To Risk or Not to Risk: Learning with Risk Quantification for IoT Task Offloading in UAVs

Anne Catherine Nguyen†, Turgay Pamuklu†, *Member, IEEE*, Aisha Syed‡,
W. Sean Kennedy‡, Melike Erol-Kantarci†, *Senior Member, IEEE*
†*School of Electrical Engineering and Computer Science, University of Ottawa*, Ottawa, Canada
‡*Nokia Bell Labs*
Emails:{anguy087, turgay.pamuklu, melike.erolkantarci}@uottawa.ca, {aisha.syed, sean.kennedy}@nokia-bell-labs.com

Abstract—A deep reinforcement learning technique is presented for task offloading decision-making algorithms for a multi-access edge computing (MEC) assisted unmanned aerial vehicle (UAV) network in a smart farm Internet of Things (IoT) environment. The task offloading technique uses financial concepts such as cost functions and conditional variable at risk (CVaR) in order to quantify the damage that may be caused by each risky action. The approach was able to quantify potential risks to train the reinforcement learning agent to avoid risky behaviors that will lead to irreversible consequences for the farm. Such consequences include an undetected fire, pest infestation, or a UAV being unusable. The proposed CVaR-based technique was compared to other deep reinforcement learning techniques and two fixed rule-based techniques. The simulation results show that the CVaR-based risk quantifying method eliminated the most dangerous risk, which was exceeding the deadline for a fire detection task. As a result, it reduced the total number of deadline violations with a negligible increase in energy consumption.

Index Terms—Risk quantification, Unmanned Aerial Vehicles, Deep Reinforcement Learning, Smart Farm

I. Introduction

Wireless technology has expanded the horizon of the agriculture industry. It has enabled more efficient and precise farming techniques with the deployment of the Internet of Things (IoT) devices and wireless connectivity. Farmers are now able to monitor their farmlands using sensors and cameras that relay back to their real-time status updates. Through image classification, IoT devices can quickly identify changes in environments and potential risks. Qazi et al. [1] highlighted the latest advancements in IoT technologies and artificial intelligence used in smart farms. The authors identified that with the assistance of artificial intelligence, smart farm networks are able to self-manage their resources in order to maintain efficient performance.

Furthermore, energy-aware reinforcement learning solutions are recently proposed for many wireless network problems. Mollahasani et al. [2] proposed actor-critic-based learning to reduce energy consumption in an open radio access network architecture. Khoramnejad et al. [3] addressed energy consumption and quality of service performance metrics in 5G networks by using a multi-agent double deep Q-network. Pamuklu et al. [4] included solar panels in their architecture as an alternative energy source for reducing the dependency on grid energy. Their reinforcement learning solution also improved the cost efficiency to improve the economic feasibility. Energy-aware-based machine learning (ML) solutions are also attracting interest in unmanned aerial vehicle (UAV) based wireless networks recently [5]. Sun et al. [6] modeled their age of information and energy-centric problem as a Markov decision process and then solved it with a policy gradient-based machine learning algorithm.

Task offloading is another emerging topic in UAV-based networks. Ebrahim et al. [7] provided a tradeoff between delay and energy by optimizing offloading decisions using deep reinforcement learning (RL). Sacco et al. [8] aimed to reduce the required information and training time for an RL-based task offloading solution. Zhao et al. [9] studied on multi-agent TD3 approach to address their continuous action space problem. Yang et al. [10] focused on service cost minimization by optimizing the joint task offloading and time-division multiple access-based channel allocation decisions.

In our previous works, we also studied task offloading problems in UAV-based smart agriculture IoT networks. First, we proposed a tabular Q-Learning approach [11], and then a deep RL version [12] for larger networks. In addition to these studies, we proposed a risk-sensitive solution [13] to isolate critical key performance indicators; thus, we can control the tradeoff between multi objectives while training our tabular ML model.

Unlike our prior works, we also looked into risk quantification techniques. In finance, there are several methods used to measure risks, such as the Sharpe ratio, the Sortino ratio, and conditional value at risk (CVaR). These measurements are used to evaluate the riskiness of investments based on their expected losses [14]. The Sharpe ratio [15] measures the risk of an investment portfolio by dividing the increase in return by the volatility of the portfolio. First, it subtracts the return of a known safe investment from the portfolio's expected return. Then the difference between the two portfolios is divided by the standard deviation of the first portfolio's returns. The difference between the two portfolios indicates the potential increase in return, and the standard deviation of the investment's return indicates the volatility of the returns. The Sortino ratio [16] is an extension of the Sharpe ratio, but it has a different denominator. Instead of dividing by the standard deviation of all the portfolio's returns, it uses the standard deviation

Fig. 1. The network model for risk quantifying approach.

of the portfolio's negative returns. CVaR was introduced by Rockafellar and Uryasev as a way to evaluate investment portfolios based on their expected shortfalls [14]. CVaR is the mean of the lowest α percentile of the portfolio's returns. This enables investors to say that the portfolio's expected returns are better than the CVaR $(1 - \alpha)$ percent of the time.

