Deep Reinforcement Learning-Based Data Download Scheduling Algorithm for Internet of Drones Systems

Image Bhattarai Cong Pu
Department of Computer Science, Oklahoma State University, Stillwater, OK, USA
Email: image.bhattarai@okstate.edu, cong.pu@ieee.org

***Abstract*—The Internet of Drones (IoD) paradigm has noticed a growing demand for efficient and sustainable operations management, with a particular emphasis on service request scheduling. As IoD systems become widespread in various domains, such as ecological surveillance, facilities evaluation, etc., it is crucial to efficiently manage and prioritize the execution of drones' data download requests at the ground stations. Due to the unique characteristics of IoD systems, e.g., high drone density and mobility, limited ground station communication range and energy resources, and substantial deployment and maintenance cost, how to enable the ground stations to scalably, intelligently, and optimally answer drones' data download requests concurrently has become a challenging issue. In this paper, we propose a *d*eep *r*einforcement learning-based *d*ata download *r*equest sch*e*duling mechanism (hereafter referred to as *DrDre*), enabling ground stations to plan and execute received data download requests from drones in a scalable, intelligent, and optimal manner. *DrDre* takes into account various scheduling parameters such as request deadline, request urgency, request priority, requested data size, and data popularity when making scheduling decisions. In addition, *DrDre* exploits the deep reinforcement learning and deep Q-learning frameworks to formulate the dynamic IoD environment into a Markov Decision Process (MDP) model and enable ground stations to learn an optimal scheduling policy in order to maximize cumulative rewards over time, respectively. We conduct extensive and comparative simulation-based experiments in a customized simulation environment using SUMO and Python. Simulation results show that *DrDre* outperforms the selected benchmark schemes in terms of request satisfaction rate, request fulfillment latency, the amount of downloaded data, and the number of incomplete requests.**

***Index Terms*—Data download request, scheduling, Internet of Drones, deep reinforcement learning, Q-learning.**

I. Introduction

The rise of unmanned aerial vehicles, also widely known as drones, has brought forth a transformative change across various domains. For instance, not only can drones be developed to transport passengers within urban areas (e.g., air taxis) [1], but they can also be used to revolutionize the way goods are delivered (e.g., aerial delivery) [2]. Leveraging the paradigm of Internet of Drones (IoD) [3], an aerial-ground cohesive ecosystem can be established, where drones communicate with ground stations and act upon directives issued by the system operator. By capitalizing on the seamless connectivity between aerial and ground networks provided by the IoD paradigm, drone-based civilian and government applications can achieve higher levels of automation and boost productivity.

This work was supported by the National Science Foundation (NSF) through SaTC under Award 2333777.

In the third decade of the 21st century, the IoD technology has already been employed to revolutionize urban landscapes through the implementation of smart city initiatives by the U.S. government. In early 2021, NASA proposed the Advanced Air Mobility (AAM) Mission [4], which strives to revolutionize communities by shifting the transportation of people and goods from the ground to the skies, available on demand. In this AAM initiative, NASA takes the responsibilities of supplying essential data to steer the progress of the electric air taxi and drone industry, and supporting the Federal Aviation Administration (FAA) in securely incorporating these aircraft into the national airspace. To pave the way for this flourishing aerial industry by 2030, the IoD-specific research challenge like service request scheduling needs to be addressed promptly. First, drones move at a high velocity (e.g., around 35-60 mph for commercial drones) and the communication range of ground stations is limited (e.g., about 12-19 miles for cellular networks), thus, the drones' data download requests are always subject to a firm deadline (i.e., it takes approximately 20 minutes for a commercial drone to fly over a ground station). Second, owing to their expanding applications across different sectors, advancements in sensing and connectivity technology, and regulatory adjustments that support their use, the drone density within the ground stations' communication range continue to rise. As a result, a high volume of data download requests might become a concern, and how to prioritize fulfilling data download requests to prevent bottlenecks and ensure timely transmission of requested data will be critical to the overall performance of scheduling. Third, drones may request data for various applications, each with distinct quality of service (QoS) requirements. For example, delivery drones may request weather information for trajectory planning and execution, while drone taxis may request pickup time, arrival time, and route lines to create real-time experiences for passengers. Therefore, the scheduling algorithm should plan and serve various data download requests based on their distinct QoS needs. Although densely deploying ground stations can be a viable solution for all above-mentioned challenges, it incurs substantial implementation and running costs.

