# IoT-Aerial Base Station Task Offloading with Risk-Sensitive Reinforcement Learning for Smart Agriculture

Turgay Pamuklu<sup>†</sup>, *Member, IEEE*, Anne Catherine Nguyen<sup>†</sup>, Aisha Syed<sup>‡</sup>, W. Sean Kennedy<sup>‡</sup>, Melike Erol-Kantarci<sup>†</sup>, *Senior Member, IEEE* <sup>†</sup>School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Canada <sup>‡</sup>Artificial Intelligence Research Lab, Nokia Bell Labs, Murray Hill, NJ, USA

Emails:{turgay.pamuklu, anguy087, melike.erolkantarci}@uottawa.ca,

{aisha.syed, william.kennedy}@nokia-bell-labs.com

Abstract-Aerial base stations (ABSs) allow smart farms to offload processing responsibility of complex tasks from internet of things (IoT) devices to ABSs. IoT devices have limited energy and computing resources, thus it is required to provide an advanced solution for a system that requires the support of ABSs. This paper introduces a novel multi-actor-based risk-sensitive reinforcement learning approach for ABS task scheduling for smart agriculture. The problem is defined as task offloading with a strict condition on completing the IoT tasks before their deadlines. Moreover, the algorithm must also consider the limited energy capacity of the ABSs. The results show that our proposed approach outperforms several heuristics and the classic Q-Learning approach. Furthermore, we provide a mixed integer linear programming solution to determine a lower bound on the performance, and clarify the gap between our risk-sensitive solution and the optimal solution, as well. The comparison proves our extensive simulation results demonstrate that our method is a promising approach for providing a guaranteed task processing services for the IoT tasks in a smart farm, while increasing the hovering time of the ABSs in this farm.

*Index Terms*—Energy Efficient Computation Offload, Unmanned aerial vehicle (UAV), Aerial base station (ABS), Smart farm, Risk-Sensitive Reinforcement Learning, Internet of Things (IoT).

#### I. INTRODUCTION

Agriculture is sensitive to environmental changes and even slight changes can alter the outcome of the crops or livestock. Smart farms use Internet of Things (IoT) sensors, such as cameras, to monitor such environmental changes. The sensors can capture images to monitor the status of hundreds of acres of land. We can then use the captured images and image recognition techniques to extract valuable information about the status of the farm such as, the presence of fire, the presence of pests, and the growth rate of the crops and livestock. Jhuria et al. trained neural networks to recognize certain diseases on crops during the growing process [1]. Dhumale et al. introduced a robot that captures images of crops, and use image processing techniques to detect pests and diseases on the crops, and spray the appropriate crops with pesticide [2]. IoT cameras have become essential wireless sensors for smart agriculture and precision farming because they provide realtime information on the status of the farm and they relay that information to a central location. With IoT sensors, farmers now have the capability to monitor multiple acres of land from one location and provide precise care to their farm.

1

Image processing is an intensive task and the central processing units on the IoT sensors may not have the capacity to perform such intensive tasks. In addition, some image processing tasks can be time-sensitive such as fire detection. In such circumstances, we can make use of aerial base stations (ABS) that can connect the sensors to much more powerful computational resources. The ABSs equipped with computational units can perform the tasks themselves, or relay the tasks to a nearby multi-access edge computing (MEC) device. This enables the task to be done before the deadline.

Because the ABSs are hovering above large remote farmlands, they must be battery operated. As previously mentioned, image processing tasks are computationaly intensive and will require large amounts of power. If the ABS does too many tasks, it can shorten the longevity of the ABS' operation time. On the other hand, without the ABS, the IoT sensors cannot send their tasks and the sensor network is no longer operational. Therefore, we aim to extend the longevity of the battery life of the ABSs. However, if the ABS does not perform any tasks in order to conserve its energy, the MEC device would have to do all of the tasks, then the tasks may not be able to meet their deadline. This may lead to detrimental consequences such as significant loss of farmland due to a slow fire detection. Therefore the network needs to have a decision making process that considers both the time-critical nature of the image processing tasks, as well as controlling the energy consumption of the ABSs in order to maximize their longevity.

Every decision comes with tradeoffs, especially in a system with multiple objectives. The network's decision making algorithm must decide which resources will process the task based on several factors such as the ABS' current energy level, and the deadline of the task. It must also prioritize between the two objectives, the ABS network's longevity, or the potential costs of not meeting the deadline. But, missing the image processing tasks' deadlines potentially can have damaging consequences, therefore the decision making process must also evaluate the risk of each possible outcome and select the optimal decisions that will minimize the risk.

Accepted Paper. DOI:10.1109/TGCN.2022.3205330. IEEE policy provides that authors are free to follow funder public access mandates to post accepted articles in repositories. When posting in a repository, the IEEE embargo period is 24 months. However, IEEE recognizes that posting requirements and embargo periods vary by funder. IEEE authors may comply with requirements to deposit their accepted manuscripts in a repository per funder requirements where the embargo is less than 24 months.



Fig. 1: Illustration of the proposed smart farm.

#### A. Main Contributions

Machine learning is a promising approach for solving wireless network optimization problems [3]–[5]. In our previous study, we presented a Q-Learning solution for a joint taskoriented delay-aware offloading as well as increased ABS hovering time [6]. Following our prior work, in this paper, we propose a risk-sensitive reinforcement learning (RL) approach to guarantee that deadlines will be met for IoT devices in our smart farm network. By using this method, we isolate the constraints from the other KPIs, and dynamically change the weight of the KPIs while training the model. The results reported in this paper prove that our risk-sensitive method significantly reduces the number of deadline violations and constrains it to a specific threshold. To the best of our knowledge, the multi-agent-based risk-sensitive RL method is proposed for the first time for a constrained MDP problem in this paper. Our contributions can be summarized as follows:

- We solved a constrained MDP problem by modifying the training phase of a machine learning (ML) algorithm for an aerial base station (ABS)-supported smart farm environment. We created a Q-table that is trained separately from the other objectives in the problem. Therefore, we could have determined the borders of a risky taskoriented constraint and adaptively changed the direction of the learning process by considering the threshold of this constraint.
- 2) We implemented this method with a multi-agent RL approach. Thus, this method can address larger networks due to the scalability capability of this approach for this risk-sensitive-based problem.
- We provided an upper-bound analysis with a MILP Solver. Therefore, we evaluated our RL approach with the different weights for our multi-objective NP-Hard problem.
- 4) We accomplished several stress tests on the proposed solution. Thus, we analyzed the robustness of the proposed method in cases of unexpected changes in the smart farm environments.

The rest of the article is organized as follows. The related work is discussed in Section II. Then, in Section III we describe our smart farm model and its problem definition. Section IV details the risk-sensitive RL approach, and Section V provides the detailed results of this approach. Finally, in Section VI, we conclude the paper.

# II. RELATED WORKS

In one of the recent unmanned aerial vehicle (UAV) based studies, Liao et al. decomposed their task offloading and resource allocation problems into two subproblems [7]. Then, they solved these two problems separately with an actor-criticbased learning algorithm and a heuristic algorithm. As a result, they demonstrated that their proposed method reduces queuing delay more than their referenced solutions. A similar problem is addressed in [8] using the successive convex approximation method. The task-oriented guaranteed delay is a main objective in our paper, and Zhao et al. also focused on this guaranteed delay problem [9]. However, they preferred the relaxation techniques to solve this hard-constrained problem. Khairy et al. also used Lagrangian relaxation for the constraints in their UAV-based communication problem [10]. Next, they solved this dual problem with a deep RL algorithm. Ghdiri et al. have a distinctive study that also targets the task deadline and the limited UAV battery problems [11]. However, their aim was to find the optimum cluster heads' locations and UAV trajectory planning. Zhang et al. also aimed at the trajectory planning problem and addressed it by proposing a Safe-DQN solution [12].