In wireless networks, financial concepts like risk can be used to evaluate and model behaviors to find an algorithm that avoids undesirable behaviors. Apandi et al. [17] use the Sharpe ratio as a decision-making parameter for base stations to use in order to find an optimal way to find user associations in order to send subcarriers. Zhou et al. [18] use CVaR to measure the average of the worst age of information times for IoT status updates at the receiver in an IoT monitoring system. The CVaR of the receiver's age of information was used as one of the considerations in their optimization algorithm. Unlike this paper, we use CVaR to measure the potential damages after certain dangerous events happen in a smart farm.

The rest of this paper is organized as follows: Sections II and III describe the system model and the proposed CVaR-based risk quantifying approach, respectively. Then, we present the performance evaluation of this method in Section IV and conclude the paper with Section V.

II. SYSTEM MODEL

Our smart farm scenario consists of a set of X IoT devices deployed across large farmland. These IoT devices perform real-time intensive image classification tasks in order to provide up-to-date updates on the farm. They can perform K different types of tasks such as fire detection, pest detection, and crop growth monitoring. One of the limitations of IoT devices being used on farms is their finite battery and computing capacity. Accurate image classification tasks require rigorous computation because they need to use deep neural networks (DNNs). If the IoT devices perform all the image classification tasks themselves, they will run out of batteries very quickly while trying to meet the heavy computing resource demands of the image classification algorithms. In addition, some of these tasks need to be done within a certain time frame; therefore they have deadlines.

In order to relieve some of the demands of the IoT devices, we have J unmanned aerial vehicles (UAV) deployed over the farmlands. A UAV device (j) contains a computing resource that is capable of performing an image classification task $\langle j, t \rangle$, locally, which is generated in time interval t. Also, a UAV provides connectivity to other UAVs or multi-access edge computing (MEC) devices ($\mathcal{L}$) where this task can be performed. With the addition of these computing resources ($J^+ = \mathcal{J} \cup \mathcal{L}$), IoT devices can offload their tasks to nearby UAVs. Upon receiving a task, the UAVs can choose to compute the task themselves, offload to another UAV, or offload to a MEC server. The UAVs also suffer from having a finite battery capacity $\Upsilon_{j'}^B$. Their current battery level at time $\mathcal{T}$, ($\Upsilon_{j'}^R$) can be modeled using the following equation,

$$
\begin{aligned}
\Upsilon_{j'}^R = \Upsilon_{j'}^B - (\Upsilon_{j'}^H + \Upsilon_{j'}^A + \Upsilon_{j'}^I) * \mathcal{T} \\
- \sum_{\substack{j \in \mathcal{J} \\ t \in \mathcal{T} \\ t' \in \mathcal{T}}} (\Upsilon_{j'}^C - \Upsilon_{j'}^I) * p_{\langle j,t \rangle j' t'},
\end{aligned} \tag{1}
$$

where $\Upsilon_{j'}^B$ is the initial battery level for UAV j', $\Upsilon_{j'}^H$ is the amount of energy required for the UAV to fly above the farm, $\Upsilon_{j'}^A$ is the amount of energy required by the antenna to send signals, $\mathcal{T}$ is the total simulation time, $\Upsilon_{j'}^I$ is the amount of energy required for the CPU to be idle, and $\Upsilon_{j'}^C$ is the amount of energy required for the CPU to run a task. $p_{\langle j,t \rangle j' t'}$ is a binary indicator used to indicate that UAV j' computed the task $\langle j, t \rangle$ at time t'. Computing a task locally will lead to some of their battery power being consumed; however, if the current UAV has too many tasks to compute, the task's queuing time will increase, which can lead to a deadline violation occurring $\mathbb{E}(v_{\langle j,t \rangle}) = 1$ because the task's processing time exceeds the task's deadline.