During the past few years, a variety of scheduling algorithms have been proposed to enhance the efficiency of data downloading, ensuring efficient and reliable data exchange. Unfortunately, the existing techniques either ignore the dynamic nature of the IoD systems or are not subject to strict IoD system constraints. For example, Pu *et al.* [5]

propose a scheduling algorithm that calculates the priority of each service request using multi-attribute decision making theory for IoD systems. However, Pu *et al.*'s approach only focuses on making an optimal scheduling decision at the specific moment without considering how these scheduling decisions could affect and influence future events. In [6], [7], the authors propose machine learning-based scheduling algorithms to optimize power consumption and reduce environmental emissions for smart grid networks. Nevertheless, these scheduling algorithms specifically designed for electricity networks are not well-suitable for IoD networks due to the fundamental differences between these two systems. Non-static topology, finite communication range, and dynamic communication conditions complicate the design of scheduling algorithms in the IoD environment.

Driven by the preceding discussion, in this paper we propose a *d*eep *r*einforcement learning-based *d*ata download *r*equest sch*e*duling mechanism (hereafter referred to as *DrDre*) to enable ground stations to plan and execute received data download requests from drones in a scalable, intelligent, and optimal manner. The basic idea is that *DrDre* exploits the deep reinforcement learning and deep Q-learning frameworks to formulate the dynamic IoD environment into a Markov Decision Process (MDP) model and enable ground stations to learn an optimal scheduling policy in order to maximize cumulative rewards over time, respectively. In addition, multiple scheduling parameters, such as request deadline, request urgency, request priority, requested data size, and data popularity, that affect the action space are taken into account to build the system state. To evaluate the performance of *DrDre*, we conduct extensive and comparative simulation-based experiments in a customized simulation environment using SUMO and Python. According to the simulation results, our approach *DrDre* has demonstrated superior performance in terms of request satisfaction rate, request fulfillment latency, the amount of downloaded data, and the number of incomplete requests, while comparing with the selected benchmark schemes.

The rest of the paper is organized as follows. We review relevant related works in Section II. Next, we introduce our approach *DrDre* in Section III. Subsequently, we present the analysis of the experimental results in Section IV. Finally, we conclude the paper in Section V.

II. Related Work

Over the last several years, there have been numerous scheduling schemes proposed for various systems. In [8], the authors leverage the ant colony optimization approach to optimize scheduling in cloud environments. They assign each task to a virtual machine with minimal execution cost and idle time, and utilize a weighted rank model to adjust deadlines and minimize execution costs. Although their work can reduce execution cost, the complexity of the ant colony optimization algorithm is not properly taken into account, where multiple iterations and updates are required to simulate the behavior of ants in nature. Moreover, the weighted rank model requires fine-tuning parameters that involve the detailed knowledge and resources of the system, which makes it impractical for general use. The authors in [9] address the network congestion issues in the Internet of Things (IoT) systems. The dueling double deep Q-network and bidirectional long short-term memory network are used to assign IoT devices to edge servers and process historical assignment data, respectively. However, the proposed solution assumes that the IoT system exhibits stable and static behavior, which is unrealistic in a real-world scenario. Additionally, frequently aggregating and transmitting model updates to the cloud can cause bottlenecks in the edge-cloud communication links. In [10], a task scheduling algorithm is proposed for small independent tasks that require parallel execution in the IoT environment, where a multi-objective function is used to model execution time, cost, and failure rate. Moreover, the proposed algorithm relies on the grey wolf algorithm to balance both exploration and exploitation in the search space in order to achieve better solutions. The weakness of this approach, however, is that the size of the search space increases after each iteration. This is because it contains multiple solutions requiring pairwise comparison and evaluation of the fitness function. As a result, the proposed approach becomes computationally expensive and less efficient for real-time scheduling applications.