Yang et al. minimized the total energy consumption of UEs and UAVs in their wireless network [13]. They have proposed several heuristics for different subproblems of their main nonconvex problem. Another heuristic solution for a UAV network was suggested by Xu et al., where the authors optimized a broad number of critical decisions [14]. As an alternative for heuristic approaches, Yao et al. introduced a game-theoretic solution for resource allocation and the channel access problem [15]. They provided a promising method in terms of reducing the total energy consumption in their proposed network. Another energy-efficient solution is proposed in [16], where UAVs process and relay cell-edge users' tasks.

Risk-sensitive RL has attracted attention in recent wireless networks papers due to its promising performance to solve constrained problems. Khalifa et al. used this method for

TABLE I:	Summary	of the	notations.
----------	---------	--------	------------

Sets	Size	Description	
$t\in\mathcal{T}$	T	set of time intervals	
$j \in \mathcal{J}$	J	set of ABSs	
$\langle j,t\rangle \in \langle \mathcal{J},\mathcal{T}\rangle$	J, T	tuple of tasks	
$l \in \mathcal{L}$		set of MEC devices	
$j' \in \mathcal{J}^+$	J+L	set of computational resources	
$k \in \mathcal{K}$	K Damar	set of task types	
	Kange	Description	
$\alpha_{\langle j,t\rangle}^{D}$	$\{0, 1\}$	a task comes to ABS j in time interval t	
$\alpha^P_{\langle j,t\rangle}$	$\mathbb{R}$	the processing time of this task	
$\alpha_{(i,t)}^{I}$	$\mathbb{R}$	the transmission delay between IoT &	
$\langle j, c \rangle$		ABS for this task	
$\alpha^{D}_{(i,t)}$	$\mathbb{R}$	the deadline for this task	
$\Delta_{\langle j,t\rangle}^{\langle j,t\rangle}$	$\mathbb{R}$	end-to-end delay of this task	
Energy Param.	Range	Description	
$\Upsilon_j^B$	N	battery capacity in ABS $j$	
$\Upsilon_i^H$	$\mathbb{R}$	hovering energy cons. in ABS $j$	
$\Upsilon_{i}^{A}$	$\mathbb{R}$	signal trans. energy cons. in ABS $j$	
$\Upsilon_{i}^{I}$	$\mathbb{R}$	idle energy cons. in ABS $j$	
$\Upsilon_i^C$	$\mathbb{R}$	computing energy cons. in ABS $j$	
$\Upsilon_{i}^{R}$	$\mathbb{R}$	remaining energy in ABS j	
Weight Param.	Range	Description	
$\Upsilon_i^L$	$\mathbb{N}^+$	battery reward level	
$\mathcal{V}_{i}^{L}$	$\mathbb{N}$	violation reward level	
Ŵ	[0, 1]	energy consumption weight	
$\Theta^D$	R	scaling factor for deadline violation	
$\Theta^M$	$\mathbb{R}$	scaling factor for mean delay	
Variables	Domain	Description	
$x_{\langle j,t\rangle j'}$	$\{0, 1\}$	task $\langle j,t\rangle$ is processed in resource $j'$	
$p^A_{\langle i,t\rangle i't'}$	$\{0, 1\}$	the time intervals $(t')$ used to processed	
(5,-75 -		the corresponding task	
$p^{S}_{\langle i,t\rangle i't'}$	$\{0, 1\}$	the starting time interval $(t')$ for the	
E	<i>c</i>	corresponding task	
$p^{E}_{\langle j,t\rangle j't'}$	$\{0, 1\}$	the ending time interval $(t')$ for the	
100 · 2 00	(0,1)	corresponding task	
$v_{\langle j,t  angle}$	$\{0, 1\}$	the corresponding task does not finish	
		before its deadline	

targeting strict latency requirements of URLLC traffic [17]. Their results showed that they can increase the number of successfully received URLLC packets while keeping the number of lost packets lower than a threshold value. The example of another wireless network solution was suggested by Alsenwi et al., where they used their novel "conditional value at risk" approach to prevent eMBB users from getting lower data rates due to concurrent URLLC traffic in the network [18].

In their task offloading problem, Zhou et al. extended the available processing resources by adding satellites to their network [19]. Then, they proposed a centralized deep risksensitive solution to achieve their energy consumption constraint. Despite the fact that this paper has a related problem, we provide a task-oriented approach to finish the IoT tasks before their deadline. Moreover, our risk-sensitive solution is a distributed approach in which each ABS has its own intelligent local agent to make offloading decisions. To the best of our knowledge, this is the first work that proposes a multiagent-based risk-sensitive RL solution for the task offloading problem in a UAV/ABS based wireless network.

#### **III. SYSTEM MODEL AND PROBLEM FORMULATION**

In this section, we briefly describe the scenario, detail the energy consumption and delay models, specify the system constraints, and finally formulate the guaranteed deadline problem [6]. Table I summarizes the notations we use in the remainder of the article.

#### A. Scenario

Figure 1 illustrates the scenario we use in this study. We assume a time-interval-based model. In each time interval ( $t \in$  $\mathcal{T}$ ), a set of ABS  $(j \in \mathcal{J})$  devices collect the tasks from IoT devices scattered accross a smart farm. If ABS j receives a task in time interval t, the decision-maker in this ABS is informed by the  $\alpha^B_{\langle j,t\rangle} = 1$  indicator <sup>1</sup>. Then, ABS j decides on a computational resource  $(j' \in \mathcal{J}^+)$  to process the task while taking into account the type of task  $(k \in \mathcal{K})$ , and the network conditions such as the expected transmission delay to offload a task to another ABS. Furthermore, the computational resource set is not limited to just ABS devices, but also contains nearby MEC devices  $(l \in \mathcal{L})$ .

# Given Data:

- A binary indicator signifying that ABS j received a task in time interval t ( $\alpha^B_{\langle j,t\rangle}$ ). The task's processing time  $(\alpha^P_{\langle j,t\rangle})$  and deadline  $(\alpha^D_{\langle j,t\rangle})$ . • Transmission delays between IoTs and ABSs  $(\alpha^I_{\langle j,t\rangle})$ .
- Energy consumption of each component in the ABSs  $(\Upsilon_j^H, \Upsilon_j^A, \Upsilon_j^I, \Upsilon_j^C).$ • Battery capacity of each ABS  $(\Upsilon_j^B).$

# **Objectives:**

- Increasing the hovering time of ABSs. We aim to maximize the operating time before we need to recharge any of the ABSs in our smart farm.
- Reducing the mean end-to-end delay for processing of • the tasks generated by IoTs.
- Ensure that the number of deadline violations will not • exceed a certain amount.

#### **Decisions:**

• Each ABS j should independently decide the optimum resource to process the received task  $\langle j, t \rangle$ . If an ABS chooses to process that task locally, it will queue the request and begin to process it whenever its processing resource is available. Otherwise, the ABS will offload the task to another ABS or a MEC device in the smart farm.