One of the objectives of the proposed methods is to extend the UAV's battery life. This will enable the UAV to hover for as long as possible. The next objective is to minimize the average mean uplink delay for task completion. By the uplink delay, we are reducing the chances of a deadline violation occurring. Not meeting the tasks' deadline $\gamma_{\langle j,t \rangle}^D$ can have severe consequences. For example, if the pests are not detected in time, a significant portion of the crops may be destroyed. These consequences can be described as risky behavior that we would like to avoid. We are proposing two methods that will make decisions on behalf

of the UAVs while considering the consequences of the risks during the decision-making process.

III. RISK QUANTIFYING (RQ) METHOD

A. Deep Risk Sensitive Learning with Risk Measurement

Deep reinforcement learning uses deep neural networks to generate Q-values for each action instead of Q-tables. These Q-values indicate to the agent the action that will lead to the best long-term gain. The agent then selects the action. As the agent goes through a series of actions and encounters different scenarios, the neural networks train themselves with new experiences to get more precise in predicting the Q-values. In addition, the neural networks allow a larger state space without being bound by physical disk space like the tabular Q-learning method. This enabled us to include more environmental information in the state such as the transmission delays between all of the computing resources. When dealing with a multi-objective problem, we need to juggle many risks at the same time and evaluate each risk. We need a measure of risk in order to properly quantify each risk so that we can determine the correct course of action. In the field of finance, there are several methods of evaluating risk in order to select the best investment. Our approach to evaluating the performance of each objective is based on the CVaR calculation.

The CVaR method is particularly applicable to our smart farm scenario because we have several different types of risks that can occur at the same time throughout monitoring the farm. Each risk can lead to different levels of damage. For example, when you miss the deadline of a fire recognition task by one second, it will cause some containable damage. The longer you exceed the deadline, the amount of damage will increase drastically and become unmanageable. Similarly, when the UAV is operating at full battery capacity, performing a task locally does not do a lot of damage since it will only consume a small percentage of the battery power. However, when the UAV's battery is almost depleted, performing a task locally now becomes more dangerous because it may cause the UAV to stop working. The potential damage of each known risk factor can be modeled according to the objectives listed in (2) as a function of time, and the sum of all the risk factors can be viewed similarly to how we view the potential returns of an investment portfolio. Using CVaR will allow us to train the agent to avoid actions that will lead to the worst damages.

Instead of only considering the returns' lowest α percentile, we consider the mean of both the highest and lowest $\alpha/2$-percentile. We found that if you only use CVaR, you are only avoiding the worst behavior. The agent only learns not to do actions that lead to the worst damages; however, it will still perform actions that lead to damages that are slightly better than the worst damages. When you include the upper $\alpha/2$-percentile in the evaluation, you can train the agent to learn good behavior along with avoiding bad ones.

Unlike the deep risk-sensitive approach presented in Section III-B4, this approach does not have a separate state to evaluate the known risk. Also, it does not consider all deadline violations

to be the same. It uses cost functions to evaluate the value of each risk based on the potential damages.

1) Objective: The objectives of this problem are similar to the objectives highlighted in [13] where we would like to minimize the hovering time of the UAV network ($\Upsilon_{j'}^R$), and minimize the mean uplink delay (δ). Instead of restricting the total number of deadline violations, we would like to minimize the total number of deadline violations that will lead to severe consequences $E(v_{\langle j,t \rangle}) = 1$. For example, if we exceed the deadline for a fire identification task, the consequences from this deadline violation will lead to more damage than if the deadline for a growth monitoring task is exceeded.

Maximize:

$$W * \min_{j' \in J} \Upsilon_{j'}^R - \frac{1-W}{2*\Theta^M}\delta - \frac{1-W}{2*\Theta^D} \sum_{\langle j,t \rangle \in \langle \mathcal{J},\mathcal{T} \rangle} \mathbb{E}(v_{\langle j,t \rangle}) \quad (2)$$

2) Risk Cost Functions: The cost functions determine the potential cost of damages an action can inflict on the smart farm. The costs for each risk is evaluated after every action, and the sum of all the individual cost forms the total cost.