The authors in [11] focus on communication and computation overhead challenges existing in the Internet of Vehicles (IoV) environment. They implement the federated learning in the IoV, where the interconnected vehicles can collaboratively train generative adversarial network models without transferring raw data. In addition, they choose a deep deterministic policy gradient reinforcement learning algorithm to optimally select participating vehicles for each aggregation round. However, since the federated learning requires multiple iterations of uploading distributed updates and downloading the aggregated model, the proposed approach necessitates a high bandwidth requirement, making it impractical for IoV networks with limited bandwidth. In [12], the authors introduce a resource scheduling algorithm to distribute tasks to mobile edge computing (MEC) nodes efficiently, prioritizing reducing energy consumption and network latency to distribute tasks to mobile edge computing (MEC) nodes. While the proposed scheduling algorithm focuses on energy consumption and network latency, other critical aspects, such as fair resource allocation and task priority, are not taken into account. In [13] , a UAV-assisted task scheduling algorithm is introduced to allow users to offload their tasks to the UAV for processing based on their task priorities while meeting diverse service requirements. To achieve its overall objective, the authors use the K-means algorithm to cluster users into different groups based on task priority and location, and deploy the genetic algorithm to optimize UAV scheduling and task offloading. Since the genetic algorithm iteratively solves task offloading and scheduling problems separately, the computational complexity of the proposed algorithm is relatively high. Furthermore, fair scheduling is missing as the proposed algorithm prioritizes tasks with high priority, leading to possible starvation of lower-priority tasks.

In summary, even though there has been significant research

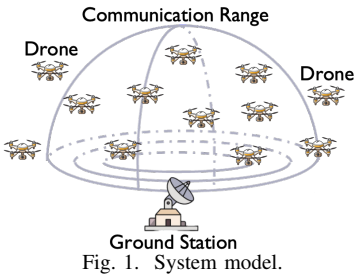

Fig. 1. System model.

conducted on applying scheduling to similar environments such as IoV and IoT, the existing solutions have certain areas that require further improvement. These outlined areas for enhancement emphasize the research gap in the realm of IoD and necessitate a novel scheduling approach. Thus, in this paper we present a deep reinforcement learning algorithm to schedule the IoD data download request from drones in a scalable, intelligent, and optimal manner. The proposed research work will be crucial to the IoD community and will act as a building block for future research efforts.

III. The Proposed Approach

A. System Model

The system model for our approach *DrDre* is shown in Fig. 1, where a group of drones (e.g., the i^{th} drone is denoted as D_i) fly past the communication range of the ground station G_k. In the proposed system, time is divided into a sequence of consecutive service windows, e.g., the j^{th} service window is represented as ω_j. ω_j is further split into two sequential sub-windows: request reception ω_j^{rec} and service execution ω_j^{exe} sub-windows. During ω_j^{rec} and ω_j^{exe}, G_k will receive and satisfy data download requests from drones, respectively. In order to notify newly arriving drones that they can still submit data download requests, G_k continuously broadcasts beacon messages within ω_j^{rec}. We assume that D_i is equipped with a global positioning system and inertial measurement units to track its trajectory information (e.g., current location as well as moving speed and direction), and the trajectory information enables D_i to estimate the deadline of its data download requests. Here, the deadline is the time when D_i flies out of the communication range of G_k. We also assume that D_i flies at a constant speed along a predetermined route within the communication range of G_k. During ω_i^{exe}, G_k starts satisfying the received data download requests from drones based on the scheduling policy learned from our approach *DrDre*. If G_k cannot satisfy D_i's data download request before the deadline, the request will be discarded.

B. Overview of Our Approach DrDre

The basic idea of our approach *DrDre* is that, when D_i enters the communication range of G_k, it submits a data download request piggybacked with its identifier, data ID, data size, and request deadline if ω_j^{rec} is still active. At the beginning of ω_j^{exe}, G_k counts the number of data download requests piggybacked with the same data ID, and this number will indicate the popularity of the requested data. Thereafter, G_k calculates the urgency and priority for each received data download request ,and feeds the request deadline, request urgency, request priority, requested data size, and data popularity to the deep reinforcement learning framework. In the *DrDre*, the dynamic IoD environment is formulated into a Markov Decision Process (MDP) model, which will be used by the deep Q-learning framework to learn the optimal scheduling policy. The scheduling parameters such as request deadline, request urgency, request priority, requested data size, and data popularity enable G_k to capture the dynamics of the IoD network during each time step, thereby helping to define the state space of the MDP model. G_k evaluates the current state, takes an action from its action space, and moves to a new state. Based on the outcomes (e.g., request satisfaction or discard) of its action, G_k either receives a positive reward or a negative reward. G_k's objective is to learn the most optimal scheduling policy to maximize the cumulative reward. A more detailed discussion of the MDP model is given in the following subsection. In summary, through continuous optimization, G_k can eventually improve the performance of scheduling operations.