#### B. Energy Consumption and Delay Models

In the interest of increasing the hovering time, we have to find the change in the remaining battery energy for each ABS  $(\Upsilon_{j'}^R)$  according to their decision. Eq. 1 calculates the remaining battery level where  $\Upsilon_{j'}^H$ ,  $\Upsilon_{j'}^A$ ,  $\Upsilon_{j'}^I$  are the hovering, signal transmission, and idle energy consumptions respectively. We assume these values do not change with the offloading decision, and they are fixed values during our evaluation period (T). On the other hand, computing energy

<sup>&</sup>lt;sup>1</sup>The length of a time interval is significantly small. Therefore, an ABS receives at most one task in a time interval. Hence, we may represent a task with a tuple "(j, t)" in which j is the received ABS and t is the received time interval.

consumption,  $\Upsilon_{j'}^C$ , is determined by task processing decisions  $(p_{(j,t)j't'}^A)$  which equals to one if the resource processes a task.

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

Finalizing each task before their deadline is another one of our key performance indicators (KPI). Therefore, for each task, we have to calculate the time difference between the task's generation at the IoT device and the task's completion at the computational resource. Eq. 2 formulates this in which (t') is the start time of the task at the resource. That starting time is determined by the binary decision variable  $p_{\langle j,t \rangle j't'}^S$ , which equals to one if resource j' has started to process that task at time interval t'. At any other time, it is equal to zero; thus, the summation of the first term equals t'. The other terms in this equation are t,  $\alpha_{\langle j,t \rangle}^I$ , and  $\alpha_{\langle j,t \rangle}^P$ , which are the received time of the task at ABS, the transmission delay between IoT and ABS, and the processing delay, respectively. Lastly, we calculate the mean delay with Eq. 3, in which we sum the end-to-end delays of all tasks and then divide it by the total number of tasks.

$$\Delta_{\langle j,t\rangle}^{E} = \sum_{j'\in\mathcal{J}^{+}} \sum_{t'\in\mathcal{T}} p_{\langle j,t\rangle j't'}^{S} * (t') - t + \alpha_{\langle j,t\rangle}^{I} + \alpha_{\langle j,t\rangle}^{P} \quad (2)$$
$$\delta = \frac{\sum_{\langle j,t\rangle\in\langle\mathcal{J},\mathcal{T}\rangle} \Delta_{\langle j,t\rangle}^{E} * \alpha_{\langle j,t\rangle}^{B}}{\sum_{\langle j,t\rangle\in\langle\mathcal{J},\mathcal{T}\rangle} \alpha_{\langle j,t\rangle}^{B}} \quad (3)$$

In Eq. 4,  $\alpha_{\langle j,t \rangle}^{D}$ ,  $\Delta_{\langle j,t \rangle}^{E}$ , and  $v_{\langle j,t \rangle}$  are the deadline, end-toend delay, and violation indicator of task  $\langle j,t \rangle$ , respectively. In addition, we have an M term which is a significantly large number. In the case of the deadline violation  $(\alpha_{\langle j,t \rangle}^{D} - \Delta_{\langle j,t \rangle}^{E} < 0)$ , the left side of this equation will be lower than zero. Thus, to grant Eq. 4, the right side of this equation is enforced to be lower than zero, which could be achieved by only  $v_{\langle j,t \rangle} = 1$ . Contrastingly, if no deadline violation has occurred, the left side of this equation will be higher than zero  $(\alpha_{\langle j,t \rangle}^{D} - \Delta_{\langle j,t \rangle}^{E} \ge 0)$ . Hence,  $v_{\langle j,t \rangle}$  may equal either to zero or one. Therefore, Eq. 5 will model that case. In this equation, if we can finish a task before its deadline  $(\alpha_{\langle j,t \rangle}^{D} - \Delta_{\langle j,t \rangle}^{E} \ge 0)$ , the left side will be higher than zero. Thus, the right side of this equation should also be higher than zero, which can only be realized by  $v_{\langle j,t \rangle} = 0$ .

$$\alpha^{D}_{\langle j,t\rangle} - \Delta^{E}_{\langle j,t\rangle} \ge -M * v_{\langle j,t\rangle}, \qquad \forall \langle j,t\rangle \in \langle \mathcal{J}, \mathcal{T} \rangle \quad (4)$$

$$\alpha^{D}_{\langle j,t\rangle} - \Delta^{E}_{\langle j,t\rangle} < M * (1 - v_{\langle j,t\rangle}), \quad \forall \langle j,t\rangle \in \langle \mathcal{J}, \mathcal{T} \rangle \quad (5)$$

#### C. System Constraints

This subsection presents the constraints of our optimization problem. Eq. 6 denotes that a computational resource  $(j' \in \mathcal{J}^+)$  should process at most one task in a time interval  $(t' \in$   $\mathcal{T}$ ). Eq. 7 guarantees that a task is processed by at most one resource in a time interval ( $t' \in \mathcal{T}$ ).

$$\sum_{\substack{\langle j,t\rangle \in \langle \mathcal{J},\mathcal{T} \rangle \\ \forall j' \in \mathcal{J}^+, \forall t' \in \mathcal{T}, \forall \varpi \in \{A,S,E\} \\ \sum x^{\varpi} = \langle 1 \rangle$$
(6)

$$\sum_{j'\in\mathcal{J}^{+}} p_{\langle j,t\rangle j't'}^{\varpi} \leq 1,$$
  
$$\forall \langle j,t\rangle \in \langle \mathcal{J},\mathcal{T}\rangle, \forall t'\in\mathcal{T}, \forall \varpi\in\{A,S,E\} \quad (7)$$

Eq. (8-11) maintain the association between the task processing binary decision indicators. Assume that the resource j' is idle in time interval t', which yields  $p^A_{\langle j,t \rangle j't'} = 0$ . Then, it starts to process task  $\langle j,t \rangle$  in the next time interval (t'+1), which leads to  $p^A_{\langle j,t \rangle j'(t'+1)} = 1$ . Hence, the starting time indicator in this time interval  $(p^S_{\langle j,t \rangle j'(t'+1)})$  also becomes one due to Eq. 8. In addition, the ending time indicator in this time interval  $(p^E_{\langle j,t \rangle j'(t'+1)})$  equals zero due to Eq. 9.

$$p^{A}_{\langle j,t\rangle j'(t'+1)} = p^{A}_{\langle j,t\rangle j't'} + p^{S}_{\langle j,t\rangle j'(t'+1)} - p^{E}_{\langle j,t\rangle j'(t'+1)}$$

$$p^{S}_{\langle j,t\rangle j'(t'+1)} + p^{E}_{\langle j,t\rangle j'(t'+1)} \le 1,$$
(8)

$$\forall \langle j, t \rangle \in \langle \mathcal{J}, \mathcal{T} \rangle, \forall j' \in \mathcal{J}^+, \forall t' \in \mathcal{T}$$
(9)

$$p^{A}_{\langle j,t\rangle j'(0)} = p^{S}_{\langle j,t\rangle j'(0)}$$

$$(10)$$

$$\sum_{t'\in\mathcal{T}} p^{\omega}_{\langle j,t\rangle j't'} \leq 1,$$
  
$$\forall \langle j,t\rangle \in \langle \mathcal{J},\mathcal{T}\rangle, \forall j'\in\mathcal{J}^+, \forall \varpi \in \{S,E\} \quad (11)$$