- Energy Risk Cost ($\mathcal{C}_e$): As the UAV's current battery power after the action ($\Upsilon_{j_a}^R$) goes down, the situation becomes more serious. This is reflected in exponential growth (g) seen in (3).

$$\mathcal{C}_e(\Upsilon_{j_a}^R) = (1 - \Upsilon_{j_a}^R)^g \quad (3)$$

- End-to-end Delay Risk Cost ($\mathcal{C}_d$): The end-to-end delay risk is a function of the task's end-to-end delay ($\Delta_{\langle j,t \rangle}$). The end-to-end delay can be defined as

$$\Delta_{\langle j,t \rangle} = \Delta_I^{j_R} + \Delta_{j_R}^{j_P} + \Upsilon_{\langle j,t \rangle}^Q + \Upsilon_{\langle j,t \rangle}^P, \quad (4)$$

where $\Delta_I^{j_R}$ is the transmission delay from the IoT device (I) to the UAV that first received the task (j_R), $\Delta_{j_R}^{j_P}$ is the transmission delay from j_R to the computing resource that will compute the task j_P, $\Upsilon_{\langle j,t \rangle}^Q$ is the time the task spends in j_P's queue, and $\Upsilon_{\langle j,t \rangle}^P$ is the time it takes for the task to be computed at by j_P. The ideal situation is when the end-to-end delay is less than the task's deadline ($\gamma_{\langle j,t \rangle}^D$). This is why it will have the lowest cost; the more the end-to-end delay is lower than the deadline. If the end-to-end delay is equal to the deadline, it is still acceptable, which is why the $\mathcal{C}_d$ is 0. If the end-to-end delay is greater than the task's deadline, then the cost will be positive because there will be some damage done to the smart farm environment. The magnitude of the cost $E(\Delta_{\langle j,t \rangle})$ depends on the end-to-end delay, the deadline, and the task type. If the task's type is a fire task ($h_{\langle j,t \rangle} = 1$), the costs associated with missing this task's deadline will be more severe and, therefore, higher. The further away the end-to-end delay is from the task's deadline, the magnitude of the cost will increase by

a power of s. If the task type is not a fire task ($h_{\langle j,t\rangle} = 0$), the magnitude of the cost will increase by a power of w.

$$\mathcal{C}_d(\Delta_{\langle j,t\rangle}) = \begin{cases} -E(\Delta_{\langle j,t\rangle}) \text{ if } \Delta_{\langle j,t\rangle} < \gamma^D_{\langle j,t\rangle} \\ 0 \text{ if } \Delta_{\langle j,t\rangle} = \gamma^D_{\langle j,t\rangle} \\ E(\Delta_{\langle j,t\rangle}) \text{ if } \Delta_{\langle j,t\rangle} > \gamma^D_{\langle j,t\rangle} \end{cases} \tag{5}$$

Where: $E(\Delta_{\langle j,t\rangle}) = h_{\langle j,t\rangle}|(\gamma^D_{\langle j,t\rangle} - \Delta_{\langle j,t\rangle})|^s$
$$+ (1 - h_{\langle j,t\rangle})|(\gamma^D_{\langle j,t\rangle} - \Delta_{\langle j,t\rangle})|^w \tag{6}$$

- Total Cost ($\mathcal{C}_T$): Total cost is the sum of the energy risk cost ($\mathcal{C}_e$) and the end-to-end delay cost ($\mathcal{C}_d$). Γ is the scaling factor for the energy cost.

$$\mathcal{C}_T = \Gamma * \mathcal{C}_e + \mathcal{C}_d \tag{7}$$

3) Risk Measurement: After every action, the total cost of that action ($\mathcal{C}_{T_a}$) is calculated and we form an ordered list of costs ($\mathbb{Z}_{j_a}$) for all actions performed by agent j_a. We assume that agent j_a has performed a total of n actions at time t (in seconds). Let $\mathbb{Z}_{L_{j_a}}$ be a list that contains the lowest $\frac{\alpha}{2}\%$ of n elements in $\mathbb{Z}_{j_a}$. Let $\mathbb{Z}_{H_{j_a}}$ be a list that contains the highest $\frac{\alpha}{2}\%$ of n elements in $\mathbb{Z}_{j_a}$. The risk measurement (η) for agent j_a after action a is defined as

$$\eta = \frac{\sum \mathbb{Z}_{L_{j_a}} + \sum \mathbb{Z}_{L_{j_a}}}{\frac{\alpha}{100} * n}. \tag{8}$$

4) Markov Decision Process Definition:

- **State:** The state includes the task type k, the CPU delays of all computing resources in UAVs and MECs $\Delta_{j' \in \mathcal{J}^+}$, all UAVs' current battery levels $\Upsilon^L_{j' \in \mathcal{J}}$, all transmission delays between computing resources $\Delta^{j_P \in \mathcal{J}^+}_{j_R \in \mathcal{J}^+}$, and the UAV that first received the task j_i. The state can be defined as

$$\mathbb{S} = \{k, \Delta_{j' \in \mathcal{J}^+}, \Upsilon^L_{j' \in \mathcal{J}}, \Delta^{j_P \in \mathcal{J}^+}_{j_R \in \mathcal{J}^+}, j_i\}. \tag{9}$$