C. Data Download Request Scheduling

When ω_j^{rec} is active, G_k broadcasts beacon messages periodically to inform nearby drones about the availability of the request reception service. Any drone who is interested in requesting data download service from G_k can send a data download request packet R_q = (D_i, $Data_{id}$, DL_i, D_s), where D_i represents the unique identifier for the i^{th} drone, $Data_{id}$ indicates the data item the drone is interested in, DL_i is the data download request deadline, and D_s is the size of the requested data. When ω_i^{exe} begins, G_k prepares the five scheduling parameters that are to be used in our approach *DrDre*. First, G_k tracks how many times each data ID $Data_{id}$ appears in received requests and uses this count to measure data popularity Pop_{id}. Next, G_k determines the priority Pri_i for each data download request. Pri_i is calculated as Pri_i = $\frac{1}{B_i}$, where B_i is the residual battery percentage for drone D_i. Drones with lower residual battery percentages are prioritized over those with higher percentages, as they may not reach the next ground station in time for requesting data download services. Finally, G_k calculates the request urgency U_i as U_i = ($\alpha \cdot \frac{1}{DL_i}$ + $\beta \cdot Pop_{id}$ + λ Pri_i). α, β, and λ represent the weights assigned to the request deadline, data popularity, and priority, respectively. Here, the request urgency increases with smaller deadlines, more frequently requested data items, and lower battery percentages.

To find the optimal scheduling decision, our approach *DrDre* formulates the dynamic IoD environment as a Markov Decision Process (MDP) model. The MDP model comprises an agent, represented by G_k, that is responsible for scheduling data download requests from multiple drones within its communication range. At every point in time t, G_k observes the current system state s_t, which includes information such as the number of received data download requests, the sizes of requested data, the request deadline, urgency, and priority. G_k then chooses an action a_t which causes the system to

transition to a new state s_{t+1}. Here, the action a_t is selected from the action space and denotes satisfying the data download request from the i^{th} drone D_i. Upon moving onto the new state, G_k either receives a positive reward or a negative reward, based on the outcome of its action. For instance, when a data download request with a higher priority is satisfied, G_k receives a positive reward. Conversely, if a data download request with a lower priority is served, or a data download request cannot be served before the deadline, G_k receives a negative reward. The goal of G_k is to discover the most optimal policy that maximizes long-term cumulative rewards for the IoD environment.

1) State Space (Observations) / Input from Environment: At t, G_k is fed the system state s_t as input. Suppose that there are N drones in the network, the system state s_t is composed of $s_t = \{s_{1,t}, s_{2,t}, s_{3,t}, \ldots, s_{N,t}\}$, where the state for the drone D_i at time t is represented as $s_{i,t} = [U_i, DL_i, Pri_i, Pop_{id}, D_s]$.

2) Action Space: At t, G_k chooses an action from the following action space, $a_t \in \{1, 2, 3, \ldots, N\}$. Here, $a_t = i$ action indicates that G_k satisfies the data download request from D_i. For instance, $a_t = 3$ indicates that the data download request from the 3^{rd} drone D_3 gets served. By learning the optimal action policy, G_k demonstrates good judgment in satisfying the data download request while balancing all aspects of the requests.

3) Rewards: G_k chooses an action at t and receives a step reward r_t based on the outcome of the action. A step reward is formulated to encourage adherence to request deadlines, minimize satisfaction latency, and maximize data download throughput, while penalizing missed deadlines. G_k receives a positive reward if D_i's data download request is satisfied before its deadline, and a negative reward if its request isn't satisfied before the deadline. Likewise, the reward function also considers request urgency and priority, where scheduling a data download request with lower priority and urgency results in a penalty. Since G_k must consider the long-term effect of its actions in addition to the immediate rewards, it uses the function below to calculate the cumulative reward.