In addition, for a special condition in which a computational resource starts to process a task at the beginning of the simulation (t' = 0), we need Eq. 10 to indicate the starting time  $(p_{\langle j,t \rangle j'(0)}^S = 1)$ . Now assume that in time interval t', the j' processes the task  $\langle j,t \rangle$ , which yields  $p_{\langle j,t \rangle j't'}^A = 1$ . Then, it stops processing that task in the next time interval (t' + 1), which leads to  $p_{\langle j,t \rangle j'(t'+1)}^A = 0$ . Therefore, the ending time indicator turns to one  $(p_{\langle j,t \rangle j't'}^E = 1)$  due to Eq. 8. Lastly, Eq. 11 ensures that a task has only one starting and ending time.

$$\sum_{t'=t}^{T} \sum_{j' \in \mathcal{J}^{+}} p^{A}_{\langle j,t \rangle j't'} * x_{\langle j,t \rangle j'} = \alpha^{B}_{\langle j,t \rangle} * \alpha^{P}_{\langle j,t \rangle},$$

$$\forall \langle j,t \rangle \in \langle \mathcal{J}, \mathcal{T} \rangle \quad (12)$$

$$\sum_{t=1}^{t-1} \sum_{j' \in \mathcal{J}^{+}} p^{A}_{\langle j,t \rangle j't'} * x_{\langle j,t \rangle j'} = 0, \quad \forall \langle j,t \rangle \in \langle \mathcal{J}, \mathcal{T} \rangle \quad (13)$$

$$\sum_{j' \in \mathcal{J}^+} x_{\langle j,t \rangle j'} \leq 1, \qquad \qquad \forall \langle j,t \rangle \in \langle \mathcal{J}, \mathcal{T} \rangle$$
(14)

Each received task  $(\alpha^B_{\langle j,t\rangle} = 1)$  must be processed by a resource for a time period equal to its processing time  $(\alpha^P_{\langle j,t\rangle})$ . Eq. 12 maintains that if a task is offloaded to j', the offloading decision equals one  $(x_{\langle j,t\rangle j'} = 1)$ , and that equation simplifies as  $\sum_{t'=t}^{T} p^A_{\langle j,t\rangle j't'} = \alpha^P_{\langle j,t\rangle}$ . Thus, we have to allocate  $\alpha^P_{\langle j,t\rangle}$  number of time intervals at j'. Eq. 13 ensures that a computational resource starts to process a task after it has received the task, meaning that task processing time intervals  $(t' \in T)$  should be higher or equal to task received time intervals  $(t \in T)$ . Lastly, Eq. 14 bounds the number of offloading decisions for each task to prevent a multiple task offloading case.

#### D. Problem Formulation

We have two problem definitions, P1 and P2, for the smart farm scenario. The main difference between these two definitions originates from their perspective on deadline violation. While minimizing the number of deadline violations is included in the objective function of P1, the approach on P2 is to limit the number of deadline violations to a threshold (Eq. 17).

(P1) Maximize:  

$$(x, v, p^S, p^A)$$

$$W * \min_{j' \in J} \Upsilon_{j'}^R - \frac{1 - W}{2 * \Theta^M} \delta$$

$$- \frac{1 - W}{2 * \Theta^D} \sum_{\langle j, t \rangle \in \langle \mathcal{J}, \mathcal{T} \rangle} v_{\langle j, t \rangle}$$
(15)
Subject to: Eqs. (6 - 14)

Eq. 15 depicts the objective function of the first problem, in which  $W, \Theta^M$ , and  $\Theta^D$  are the weight of the hover time, mean delay, and deadline violation scaling factors, respectively.  $\Upsilon_{j'}^R$  is the remaining battery energy of ABS j', which is calculated by Eq. 1. Task processing decisions  $(p_{\langle j,t\rangle}^A)$  play a significant role on this calculation.  $\delta$  is the mean delay of all received tasks, and  $v_{\langle j,t\rangle}$  is the indicator of deadline violation is determined by Eqs. (4-5). The starting time decision variables  $(p_{\langle j,t\rangle}^A)$  affects these equations. Lastly, these two decision variables  $(p_{\langle j,t\rangle}^A, p_{\langle j,t\rangle}^S)$  are limited by the corresponding task's offloading decision  $(x_{\langle j,t\rangle})$ , as shown in Eqs. (12-13).

As we mentioned earlier, the deadline violation KPI is not included in the objective function of P2 (Eq. 16). Instead, it is considered as a constraint in Eq. 17, in which  $\mathbb{V}$  is the upper bound of the allowed number of deadline violations.

(P2) Maximize:  

$$(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{p^{S}}, \boldsymbol{p^{A}})$$
  $W * \min_{j' \in J} \Upsilon_{j'}^{R} - \frac{1 - W}{\Theta^{M}} \delta$  (16)

 $\begin{array}{ll} \mathbf{,v,p^{-}, p^{-}, p^{$ 

Task offloading problems are generalized assignment problems that are NP-Hard [20]. Therefore, we solve these two problems with a MILP solver and we propose machine learning-based solutions. The latter is explained explicitly in the next section.

#### IV. PROPOSED MACHINE LEARNING-BASED METHODS

This section details the reinforcement learning algorithms we use to solve the explained smart farm problems. These algorithms run independently in each ABS in a distributed manner which improves the scalability of the proposed solutions. In the following subsection, we start to explain a solution for P1.

# A. Q-Learning Approach

P1 (Eq. 15) has three KPIs in the objective function, and a finite-horizon MDP can address them with a tuple  $\mathcal{F} = \{\mathbb{P}, \mathbb{A}, \mathbb{R}, \mathbb{S}, \Pi\}$ . The details of that tuple are:

State Transitions: P : S<sub>1</sub>xAxS<sub>2</sub> ⇒ R
 A transition is triggered whenever an ABS receives a new

task:

- 1) First, the agent analyzes its current state  $\mathbb{S}_1$ .
- 2) Second, it makes a decision for the action  $\mathbb{A}$ .
- Finally, it observes the environmental changes due to its action and determines the new state S<sub>2</sub> and the immediate reward ℝ.

Because of the uncertainty of task arrivals and the stochastic properties of the wireless channel model, the state transitions are not deterministic.

• Action:  $\mathbb{A} = \{j_a | j_a \in \mathcal{J}^+\}$ 

Delegating the received task to a computational resource  $(j_a)$  is the action that an ABS agent should do in this MDP framework. The resource may be at the same ABS, another ABS, or a MEC. Therefore the action domain includes all of these options  $(\mathcal{J}^+)$ .

- State: S = {k, {Δ<sub>j'</sub>|j' ∈ J<sup>+</sup>}, {Υ<sub>j</sub><sup>L</sup>|j ∈ J}} Task type (k) contains two pieces of information (task processing time and deadline) for determining the impact of an action on the system performance. Second, the expected delay (Δ<sub>j'</sub>) is crucial for calculating the system mean delay and deadline violations. Finally, ABS battery levels (Υ<sub>j'</sub><sup>L</sup>) should be included in the state space in order to improve the hover time KPI.
- **Policy**( $\Pi$ ): We choose the epsilon-greedy policy for the tradeoff between exploration and exploitation in the learning process. Also, we use the value iteration method to compute the optimal policy illustrated by Fig. 2a. In this figure,  $\alpha$ ,  $\gamma$ , and  $\mathbb{R}$  are the learning rate, discount, and immediate reward, respectively. The latter is detailed in the following paragraph.
- Reward:

(17)

$$\mathbb{R} = W * (\Upsilon_{j_{a}}^{L} - 1) - \frac{1 - W}{2 * \Theta^{M}} * \Delta_{j_{a}} + \frac{1 - W}{2 * \Theta^{D}} * \left[ (1 - \mathbb{E}(v_{j_{a}})) + \mathcal{V}_{j_{a}}^{L} * \mathbb{E}(v_{j_{a}}) \right]$$
(18)  
$$\Upsilon_{j_{a}}^{L} = \begin{cases} 2, & \text{if } \mathbb{E}(\Upsilon_{j_{a}}^{R}) - \max_{j' \in \mathcal{J}}(\mathbb{E}(\Upsilon_{j'}^{R})) \geq -\epsilon \\ 0, & \text{if } \mathbb{E}(\Upsilon_{j_{a}}^{R}) - \max_{j' \in \mathcal{J}}(\mathbb{E}(\Upsilon_{j'}^{R})) \leq -2 * \epsilon \\ 1, & \text{otherwise} \end{cases}$$
(19)

$$\mathcal{V}_{j_a}^L = \begin{cases} \mathcal{P}^4, & \text{if } \mathbb{E}(v_{j_m}) = 0\\ \mathcal{P}^3, & \text{if } \mathbb{E}(v_{j_r}) = 0\\ \mathcal{P}^2, & \text{if } \exists j' \in (\mathcal{J}/(j_r \cup j_a))(\mathbb{E}(v_{j'})) = 0\\ \mathcal{P}^1, & \text{otherwise} \end{cases}$$
(20)

The main goal of the MDP model should be compatible with the objective function (Eq. 15) in order to obtain the maximization problem in P1. However, an MDP model may only attain that goal as a summation of the immediate rewards,  $\mathbb{R}$ , which can be found by following



Fig. 2: Value iteration processes in Q-Learning and Risk-sensitive RL methods for an example scenario with one MEC server and 4 ABSs (U1 to U4).

the policy  $\Pi$  in each state-action pair (SxA). Therefore, we transform the objective function into Eq. 18 which evaluates the recent action ( $\mathbb{A} = j_a$ ) regarding the KPIs in the objective function.

P1 has three objectives, therefore the immediate reward should also consist of these three objectives. The first objective, increasing hovering time, is described as the total operating time before any of the ABSs have to recharge their battery, and in the problem definition section, we formulate that KPI as a MaxMin problem. We address that KPI with the battery reward levels (Eq. 19) in the MDP model. That value has three levels [0,1,2]according to the expected remaining energy difference between the selected ABS  $\mathbb{E}(\Upsilon_{j_a}^R)$  and the ABS, which has the maximum battery energy  $\max_{j' \in \mathcal{J}} (\mathbb{E}(\Upsilon_{j'}^R))$ . If that energy difference is lower than a certain level  $(\epsilon)$ , the reward function returns positive feedback. Therefore we can balance the battery energies between the ABSs. Lastly, hysteresis  $[-\epsilon, -2\epsilon]$  is introduced to prevent a pingpong effect.

The second objective is minimizing mean delay, which is calculated for each action separately in the immediate reward calculation  $(\Delta_{j_a})$ . Therefore, we will get the total delay for all actions due to the cumulative reward calculation at the end of a simulation.

The third objective depends on the expected deadline violation  $\mathbb{E}(v_{j_a})$ . It is equal to zero if the action  $j_a$  is expected not to cause a deadline violation, and the reward function returns one for this objective. Otherwise, the reward function returns a negative value which is defined in Eq. 20. That value has four severity levels. The highest severity level,  $\mathcal{P}^4$ , occurs when we could

have avoided the violation if we had chosen a MEC as a computational resource ( $\mathbb{E}(v_{j_m}) = 1$ ). This case has the highest severity level because MEC devices have more powerful resources, and they don't have to consume valuable battery energy in order to process a task. The second-highest severity level,  $\mathcal{P}^3$ , occurs when a local resource could have processed that task without causing a violation. This case should have a higher penalty because offloading a task caused extra overhead and delay without benefiting from this action. The third level denotes the case if we have another ABS that could have processed that task without causing a violation. The final and the lowest severity level is a case where we expect a deadline violation as a result of any action.

#### B. Risk-Sensitive Learning Approach

P2 has a deadline violation constraint (Eq. 17) which can be achieved by only choosing an appropriate offloading action in each state-action pair. Therefore, that constraint is isolated from the other KPIs in order to limit the number of deadline violations to be lower than a threshold ( $\mathbb{V}$ ). This separation can be addressed by a finite-horizon constrained MDP (CMDP) [21]. The CMDP is defined by tuple  $\mathcal{F} = \{\mathbb{P}, \mathbb{A}, \mathbb{R}, \mathbb{C}, \mathbb{S}, \Pi\}$ . The details of that tuple are:

- State Transitions: P : S<sub>1</sub>xAxS<sub>2</sub> ⇒ R
   State transitions are the same as in the Q-Learning approach that is explained in Section IV-A.
- State: S = {k, {Δ<sub>j'</sub>|j' ∈ J<sup>+</sup>}, {Υ<sub>j</sub><sup>L</sup>|j ∈ J}, j<sub>a</sub>} In addition to the states in the Q-Learning approach, the risk-sensitive method includes the recent offloading decision into the state space. Thus, we can split the states into two groups regarding risk expectation. Briefly, the

"risk states set" is a subset of the "states set", and an agent falls into such a state when a deadline violation is expected ( $\Phi = \{s \mid s \in \mathbb{S}, and \mathbb{E}(v_{j_a})\}$ ). Here, the expected deadline violation ( $\mathbb{E}(v_{j_a}) = I(\alpha_k^D \leq \Delta_{j_a})$ ) is calculated by Eqs.(4-5).

- Action: A = {j<sub>a</sub>|j<sub>a</sub> ∈ J<sup>+</sup>} The set of available resources (j<sub>a</sub>) is the actions for the risk-sensitive approach.
- Reward:

$$\mathbb{R} = -\left[W * (\Upsilon_{j_a}^L - 1) - \frac{1 - W}{\Theta^M} * \Delta_{j_a}\right]$$
(21)

The main goal of the MDP model should be compatible with the objective function (Eq. 16) to obtain the maximization problem in P2. Therefore, we transform this objective function into Eq. 21 which evaluates the recent action ( $\mathbb{A} = j_a$ ) in terms of the KPIs in the objective function. The details of this equation are explained in Section IV-A. Another point we want to emphasize is that, unlike the Q-Learning approach, we reverse the reward function sign because the policy's exploitation phase chooses an action with the minimum q-value ( $\arg \min_{j_a \in \mathcal{J}^+} \mathbb{S}_1$ ).

Cost for Risk:

$$\mathbb{C} = \begin{cases} -\mathcal{V}_{j_a}^L, & \text{if } \mathbb{S}_1 \in \Phi\\ -1, & \text{otherwise} \end{cases}$$
(22)

In order to isolate and then prevent the risk in P2, we propose Eq. 22 to calculate the risk of each state-action pair. Calculation of the severity level  $(\mathcal{V}_{j_a}^L)$  is detailed in Section IV-A.

Policy(Π): Since we have a separate cost function to evaluate the risk for each state-action pair, we have a q-table solely dedicated to the risk estimation. Fig. 2b shows the value iteration to calculate the risk in a smart farm. Meanwhile, we have another q-table to evaluate the other KPIs, shown in Fig. 2c. While these two isolated q-tables are used for the cumulative reward and risk calculations, we have to choose the same state-action pair in each step for these tables. Therefore we have a third q-table that picks the action for the corresponding state by using the classic epsilon-greedy approach (Fig. 2d). A value iteration process does not generate that q-table, instead, it is an aggregation of the previous two tables with a *ζ* weighting factor.