- **Action:** Each UAV will act as an independent agent. Upon receiving a task, an agent must select a destination j' which is a computing resource, out of all the computing resources available J^+, where the task will be computed. The agent can choose to do the task on the local computing resource or send the task to a nearby UAV or MEC server. Therefore, the set of all possible actions is defined as

$$\mathbb{A} = \{x_{j' \in J^+}\}. \tag{10}$$

The reward function $\mathbb{R}_{RQ}$ is defined as

$$\mathbb{R}_{RQ} = (1 - \beta)RV_c + \beta RV_m, \tag{11}$$

where RV_c is the reward value of the current UAV, RV_m is the reward value of the UAV with the highest costs, and β is the degree to which we are considering the UAV with the highest costs. The values for RV_m and RV_c are defined using (12), and depend on the UAV's CVaR value before (η_p) and after (η_a) the action was taken. If $\eta_a < \eta_p$, then the action caused the average risk cost to decrease, and the agent will be rewarded with the

highest reward. If $\eta_a < \eta_p$, then the action caused the average risk cost to increase which means that the agent is engaging in actions that are significantly increasing battery consumption, uplink mean delay, or exceeding the deadline violation of a dangerous task. Therefore, the reward value is negative.

$$RV = \begin{cases} 20, & \text{if } (\eta_a < \eta_p) \wedge (\eta_a < 0) \\ 10, & \text{if } (\eta_a < \eta_p) \wedge (\eta_a = 0) \\ 1, & \text{if } (\eta_a < \eta_p) \wedge (\eta_a > 0) \\ 5, & \text{if } (\eta_a = \eta_p) \wedge (\eta_a < 0) \\ 2, & \text{if } (\eta_a = \eta_p) \wedge (\eta_a = 0) \\ -5, & \text{if } (\eta_a = \eta_p) \wedge (\eta_a > 0) \\ 1, & \text{if } (\eta_a > \eta_p) \wedge (\eta_a < 0) \\ -1, & \text{if } (\eta_a > \eta_p) \wedge (\eta_a = 0) \\ -10, & \text{if } (\eta_a > \eta_p) \wedge (\eta_a > 0) \end{cases} \tag{12}$$

B. Baseline Methods

1) Round Robin (RR): In the round robin, there is a fixed order to determine the destination of the task. All computing resources are placed on an ordered list, and the task's destination is determined by knowing the previous task's destination and sending the current task to the next destination on the ordered list.

2) Lowest Queue Time and Highest Energy First (QHEF): Similar to the RR algorithm, QHEF uses fixed rules to determine the destination of the task. The algorithm determines the destination where the task will be computed by first finding the destination with the lowest queuing time. The next step is to find the destination with the highest energy level, out of all the computing resources with the lowest queue time. If the computing resource with the highest energy level and lowest queue time is higher than the receiving UAV's current energy level, then the task will be offloaded to another destination. If not, then the receiving UAV will compute the task locally.

3) Deep Q-Learning: We used the Deep Q-Learning algorithm presented by Nguyen et al. [12]. The Deep Q-Learning algorithm uses the action set defined in (10), the state defined in (9) and the epsilon-greedy policy. The reward function defined in (13) consists of two subparts: the reward for choosing an action that does not increase energy consumption ($\Upsilon^L_{j_a} - 1$) over threshold e, and the reward for selecting a destination that did not lead to a deadline violation $(1 - \mathbb{E}(v_{j_a})) + \mathcal{V}^L_{j_a} * \mathbb{E}(v_{j_a})$. Where $\mathcal{V}^L_{j_a}$, determines the magnitude of the negative reward if a deadline violation occurred when it could have been prevented if the task had been sent to the MEC server ($\mathbb{E}(v_{j_m}) = 0$), the task was performed locally ($\mathbb{E}(v_{j_r}) = 0$), the task was performed by another UAV ($\exists j' \in (\mathcal{J}/(j_r \cup j_a))(\mathbb{E}(v_{j'})) = 0$), or a deadline violation was inevitable.

$$\mathbb{R} = (\Upsilon^L_{j_a} - 1) + (1 - \mathbb{E}(v_{j_a})) + \mathcal{V}^L_{j_a} * \mathbb{E}(v_{j_a}) \tag{13}$$

$$\Upsilon^L_{j_a} = \begin{cases} 2, & \text{if } \mathbb{E}(\Upsilon^R_{j_a}) - \max_{j' \in \mathcal{J}}(\mathbb{E}(\Upsilon^R_{j'})) \geq -e \\ 0, & \text{if } \mathbb{E}(\Upsilon^R_{j_a}) - \max_{j' \in \mathcal{J}}(\mathbb{E}(\Upsilon^R_{j'})) \leq -2e \\ 1, & \text{otherwise}, \end{cases} \tag{14}$$

Fig. 2. Reward convergence for RQ method.