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{1}$$

The discount factor γ is a value between 0 and 1, which signifies how important future rewards are. When γ is close to 0, G_k will prioritizes immediate rewards. If γ approaches 1, G_k will value its long-term rewards. Hence, the most optimal policy π^* is given by $\pi^* = \arg\max_{\pi \in \mathcal{P}} R_\pi$, where $\mathcal{P}$ is the set of all possible policies and π^* is the optimal policy. According to [14], the value function V^π satisfies the following Bellman equation for a given policy $\pi \in \mathcal{P}$ is given by

$$V^\pi(s_t) = r(s_t, \pi(s_t)) + \gamma \sum_{s_{t+1}} P(s_{t+1} \mid s_t, \pi(s_t)) V^\pi(s_{t+1}) \tag{2}$$

To solve the equation above, the value of state transition function $P(s_{t+1}|s_t, \pi(s_t))$ is required. However, the exact transition probability $P(s_{t+1}|s_t, a_t)$ is unknown by G_k. Therefore, Q-learning is applied by G_k as it does not rely on transition probabilities.

D. Deep Q-Learning (DQN)

Q-learning enables the learning of optimal policies without explicitly requiring the knowledge of transition probabilities. Unlike traditional dynamic programming approaches, Q-learning doesn't estimate state transitions but rather learns the most optimal policy through direct interaction with the environment. According to [14], Q-values Q^* for each (s_t, a_t) is given by $Q^*(s_t, a_t) = r(s_t, a_t) + \gamma \sum_{s_{t+1} \in s_t} P(s_{t+1}|s_t, \pi(s_t)) V^*(s_t)$. Since $V^*(s_t) = max_{s_t} Q^*(s_t, a_t)$, the optimal policy can be written as

$$\pi^*(s_t) = \arg\max_{a_t} Q^*(s_t, a_t). \tag{3}$$

The Q-value is incrementally updated by the equation below:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \big[r(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \big], \tag{4}$$

where $\alpha \in [0,1]$ represents the learning rate that controls the size of each update. The problem, however, is that due to the high-dimensional state space, storing and updating values in the Q-table becomes impractical. To overcome this limitation, a deep Q-network (DQN) is employed by G_k to approximate Q-values without maintaining a table. For a neural network with parameters θ, the Q-values can be approximated by using the loss function below:

$$L_i(\theta_i) = \mathbb{E}\left[(y_i - Q(s_t, a_t; \theta_i))^2 \right] \tag{5}$$

where $y_i = r + \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_{i-1})$. For each iteration, the value depends on the parameters from the previous iteration, θ_{i-1}, which are fixed and not updated. G_k's objective is to compute the weights of the neural network so that the loss function attains the most minimal value. To this end, gradient descent is used, which is an algorithm that minimizes errors between target values and actual values,

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s_t, a_t, r_t, s_{t+1}} \big[(y_i - Q(s_t, a_t; \theta_i)) \times \nabla_{\theta_i} Q(s_t, a_t; \theta_i) \big] \tag{6}$$

Here y_i is the target value and $Q(s_t, a_t; \theta_i)$ is the current Q-value estimate for θ_i. $\nabla_{\theta_i} Q(s_t, a_t; \theta_i)$ is the gradient of the Q-network for weight θ_i. Gradient descent can prove to be inefficient in computing the cost and gradients for a large training set. A more effective approach can be to use Stochastic Gradient Descent (SGD) which uses small batches of data to update parameters to estimate gradient. However, the standard Q-learning algorithm might diverge especially when used with large neural networks. In [15], to ensure that large neural networks do not oscillate, modifications to the algorithm were introduced. First, the authors drew samples of the agent's experience at random instead of taking the latest experience samples. This helps in lowering the correlation between the most recent samples, thereby reducing the variance of updates. Second, the target y_i was generated using parameters with older values by cloning and using a separate Q-network. This

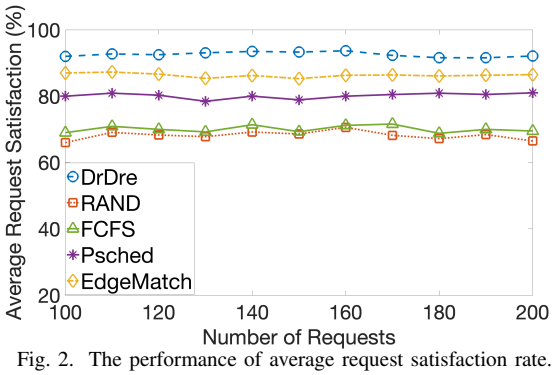

Fig. 2. The performance of average request satisfaction rate.