Choosing a decent  $\zeta$  is essential to improve the convergence of the proposed risk-sensitive solution. Deciding on a higher  $\zeta$  prioritizes the risk side of the problem, while choosing a lower one may improve the other KPIs. We decide on a proper  $\zeta$  with a dynamic approach shown in Algorithm 1. Given parameters  $\zeta^{I}, \mathbb{V}, \lambda, N^{UP}, N^{EP}, \mathbb{G}$  are the initial weighting value, deadline violation threshold, updating step size, updating frequency, the number of episodes, and the gap value, respectively.

This algorithm starts with the initialization of episode and  $\zeta$  values. Then, for each episode, we calculate the three q-tables as they are illustrated in Figs. (2b, 2c, 2d). Line 9 checks the updating frequency, and then Line 10 compares the number

**Algorithm 1** Dynamic update of  $\zeta$  weighting value between the risk and reward

1:	Given: $\zeta^{I}$ , $\mathbb{V}$ , $\lambda$ , $N^{UP}$ , $N^{EP}$ , $\mathbb{G}$
2:	Output: $Q^D, Q^R, Q^C$
3:	$episode = 0, \zeta = \zeta^I$
4:	while $episode < N^{EP}$ do
5:	$Q^D$ :: value iteration for risk (Fig. 2b)
6:	$Q^R$ :: value iteration for reward (Fig. 2c)
7:	$Q^C = \zeta * Q^D + (1 - \zeta) * Q^R \qquad \text{(Fig. 2d)}$
8:	episode = episode + 1
9:	if $episode \equiv 0 \pmod{N^{UP}}$ then
10:	if $\sum v_{\langle j,t \rangle} + \mathbb{G} \leq \mathbb{V}$ then
	$\langle j,t \rangle \in \langle \mathcal{J},\mathcal{T} \rangle$
11:	$\zeta = \min(\zeta - \lambda, 0)$
12:	else
13:	$\zeta = \max(\zeta + \lambda, 1)$
14:	end if
15:	end if
16:	end while

of deadline violations with the threshold. In that equation, we add an additional gap ( $\mathbb{G}$ ) due to the uncertainty property of the tasks. Therefore, the algorithm guarantees Eq. 17 in some cases where the IoT devices demand to process an extremely high number of tasks. Following this comparison, Line 11 decreases the weight of the risk in the case of where Eq. 17 is guarenteed. Otherwise, Line 13 increases the weight of the risk to attain that constraint.

The proposed risk-sensitive method is an episodic modelfree reinforcement learning algorithm. Jin et al. [22] prove that the worst-case complexity of this algorithm is  $\tilde{\mathcal{O}}(H^2\sqrt{|\mathbb{A}||\mathbb{S}|N^{EP}})$ , where H is the number of steps in an episode,  $|\mathbb{A}|$ , and  $|\mathbb{S}|$  are the action and state space sizes, and  $N^{EP}$  is the number of episodes. We train our risksensitive method offline with a collected dataset from the simulation platform. Each of these datasets has a number of tasks defining the number of steps (H) in an episode. Thus, our algorithm's complexity quadratically increases with the number of tasks in each episode. According to Section IV.B., with the increasing number of ABSs (J), the state space  $(|\mathbb{S}|)$  changes quadratically, and the action space  $(|\mathbb{A}|)$  changes linearly. Therefore, computation time grows as  $J^{\sqrt{3}}$  with the number of ABSs in the problem. Meanwhile, the number of episodes  $(N^{EP})$  is determined according to the algorithm's convergence, which is not only related to the size of the given data. The convergence of the algorithm (training phase) is also affected by the content of the data. Therefore, we empirically determine the value of  $N^{EP}$ .

# V. NUMERICAL RESULTS

# A. Simulation Platform

We chose Omnet++ [23] as a discrete event simulator to simulate the IoT traffic. In addition, we used Simu5G libraries [24], which is developed over Omnet++, to simulate a 5G channel model between IoT devices, ABSs and MEC platform. These libraries implement the TR 36.873 specification [25] to calculate the channel fading in new radio numerology ( $\mu = 0$ )

TABLE II: Simulation parameters.

Task Type	$(1/\lambda)$	$\alpha^D_{\langle j,t\rangle}$	$egin{aligned} & lpha^P_{\langle j,t angle} \ & ( ext{ABS}) \end{aligned}$	$\begin{array}{c} \alpha^P_{\langle j,t\rangle} \\ (\text{MEC}) \end{array}$
Fire detection	0.25s	1s	0.1s	0.05s
Pesticide detection	0.25s	2s	0.2s	0.1s
Growth monitoring	0.5s	15s	1.5s	0.75s

TABLE III: Energy consumption parameters.

ABS No $(j')$	$\Upsilon^B_{j'}$	$\Upsilon^H_{j'}$	$\Upsilon^A_{j'}$	$\Upsilon^{I}_{j'}$	$\Upsilon^C_{j'}$
0	570	211	17	4320	12960
1	570	211	17	4320	12960
2	627	211	17	4320	12960
3	627	211	17	4320	12960

of a 5G physical layer. Shadow fading standard deviation and carrier frequency are chosen at 4dB and 2GHz for lineof-sight communication, respectively. Table II presents the parameters used in the simulations. In particular, we looked at a smart farm scenario with four ABSs (*J*=4) and one MEC (*L*=1) devices. We focused on three types of image processing operations ( $\mathcal{K}$ ) that have different arrival rates, deadlines and processing times. Interarrival times between the tasks are exponentially distributed with a mean of  $(1/\lambda)$ . Deadlines and processing times are constants. The results are generated as the average of ten runs with different seeds and we set the simulation time to be 50 seconds (T = 50s).

#### B. Baselines

1) Round Robin: Every computational resource located at the ABSs and the MEC is utilized in a round robin fashion when assigning the tasks.

2) Lowest Queue Time and Highest Energy First: This algorithm considers the destinations' remaining energy level and queue time. As stated in [6], queue time is the total time it takes for the resource to compute all of the tasks in its queue. Every ABS stores the remaining energy level and queing time of every possible offloading destination. The ABSs regularly update their neighbours with such information. The decision making algorithm begins by finding the lowest queue time amongst the neighbouring resources. If an ABS's queue time is lower than the current ABS's queue time by at least 0.5s, we will use the this value as the lowest queue time. Else, the algorithm will use the current ABS's queue time as the lowest queue time. Next, the algorithm will find the neighbour that has the highest remaining energy level and the queue time is equal or less than the lowest queue time. If such a node exists and its energy level is higher than the current ABS's energy level by at least 1%, then the current ABS will offload the task to that node. If not, then the task will be computed locally.

3) Energy-Centric Approach: This method prioritizes the Max-min fairness over the deadline violation minimization. Therefore, it provides a higher hovering time for the smart farm. Implementation is based on the proposed risk-sensitive method. Meanwhile, the size of the difference between the highest energy battery and the lowest energy battery is defined as risk. Thus Algorithm 1 keeps the  $\zeta$  value at higher levels while this gap is larger than a certain threshold.

TABLE IV: Machine Learning Parameters.