Fig. 3. Remaining battery levels for all UAVs across the network.

$$\mathcal{V}_{j_a}^L = \begin{cases} -40, & \text{if } \mathbb{E}(v_{j_m}) = 0 \\ -20, & \text{if } \mathbb{E}(v_{j_r}) = 0 \\ -10, & \text{if } \exists j' \in (\mathcal{J}/(j_r \cup j_a))(\mathbb{E}(v_{j'})) = 0 \\ -1, & \text{otherwise.} \end{cases} \quad (15)$$

4) Deep Risk Sensitive (DRS) Reinforcement Learning: We extended Pamuklu et al.'s work in [13] by adding deep learning to their risk-sensitive approach. Instead of using Q-tables to predict the risk and reward Q-values, we used deep neural networks. Deep learning enabled us to extend the state space to include transmission delays between IoT devices and UAVs, UAVs to UAVs, and UAVs to MEC. The state space is defined in (9). The action set is defined in (10). For each action, the reward and risk is evaluated separately. The reward evaluates how close the agent's action came to achieving objective P2 in [13]. It is defined as follows,

$$\mathbb{R}_{DRS} = -\left[W * (\Upsilon_{j_a}^L - 1) - \frac{1 - W}{\Theta^M} * \Delta_{j_a} \right], \quad (16)$$

where the $(\Upsilon_{j_a}^L - 1)$ portion is proportional to how the action impacted the battery level of the computing resource that computed the task. Δ_{j_a} is the mean uplink delay of the computing resource a after the task was added to its queue.

The risk function can be defined as

$$\mathbb{C} = \begin{cases} -\mathcal{V}_{j_a}^L, & \text{if state} \in \mathbb{RS} \\ -1, & \text{otherwise} \end{cases} \quad (17)$$

Where: $\mathbb{RS} = \{r | r \subset \mathbb{S}, \text{ and } \mathbb{E}(v_{j_a})\}$

It addresses the risk constraint defined in the objective function [13]. If the agent reaches a risk state, then the agent will be penalized. The magnitude of that penalization is determined by $\mathcal{V}_{j_a}^L$. If a deadline could have been avoided, then the magnitude of $\mathcal{V}_{j_a}^L$ is high, otherwise, it is low.

IV. PERFORMANCE EVALUATION

In order to simulate our MEC-aided UAV smart farm network, we used Simu5G [19]. It is a 5G network simulation

library that runs on a network simulator platform called Omnet++ [20]. Our simulation network consisted of sixteen IoT devices ($X = 16$), four UAVs ($J = 4$), and one MEC server ($L = 1$). The IoT devices generated three types of tasks ($K = 3$), fire detection, pest detection, and growth monitoring. Table I shows the parameters that we used for each task during our simulation. The tasks' interarrival time ($1/\lambda$) followed an exponential distribution. Each task had a unique deadline ($\gamma_{\langle j,t \rangle}^D$), and the deadlines for all fire detection, pest detection and growth monitoring tasks were 1s, 2s, and 15s respectively. The MEC processing times for each task ($\gamma_{\langle j,t \rangle}^P$ (MEC)) were lower than the UAVs' processing times ($\gamma_{\langle j,t \rangle}^P$ (UAV)) because we assumed that the MEC server had more computing resources and therefore will compute the tasks faster.

For the energy consumption, the following parameters were used with (1), UAVs with index 0 and 1 had full battery capacity ($\Upsilon_{j'}^B$) of 570 watt-hours, meanwhile, UAVs with index 2 and 3's full battery capacity was 627 watt-hours. Hovering ($\Upsilon_{j'}^H$) consumed 211 watt-hours. The antenna ($\Upsilon_{j'}^A$) needed 17 watt-hours. In order to operate the CPU[1] we needed 4320 watt-hours if the CPU is idle ($\Upsilon_{j'}^I$) and 12960 watt-hours if the CPU is computing a task ($\Upsilon_{j'}^C$).