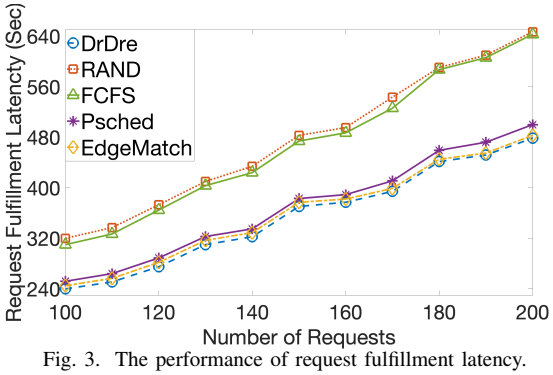

Fig. 3. The performance of request fulfillment latency.

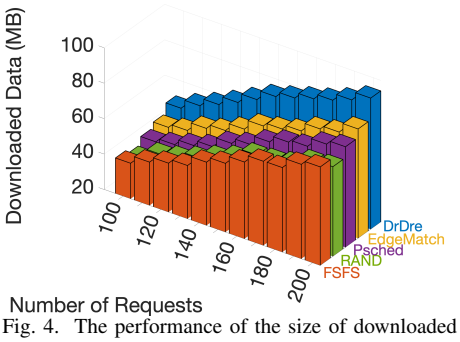

Fig. 4. The performance of the size of downloaded data.

makes the algorithm more stable and less divergent. Lastly, the error term y_i - $Q(s_t, a_t; \theta_i)$ was set to have a value between -1 and 1. This was done so that the outliers outside of the range (-1, 1) did not introduce volatility during the learning phase.

IV. Performance Evaluation

To assess the performance of our approach *DrDre*, we develop a customized simulation environment on an Apple iMac with a M3 chip and 24GB of memory. In addition, we select SUMO to accurately simulate drones' movement and service request load, and the patterns of drone traffic and mobility are saved in the CSV trace file. The CSV trace file contains approximately 1 million entries generated over multiple iterations in SUMO, and those entries will serve as the dataset for the Deep Q-Network (DQN) to train and determine the optimal policy. To be specific, in the SUMO simulation each drone moves along a predetermined route and generates data download requests relative to its position. Time is divided into distinct service windows. During each service window, a drone generates either one or two data download requests. We assume that the number of drones in the network ranges from 10 to 20, and the size of requested data varies between 1 MB and 10 MB. We also assume that each drone is equipped with a limited battery supply to introduce variance in the priority of data download requests. For performance comparison and analysis, we choose First-Come-First-Served (FCFS), Random Selection (RAND), Psched [5], and EdgeMatch [16], and compare them with our approach *DrDre*.

First, we measure the average request satisfaction rate of *DrDre*, FCFS, RAND, Psched, and EdgeMatch and present the results in Fig. 2. Here, the average request satisfaction rate is calculated as the total number of satisfied requests divided by the total number of received requests over a number of randomly drawn service windows. As depicted in Fig. 2, our approach *DrDre* shows the highest average request satisfaction rate because it continuously improves the scheduling policy and maximizes future rewards based on past experiences. Over the long term, our approach *DrDre* is able to prioritize requests while balancing multiple factors such as request deadline, request urgency, data size, request priority, and data popularity, finally maximizing request satisfaction rate. EdgeMatch shows a comparatively lower average request satisfaction rate than our approach *DrDre*. EdgeMatch employs a multi-criteria decision-making approach, it mainly relies on weights to compute a comprehensive score for each request, making it less adaptive to dynamic IoD networks than our approach *DrDre*. Psched achieves a lower request satisfaction rate than EdgeMatch and *DrDre*. This is because Psched focuses solely on static priority scores of requests within service windows, without considering future events. FCFS and RAND lack intelligence when making scheduling decisions, simply satisfying requests in the order they arrive or in a random order, respectively. As a result, they obtain the lowest request satisfaction rate.