Parameter	Notation	Value
Learning Rate	$\alpha$	0.05
Discount	$\gamma$	0.85
Number of Episodes	$N^{EP}$	100k
Updating frequency	$N^{UP}$	1k
Initial weighting value	$\zeta^{I}$	0.00
Step Size	$\lambda$	0.02
Max. Deadline Violation	$\mathbb{V}$	3
Gap value	G	2
Weighting value for Q-Learning	W	0.5
Mean delay & deadline violation scalers	$\Theta^M, \Theta^D$	1, 1
Deadline violation severity levels	$\mathcal{P}^{4,3,2,1}$	$-\{4, 3, 2, 1\}$

#### C. Energy Consumption Parameters

We used the parameters found in Table III and Eq. 1 in order to model an ABS's remaining battery energy. By considering limited simulation time, idle and busy CPU energy consumptions are defined as ABSs that would be run for ten hours to expose the performance differences between the techniques.

## D. Convergence of Machine Learning Approaches

We use an offline learning approach for machine learning (ML) solutions. First, we generate one hundred different data sets with Omnet++ for the learning phase. Then, we run Q-Learning, Energy-centric and Risk-sensitive methods for 100k episodes, in which each episode uses another data set to prevent an overfitting problem. Then, we transfer the generated tables in this learning phase to the test phase. Finally, in the test phase, we use Omnet++ to simulate a real 5G wireless channel model and evaluate the generated tables according to several KPIs. That KPI evaluation is detailed in the following subsection. Table IV represents the numerical values used in ML algorithms.

Fig. 3a illustrates the convergence of the Q-learning baseline method. As seen, this method practices the arg max approach to find the state-action value. If we turn to the risk-sensitive method, Fig. 3b and Fig. 3c show the convergence of reward and risk tables in the learning phase, respectively. Owing to multi-actor-based approach and use of different datasets in each episode, the variations in the cumulative reward and risk represent the extreme levels in both tables. On the other hand, it is seen that each actor (in other terms, each ABS) learns at the same speed and approaches the zero level.

Fig. 4a shows the trend of convergences in these two tables. As mentioned in Algorithm 1, we prioritize the risk table at the beginning of the learning phase. Fig. 4a confirms that algorithm; while the cumulative reward reduces in the risk table 57% (from 140 to 60), it is only 20% (from 14 to 9) in the reward table at the same episode. We should note that the tradeoff between different KPIs may adaptively change with Algorithm 1 for different smart farm scenarios. This can be seen easily in Fig. 4b. This figure shows the learning trend of the Energy-centric approach, which prioritizes the MaxMin fairness of energy usage in the batteries of ABSs.



(a) Q-learning convergence. (b

(b) Reward convergence in Risk-sensitive.(c) Risk convergence in Risk-sensitive.Fig. 3: Convergences of Q-learning and Risk-sensitive method.



Fig. 4: Dynamically change of convergences between reward and risk q-tables.

# E. KPI Evaluations

Fig. 5 shows the remaining energy levels in the batteries of ABSs for different solution methods. As it is explained in the problem definition, we aim to increase the hovering time of ABSs. Then, we propose a MaxMin solution in which we maximize the energy of the ABS, which has the minimum battery. That value is 78% for our proposed risk-sensitive method, which is higher than the heuristics but lower than the Q-Learning approach. The main reason for that output is prioritizing deadline violations in risk-sensitive solutions. Also, the energy-centric baseline performs better than the risksensitive approach and provides an extensive fairness between the ABSs. Meanwhile, the risk definition of that approach might be revised for outperforming the classical Q-learning method.

Fig. 6 illustrates the mean delay KPI. It is clear that ML approaches outperform the heuristics. Meanwhile, there is no decisive optimal ML solution in terms of mean delay minimization in different computational resources. Q-Learning has better performance in the ABSs, but it has a dramatically higher delay in MEC. This is because the Q-Learning policy sends more tasks to MEC. On the other hand, the risk-sensitive approach provides lower overall delay than all other methods.

The main performance improvement of the risk-sensitive approach can be seen in Fig. 7. This figure demonstrates the deadline violations in different ABSs and MEC. Although the Q-Learning and Energy-centric methods outperform the heuristics, they can not guarantee that the number of deadline violations will be lower than a threshold. In contrast, this figure demonstrates that our risk-sensitive solution establishes risk-free offline learning for a multi-objective optimization problem.

#### F. Upper Bound Comparison

We chose Gurobi Solver to find the upper bounds for the maximization problems [26]. Owing to the fact that these problems are NP-hard, a MILP solver may provide a solution for small space problems [27]. Therefore, we had to limit the simulation time to four seconds (T = 4s), and we ran the solver for 48 hours for each case to find the optimum solution. Furthermore, in order to increase the number of tasks in this shorter time duration, we chose interarrival time as 0.125s for all task types. Finally, we reduce the deadlines for fire detection (0.2s) and pesticide detection tasks (0.6s) to make the problem harder than the standard configuration.



Fig. 5: Remaining energy levels in different methods.



Fig. 6: Mean delay in different methods.



Fig. 7: Number of deadline violations.

TABLE V: Upper Bound Comparison (P1: Eq. 15, P2: Eq. 16, M: MILP Solver, R: Risk-Sensitive ML, Q: Q-Learning ML)

Problem	Method	W	$\min_{j'\in J}\Upsilon^R_{j'}$	δ	$\sum_{\substack{j \in \mathcal{J} \\ t \in \mathcal{T}}} v_{jt}$
P1	М	1	98.2%	1.89s	$\frac{\iota \in I}{72}$
P1	0	1	98.2%	4.01s	89
P1	Ň	0.5	97.0%	0.92s	14
P1	Q	0.5	97.7%	0.68s	42
P1	Ň	0	96.3%	0.84s	9
P1	Q	0	97.0%	0.44s	26
P2	М	1	97.3%	0.88s	15
P2	М	0.5	96.7%	0.89s	15
P2	М	0	96.3%	0.85s	15
P2	R	Ν	96.5%	0.45s	13

The top part of Table V shows the upper bounds of P1 (Eq.15) with different weight factors (W). The first row (W = 1) represents the maximum hovering time for P1. Our proposed ML solution for P1 can reach that level (second row) if we only add the hovering time maximization in the reward function. On the other hand, in a balanced solution (W = 0.5), it is seen that the ML solution tends to favor hovering time. Although we can get higher hovering time by ML, the MILP solution outperforms this method in terms of mean delay and deadline violation minimization. Lastly, if we remove the hovering time KPI from the multi-objective function (W = 0), the MILP solver reduces the mean delay and the number of violations significantly.

The bottom section of Table V shows the upper bounds of P2 (Eq.16). First, we find a feasible solution with a significantly higher deadline violation threshold ( $\mathbb{V} = 15$ ) than the solution of MILP P1<sup>2</sup>. It is undoubtedly seen that the risk-sensitive ML surpasses the MILP solution regarding minimizing the number of deadline violations. Moreover, the ML solutions are far better for reducing the mean delay. In addition, its hovering time maximization performance is almost the same as the MILP solution.

### G. Processing Time and Deadline Alterations

It has been claimed that training an ML solution with offline data provides an efficient and reliable solution for a wireless network problem [28], [29]. Therefore, we trained our ML solutions with preliminary data and then solve those finding solutions within a simulation platform. In this subsection, we provide a kind of stress test for our ML solutions. We generate a one-hundred offline dataset that has different processing time and deadline for pesticide detection (PD) task (Fig 8). The results shows that the risk-sensitive solution adapts better than Q-Learning in most KPIs for the changes of these parameters.

#### VI. CONCLUSION

This study focused on a smart agriculture scenario that uses ABSs to augment the computational resources of IoT

<sup>&</sup>lt;sup>2</sup>P2 has an additional constraint for limiting the number of deadline violations. Adding an extra constraint usually increases the solution space of a MILP solution. Therefore it may need more time to find a feasible solution. It has to be reminded that we limit the MILP solutions by 48 hours.