We used the following reinforcement learning parameters, a learning rate of 0.05 and a discount factor of 0.85. Our simulation time was 5s ($\mathcal{T} = 5$). We trained our model using a dataset of 100 recorded simulations. In terms of the cost function parameters for our Risk Quantifying method, we used an energy risk growth value of 2 ($g = 2$), fire task end-to-end delay risk growth value of 8 ($s = 8$), and other task risk growth value of 1 ($w = 1$). For the risk measurement, we used an α value of 2. In RQ's reward function, $\beta = 0.75$.

---

[1] Idle and running CPU energy consumptions are selected based on limited simulation time. Thus, UAVs that will operate for ten hours can be simulated, and the difference in energy performance between the methods can be tested.

Fig. 4.   Deadline violation distribution per task type.

Fig. 5.   Uplink delay distribution per node.

TABLE I
SIMULATION TASKS' PARAMETERS FROM [13]

Task Type	$(1/\lambda)$	$\gamma^D_{\langle j,t \rangle}$	$\gamma^P_{\langle j,t \rangle}$ (UAV)	$\gamma^P_{\langle j,t \rangle}$ (MEC)
Fire detection	0.25s	1s	0.1s	0.05s
Pest detection	0.25s	2s	0.2s	0.1s
Growth monitoring	0.5s	15s	1.5s	0.75s

A. Simulation Results

Every key performance indicator plot shows the average results over 10 runs with different seeds.

1) Convergence: Fig. 2 demonstrates the cumulative reward for the proposed RQ method after 10000 episodes. We can see that the reward converges after 8750 episodes. Each UAV is an independently trained agent and has its own deep neural network to predict the most accurate Q-value. Despite being independent of one another, they all have similar performances and converged towards the same value at the end of the training period.

2) Remaining Energy Level: The remaining energy level is a KPI used to determine how long the UAV network can remain hovering. A high remaining energy level signifies that the UAV can hover for a longer period of time, whereas a low remaining energy level indicates that the UAV will not hover for a long period of time. In Fig. 3, QHEF had the highest minimum remaining energy level at 97.04%, and DQL was not far behind at 96.97%. RQ's minimum remaining energy level was 96.65% which is 0.39 lower than QHEF, meaning the difference between their performances is not significant.

3) Deadline Violations: A deadline violation occurs when the uplink delay exceeds the task's deadline. The RQ had the lowest percentage of deadline violations. The ML algorithms outperformed all other algorithms because they made deadline violation a focus in their objective functions. In Fig. 4, we can see that the RQ method was able to outperform the deep risk-sensitive method because it was able to eliminate fire task

deadline violations. For all the other methods, the fire task made up most of their deadline violations.

4) Mean Uplink Delay: The uplink delay is the sum of all the transmission delays involved with sending the task to the destination computing resource, the queuing time, and CPU processing time. Fig. 5 illustrates the uplink delay for each algorithm across every node in the network. We can see that the RQ method had the lowest average uplink delay at around 0.6 seconds. This is because it has a cost function that considers the end-to-end delay, and the more the uplink delay exceeds the task's deadline, the higher the cost. This would then lead to the agent's risk costs increasing. The ML algorithms outperformed the heuristic algorithms in uplink delay because they could self-adjust their destination-finding rules and adapt to new situations; meanwhile, the RR and QHEF algorithms had fixed rules.

V. CONCLUSION

In this study, we presented machine learning algorithms with awareness of risk quantification in an IoT-aided smart farm setting. We used two different techniques to identify risk behaviors in order to train the agent to avoid such behaviors that can lead to severe consequences. The first technique was an extension of our previous study, where we had a separate state to identify whether the agent had entered a risky situation. The second technique used cost functions to evaluate the potential damages of each action and used a risk parameter based on CVaR in order to evaluate the agent's decision-making. We compared our techniques with another deep RL technique and two non-machine learning heuristic algorithms and saw that the risk parameter technique (RQ) was able to completely avoid the risky behavior which is exceeding the deadline on a fire recognition task. This, in turn, reduced the total number of deadline violations and the mean uplink delay. In the future, we would like to include multi-UAV trajectory planning into our problem.

REFERENCES

[1] S. Qazi, B. A. Khawaja, and Q. U. Farooq, "IoT-equipped and AI-enabled next generation smart agriculture: A critical review, current challenges and future trends," *IEEE Access*, vol. 10, pp. 21219–21235, 2022.