Second, we present the results of request fulfillment latency for varying numbers of requests in Fig. 3. Here, the request fulfillment latency is regarded as the amount of time elapsed between when a drone submits a request and when the ground station serves the request. In Fig. 3, it is clearly shown that our approach *DrDre* delivers the lowest latency compared to the other four approaches. The rationale behind that is our approach *DrDre* continuously learns to prioritize requests with smaller data sizes while taking into account their associated deadline, thereby reducing the overall request fulfillment latency. EdgeMatch and Psched assign static weight to scheduling parameters such as deadline and data size, and it does not flexibly adapt to dynamic IoD environment as our approach *DrDre* does, thus, a higher request fulfillment latency is obtained. FCFS has a considerably larger request fulfillment latency because it mainly handles requests based on their arrival time. The worst performance is achieved by RAND as it randomly chooses service requests to satisfy, which causes unurgent requests to be served earlier than urgent ones. As a result, the overall request fulfillment latency increases significantly.

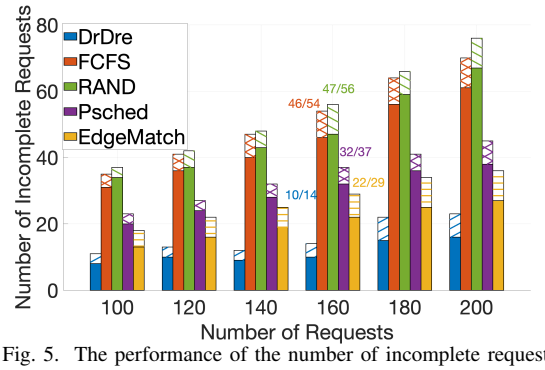

Fig. 5. The performance of the number of incomplete requests.

Third, we measure the amount of downloaded data by changing the number of requests for FCFS, RAND, Psched, and EdgeMatch, and *DrDre* in Fig. 4. As can be seen in Fig. Fig. 4, RAND downloads the least amount of data because it randomly satisfies requests without considering data size or service completion likelihood. FCFS shows a higher amount of downloaded data than RAND. As FCFS serves more requests than RAND, drones can download more data using FCFS. Psched and EdgeMatch perform better than both FCFS and RAND because it consider multiple scheduling factors. Our approach *DrDre* achieves the best performance in terms of the amount of downloaded data. This is because not only does *DrDre* have the highest request satisfaction rate which causes more data to be downloaded, but *DrDre* also learns to improve the scheduling policy over time to satisfy more requests.

Finally, we demonstrate the number of incomplete requests against the number of requests in Fig. 5, where the dashed bar area indicates an increase in the number of incomplete requests as the data size of each request is uniformly increased. Apparently, our approach *DrDre* generates the fewest incomplete requests, and the increment is low when data size is increased. Psched shows a higher number of incomplete requests but with a slow increment. This is because Psched considers data size as a primary parameter for scheduling. The number of incomplete requests of EdgeMatch is higher than that of our approach *DrDre*. In addition, the dashed bar area reveals a high increment. This is due to the lack of adaptability to dynamic IoD environment and static weight assignment. The worst performance belongs to FCFS and RAND due to their simplified design of scheduling policies.

V. Conclusion

In this paper, a deep reinforcement learning-based data download request scheduling mechanism (*DrDre*) was designed for Internet of Drones (IoD) system, where drones submit requests to ground stations for downloading data. In order to treat all data download requests with scalability, intelligence, and efficiency, ground stations employ *DrDre*, which relies on deep Q-network (DQN), to determine the most optimal scheduling policy. Our approach *DrDre* takes into account scheduling parameters such as request deadline, request urgency, data size, request priority, and data popularity to model the fluid and ever-changing aspects of IoD systems and formulate it as a Markov Decision Process (MDP). For performance evaluation, we chose SUMO to effectively replicate drone movements and create realistic mobility traces for training at ground stations. Furthermore, we implemented *DrDre* along with other four approaches in the customized simulation environment and carried out comprehensive experiments to analyze their performance. The experimental results demonstrate that our approach *DrDre* outperforms its counterparts, making it a viable approach for scheduling data download requests in IoD systems. For future work, we aim to improve *DrDre* and enable adjacent ground stations to collaborate in fulfilling data download requests across consecutive service windows.

