour, H. Moungla, M. A. Cherif, H. Afifi, and M. Guizani, "Edge and fog computing for IoT: A survey on current research activities & future directions," *Comput. Commun.*, vol. 180, pp. 210–231, Dec. 2021.

[8] H. Wang, G. Ding, F. Gao, J. Chen, J. Wang, and L. Wang, "Power control in UAV-supported ultra dense networks: Communications, caching, and energy transfer," *IEEE Commun. Mag.*, vol. 56, no. 6, pp. 28–34, Jun. 2018.

[9] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[10] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[11] N.-N. Dao, Q.-V. Pham, N. H. Tu, T. T. Thanh, V. N. Q. Bao, D. S. Lakew, and S. Cho, "Survey on aerial radio access networks: Toward a comprehensive 6G access infrastructure," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1193–1225, 2nd Quart., 2021.

[12] X. Cao, P. Yang, M. Alzenad, X. Xi, D. Wu, and H. Yanikomeroglu, "Airborne communication networks: A survey," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 1907–1926, Sep. 2018.

[13] B. Li, Z. Fei, and Y. Zhang, "UAV communications for 5G and beyond: Recent advances and future trends," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2241–2263, Apr. 2019.

[14] Y. Guo, Q. Li, Y. Li, N. Zhang, and S. Wang, "Service coordination in the space-air-ground integrated network," *IEEE Netw.*, vol. 35, no. 5, pp. 168–173, Sep. 2021.

[15] J. Liu, Y. Shi, Z. Md. Fadlullah, and N. Kato, "Space-air-ground integrated network: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2714–2741, 4th Quart., 2018.

[16] N. Cheng, W. Xu, W. Shi, Y. Zhou, N. Lu, H. Zhou, and X. Shen, "Air-ground integrated mobile edge networks: Architecture, challenges, and opportunities," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 26–32, Aug. 2018.

[17] A. Baltaci, E. Dinc, M. Ozger, A. Alabbasi, C. Cavdar, and D. Schupke, "A survey of wireless networks for future aerial communications (FACOM)," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 4, pp. 2833–2884, 4th Quart., 2021.

[18] Y. Xiao, J. Liu, J. Wu, and N. Ansari, "Leveraging deep reinforcement learning for traffic engineering: A survey," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 4, pp. 2064–2097, 4th Quart., 2021.

[19] Y. Bai, H. Zhao, X. Zhang, Z. Chang, R. Jäntti, and K. Yang, "Toward autonomous multi-UAV wireless network: A survey of reinforcement learning-based approaches," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 4, pp. 3038–3067, 2023.

[20] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and L. Hanzo, "Multi-agent deep reinforcement learning-based trajectory planning for multi-UAV assisted mobile edge computing," *IEEE Trans. Cognit. Commun. Netw.*, vol. 7, no. 1, pp. 73–84, Mar. 2021.

[21] Z. Chang, H. Deng, L. You, G. Min, S. Garg, and G. Kaddoum, "Trajectory design and resource allocation for multi-UAV networks: Deep reinforcement learning approaches," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 5, pp. 2940–2951, Sep./Oct. 2023.

[22] B. Zhang and K. Yang, "Multi-UAV searching trajectory optimization algorithm based on deep reinforcement learning," in *Proc. IEEE 23rd Int. Conf. Commun. Technol. (ICCT)*, Oct. 2023, pp. 640–644.

[23] B. Zhang, X. Lin, Y. Zhu, J. Tian, and Z. Zhu, "Enhancing multi-UAV reconnaissance and search through double critic DDPG with belief probability maps," *IEEE Trans. Intell. Vehicles*, vol. 9, no. 2, pp. 3827–3842, Feb. 2024.

[24] X. Cheng, R. Jiang, H. Sang, G. Li, and B. He, "Trace pheromone-based energy-efficient UAV dynamic coverage using deep reinforcement learning," *IEEE Trans. Cognit. Commun. Netw.*, vol. 10, no. 3, pp. 1063–1074, Jun. 2024.

[25] Z. Ning, Y. Yang, X. Wang, Q. Song, L. Guo, and A. Jamalipour, "Multi-agent deep reinforcement learning based UAV trajectory optimization for differentiated services," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 5818–5834, May 2024.

[26] H. Hao, C. Xu, W. Zhang, S. Yang, and G.-M. Muntean, "Joint task offloading, resource allocation, and trajectory design for multi-UAV cooperative edge computing with task priority," *IEEE Trans. Mobile Comput.*, vol. 23, no. 9, pp. 8649–8663, Sep. 2024.

[27] H. Shi, Y. Tian, H. Li, J. Huang, L. Shi, and Y. Zhou, "Task offloading and trajectory scheduling for UAV-enabled MEC networks: An MADRL algorithm with prioritized experience replay," *Ad Hoc Netw.*, vol. 154, Mar. 2024, Art. no. 103371.
[28] C. Zhang, Z. Li, C. He, K. Wang, and C. Pan, "Deep reinforcement learning based trajectory design and resource allocation for UAV-assisted communications," *IEEE Commun. Lett.*, vol. 27, no. 9, pp. 2398–2402, Sep. 2023.
[29] O. S. Oubbati, M. Atiquzzaman, A. Baz, H. Alhakami, and J. Ben-Othman, "Dispatch of UAVs for urban vehicular networks: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 70, no. 12, pp. 13174–13189, Dec. 2021.
[30] M. Yan, R. Xiong, Y. Wang, and C. Li, "Edge computing task offloading optimization for a UAV-assisted Internet of Vehicles via deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 73, no. 4, pp. 5647–5658, Apr. 2024.
[31] Z. Gao, L. Yang, and Y. Dai, "MO-AVC: Deep-reinforcement-learning-based trajectory control and task offloading in multi-UAV-enabled MEC systems," *IEEE Internet Things J.*, vol. 11, no. 7, pp. 11395–11414, Apr. 2024.
[32] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.
[33] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 281–305, Mar. 2012.

BARIS YAMANSAVASCILAR received the B.S. degree in computer engineering from Yildiz Technical University, Istanbul, in 2015, and the M.S. degree in computer engineering from Bogazici University, Istanbul, in 2019, where he is currently pursuing the Ph.D. degree with the Computer Engineering Department. He is currently a Senior Research Engineer with Airties. His research interests include machine learning, deep reinforcement learning, edge/cloud computing, and software-defined networks.

ATAY OZGOVDE (Senior Member, IEEE) received the B.S. and M.S. degrees from Bogazici University, Istanbul, in 1995 and 1998, respectively, and the Ph.D. degree from the NETLAB Research Group, Bogazici University, in 2009. In 2002, he started working as a Research Assistant with the Computer Engineering Department, Bogazici University. He was a Researcher with the WiSE-Ambient Intelligence Group to complete his postdoctoral research. Currently, he is an Assistant Professor with the Computer Engineering Department, Bogazici University. His research interests include wireless sensor networks, embedded systems, distributed systems, pervasive computing, SDN, and mobile cloud computing.

CEM ERSOY (Senior Member, IEEE) received the Ph.D. degree from Polytechnic University, New York City, in 1992. He was a Research and Development Engineer in NETAS A.S., between 1984 and 1986. He became a Professor in computer engineering with Bogazici University. His research interests include wireless/sensor networks, activity recognition and ambient intelligence for pervasive health applications, green 5G and beyond networks, and mobile cloud/edge/fog computing. He is a member of IFIP and was the Chairperson of the IEEE Communications Society Turkish Chapter for eight years.