[2] S. Mollahasani, T. Pamuklu, R. Wilson, and M. Erol-Kantarci, "Energy-aware dynamic DU selection and NF relocation in O-RAN using actor–critic learning," *Sensors*, vol. 22, p. 5029, Jul 2022.

[3] F. Khoramnejad, R. Joda, A. B. Sediq, H. Abou-Zeid, R. Atawia, G. Boudreau, and M. Erol-Kantarci, "Delay-Aware and Energy-Efficient Carrier Aggregation in 5G using Double Deep Q-Networks," *IEEE Transactions on Communications*, vol. 70, no. 10, pp. 6615–6629, 2022.

[4] T. Pamuklu, M. Erol-Kantarci, and C. Ersoy, "Reinforcement learning based dynamic function splitting in disaggregated green open RANs," in *ICC 2021 - IEEE International Conference on Communications*, Jun 2021.

[5] M. Abrar, U. Ajmal, Z. M. Almohaimeed, X. Gui, R. Akram, and R. Masroor, "Energy efficient UAV-enabled mobile edge computing for IoT devices: A review," *IEEE Access*, vol. 9, pp. 127779–127798, 2021.

[6] M. Sun, X. Xu, X. Qin, and P. Zhang, "AoI-energy-aware UAV-assisted data collection for IoT networks: A deep reinforcement learning method," *IEEE Internet of Things Journal*, vol. 8, no. 24, pp. 17275–17289, 2021.

[7] M. A. Ebrahim, G. A. Ebrahim, H. K. Mohamed, and S. O. Abdellatif, "A deep learning approach for task offloading in multi-UAV aided mobile edge computing," *IEEE Access*, vol. 10, pp. 1–1, Sep 2022.

[8] A. Sacco, F. Esposito, G. Marchetto, and P. Montuschi, "A self-learning strategy for task offloading in UAV networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 4, pp. 4301–4311, 2022.

[9] N. Zhao, Z. Ye, Y. Pei, Y. C. Liang, and D. Niyato, "Multi-agent deep reinforcement learning for task offloading in UAV-assisted mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 21, no. 9, pp. 6949–6960, 2022.

[10] C. Yang, B. Liu, H. Li, B. Li, K. Xie, and S. Xie, "Learning based channel allocation and task offloading in temporary UAV-assisted vehicular edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 9, pp. 9884–9895, 2022.

[11] A. C. Nguyen, T. Pamuklu, A. Syed, W. S. Kennedy, and M. Erol-Kantarci, "Reinforcement learning-based deadline and battery-aware offloading in smart farm IoT-UAV networks," in *ICC 2022 - IEEE International Conference on Communications*, pp. 189–194, 2022.

[12] A. C. Nguyen, T. Pamuklu, A. Syed, W. S. Kennedy, and M. Erol-Kantarci, "Deep reinforcement learning for task offloading in UAV-aided smart farm networks," in *IEEE FNWF*, Sep 2022.

[13] T. Pamuklu, A. C. Nguyen, A. Syed, W. S. Kennedy, and M. Erol-Kantarci, "IoT-aerial base station task offloading with risk-sensitive reinforcement learning for smart agriculture," *IEEE Transactions on Green Communications and Networking*, pp. 1–1, 2022.

[14] R. T. Rockafellar, S. Uryasev, *et al.*, "Optimization of conditional value-at-risk," *Journal of risk*, vol. 2, pp. 21–42, 2000.

[15] W. F. Sharpe, "The sharpe ratio," *The Journal of Portfolio Management*, vol. 21, no. 1, p. 49–58, 1994.

[16] F. A. Sortino and L. N. Price, "Performance measurement in a downside risk framework," *The Journal of Investing*, vol. 3, no. 3, p. 59–64, 1994.

[17] N. I. A. Apandi, S. Tian, W. Hardjawana, P. L. Yeoh, and B. Vucetic, "Sharpe ratio for joint user association and subcarrier allocation design in downlink heterogeneous cellular networks," in *IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications*, pp. 1–5, 2017.

[18] B. Zhou, W. Saad, M. Bennis, and P. Popovski, "Risk-aware optimization of age of information in the internet of things," in *International Conference on Communications*, pp. 1–6, IEEE, 2020.

[19] G. Nardini, D. Sabella, G. Stea, P. Thakkar, and A. Virdis, "Simu5g–an omnet++ library for end-to-end performance evaluation of 5G networks," *IEEE Access*, vol. 8, pp. 181176–181191, 2020.

[20] OpenSim Ltd., "What is OMNeT++?," 2019.