References

[1] L. Yu, Z. Li, N. Ansari, and X. Sun, "Hybrid Transformer Based Multi-Agent Reinforcement Learning for Multiple Unmanned Aerial Vehicle Coordination in Air Corridors," *IEEE Transactions on Mobile Computing (Early Access)*, pp. 1–14, 2025.

[2] J. Lin, B. Alkouz, A. Bouguettaya, and A. A. Safia, "Dynamic and Immersive Framework for Drone Delivery Services in Skyway Networks," *ACM Transactions on Internet Technology*, pp. 1–30, 2025.

[3] I. Bhattarai, C. Pu, K. Choo, and D. Korać, "A Lightweight and Anonymous Application-Aware Authentication and Key Agreement Protocol for the Internet of Drones," *IEEE Internet of Things Journal*, vol. 11, no. 11, pp. 19 790–19 803, 2024.

[4] *NASA's Advanced Air Mobility Mission*, Last accessed: February 3, 2025, https://www.nasa.gov/mission/aam/.

[5] C. Pu and L. Carpenter, "Psched: A Priority-Based Service Scheduling Scheme for the Internet of Drones," *IEEE Systems Journal*, vol. 15, no. 3, pp. 4230–4239, 2021.

[6] A. Ebrie and Y. Kim, "Reinforcement learning-based optimization for power scheduling in a renewable energy connected grid," *Renewable Energy*, vol. 230, p. 120886, 2024.

[7] Q. Meng, S. Hussain, F. Luo, Z. Wang, and X. Jin, "An Online Reinforcement Learning-Based Energy Management Strategy for Microgrids With Centralized Controll," *IEEE Transactions on Industry Applications*, vol. 61, pp. 1501–1510, 2024.

[8] L. Ye, L. Yang, Y. Xia, and X. Zhao, "A Cost-Driven Intelligence Scheduling Approach for Deadline-Constrained IoT Workflow Applications in Cloud Computing," *IEEE Internet of Things Journal*, vol. 11, pp. 16 033 – 16 047, 2024.

[9] T. Zhang, K. Lam, and J. Zhao, "Device Scheduling and Assignment in Hierarchical Federated Learning for Internet of Things," *IEEE Internet of Things Journal*, vol. 11, pp. 18 449 – 18 462, 2024.

[10] S. Seifhosseini, M. Shirvani, and Y. Ramzanpoor, "Multi-objective cost-aware bag-of-tasks scheduling optimization model for IoT applications running on heterogeneous fog environment," *Computer Networks*, vol. 240, p. 110161, 2024.

[11] L. Wu, H. Lin, and X. Wang, "Federated Training Generative Adversarial Networks for Heterogeneous Vehicle Scheduling in IoV," *IEEE Internet of Things Journal*, pp. 1 – 1, 2024.

[12] S. Safavat and D. Rawat, "Energy-efficient resource scheduling using x-cnn and cd-sbo for sdn based mec enabled iov," in *Proc. IEEE CCNC*, 2023, pp. 411–416.

[13] J. Tian, D. Wang, H. Zhang, and D. Wu, "Service Satisfaction-Oriented Task Offloading and UAV Scheduling in UAV-Enabled MEC Networks," *IEEE Transactions on Wireless Communications*, vol. 22, no. 12, pp. 8949–8964, 2023.

[14] M. Alsheikh, D. Hoang, D. Niyato, H. Tan, and S. Lin, "Markov decision processes with applications in wireless sensor networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1239–1267, 2015.

[15] V. Mnih *et al.*, "Human-Level Control Through Deep Reinforcement Learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[16] A. Bandyopadhyay, V. Mishra, S. Swain, K. Chatterjee, S. Dey, S. Mallik, A. Al-Rasheed, M. Abbas, and B. Soufiene, "Edgematch: A smart approach for scheduling iot-edge tasks with multiple criteria using game theory," *IEEE Access*, vol. 12, pp. 7609–7623, 2024.