Fig. 8: Impact of pesticide detection task parameter variations. (a), (b), (c): Increasing processing time. (d): Increasing deadline.

devices that need to process deadline-critical image processing tasks. We defined a problem definition that made sure that the tasks are completed before their deadlines and improved the hovering time of these ABSs. We addressed this using a CMDP, and then proposed a risk-sensitive approach for this CMDP that constrained the number of deadline violations. The results show that a risk-sensitive approach is feasible for guaranteeing a constraint-based KPI while providing an energy-efficient solution for the ABSs. As future work, we are planning to provide a solution for more extensive state-space problems.

#### ACKNOWLEDGEMENT

This work is supported by MITACS Canada Accelerate program under collaboration with Nokia Bell Labs.

#### REFERENCES

- [1] M. Jhuria, A. Kumar, and R. Borse, "Image processing for smart farming: Detection of disease and fruit grading," in 2013 IEEE 2nd International Conference on Image Information Processing, IEEE ICIIP 2013, pp. 521-526, 2013.
- [2] N. R. Dhumale and P. C. Bhaskar, "Smart agricultural robot for spraying pesticide with image processing based disease classification technique,' in 2021 International Conference on Emerging Smart Computing and Informatics, ESCI 2021, pp. 604-609, Institute of Electrical and Electronics Engineers Inc., mar 2021.
- [3] M. Akbari, M. R. Abedi, R. Joda, M. Pourghasemian, N. Mokari, and M. Erol-Kantarci, "Age of Information Aware VNF Scheduling in Industrial IoT Using Deep Reinforcement Learning," IEEE Journal on Selected Areas in Communications, vol. 39, pp. 2487-2500, aug 2021.
- [4] F. Khoramnejad, R. Joda, and M. Erol-Kantarci, "Distributed Multi-Agent Learning for Service Function Chain Partial Offloading at the Edge," in 2021 IEEE International Conference on Communications Workshops (ICC Workshops), pp. 1-6, IEEE, jun 2021.
- [5] 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, pp. 1-6, IEEE, jun 2021.
- A. C. Nguyen, T. Pamuklu, A. Syed, W. S. Kennedy, and M. Erol-[6] Kantarci, "Reinforcement Learning-Based Deadline and Battery-Aware Offloading in Smart Farm IoT-UAV Networks," in ICC - International Conference on Communications, IEEE, 2022.

- [7] H. Liao, Z. Zhou, Z. Wang, S. Mumtaz, and M. Guizani, "Learning-Based Queue-Aware Task Offloading and Resource Allocation for Air-Ground Integrated PIoT," in ICC 2021 - IEEE International Conference on Communications, pp. 1-6, IEEE, jun 2021.
- [8] K. Xiong, Y. Liu, L. Zhang, B. Gao, J. Cao, P. Fan, and K. B. Letaief, "Joint Optimization of Trajectory, Task Offloading and CPU Control in UAV-assisted Wireless Powered Fog Computing Networks," IEEE Transactions on Green Communications and Networking, vol. 2400. no. c, pp. 1-13, 2022.
- [9] J. Zhao, Y. Wang, Z. Fei, and X. Wang, "Uav deployment design for maximizing effective data with delay constraint in a smart farm," in 2020 IEEE/CIC International Conference on Communications in China (ICCC), pp. 424-429, IEEE, 2020.
- [10] S. Khairy, P. Balaprakash, L. X. Cai, and Y. Cheng, "Constrained Deep Reinforcement Learning for Energy Sustainable Multi-UAV Based Random Access IoT Networks With NOMA," IEEE Journal on Selected Areas in Communications, vol. 39, pp. 1101-1115, apr 2021.
- [11] O. Ghdiri, W. Jaafar, S. Alfattani, J. B. Abderrazak, and H. Yanikomeroglu, "Energy-Efficient Multi-UAV Data Collection for IoT Networks with Time Deadlines," in GLOBECOM 2020 - 2020 IEEE Global Communications Conference, pp. 1-6, IEEE, dec 2020.
- [12] T. Zhang, J. Lei, Y. Liu, C. Feng, and A. Nallanathan, "Trajectory optimization for UAV emergency communication with limited user equipment energy: A Safe-DQN approach," IEEE Transactions on Green Communications and Networking, vol. 5, no. 3, pp. 1236–1247, 2021.
- [13] Z. Yang, C. Pan, K. Wang, and M. Shikh-Bahaei, "Energy Efficient Resource Allocation in UAV-Enabled Mobile Edge Computing Networks," IEEE Transactions on Wireless Communications, vol. 18, pp. 4576-4589, sep 2019.
- [14] Y. Xu, T. Zhang, D. Yang, and L. Xiao, "UAV-Assisted Relaying and MEC Networks: Resource Allocation and 3D Deployment," in 2021 IEEE International Conference on Communications Workshops (ICC Workshops), pp. 1-6, IEEE, jun 2021.
- K. Yao, J. Chen, Y. Zhang, L. Cui, Y. Yang, and Y. Xu, "Joint [15] Computation Offloading and Variable-width Channel Access Optimization in UAV Swarms," in GLOBECOM 2020 - 2020 IEEE Global Communications Conference, pp. 1-6, IEEE, dec 2020.
- [16] T. Wang, Y. Li, and Y. Wu, "Energy-Efficient UAV Assisted Secure Relay Transmission via Cooperative Computation Offloading," IEEE Transactions on Green Communications and Networking, vol. 5, no. 4, pp. 1669–1683, 2021.
- [17] N. B. Khalifa, M. Assaad, and M. Debbah, "Risk-Sensitive Reinforcement Learning for URLLC Traffic in Wireless Networks," in 2019 IEEE Wireless Communications and Networking Conference (WCNC), vol. 2019-April, pp. 1-7, IEEE, apr 2019.
- M. Alsenwi, N. H. Tran, M. Bennis, A. Kumar Bairagi, and C. S. Hong, [18] "eMBB-URLLC Resource Slicing: A Risk-Sensitive Approach," IEEE Communications Letters, vol. 23, pp. 740-743, apr 2019.
- [19] C. Zhou, W. Wu, H. He, P. Yang, F. Lyu, N. Cheng, and X. Shen, "Deep Reinforcement Learning for Delay-Oriented IoT Task Scheduling in SAGIN," IEEE Transactions on Wireless Communications, vol. 20, pp. 911-925, feb 2021.
- [20] L. Zhang, J. Luo, L. Gao, and F. C. Zheng, "Learning-Based Computation Offloading for Edge Networks with Heterogeneous Resources," IEEE International Conference on Communications, vol. 2020-June, 2020.
- [21] E. Altman, Constrained Markov Decision Processes. CRC Press, 2004.
- [22] C. Jin, Z. Allen-Zhu, S. Bubeck, and M. I. Jordan, "Is Q-learning provably efficient?," Advances in Neural Information Processing Systems, vol. 2018-December, no. NeurIPS, pp. 4863–4873, 2018. [23] OpenSim Ltd., "What is OMNeT++?," 2019.
- [24] 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.
- [25] 3GPP, "Study on 3D channel model for LTE (TR 36.873)," Tech. Rep. Release 12.8.0, 3GPP, 2018.
- [26] L. Gurobi Optimization, "Gurobi Optimizer Reference Manual," 2021.
- [27] T. Pamuklu and C. Ersoy, "GROVE: A Cost-Efficient Green Radio Over Ethernet Architecture for Next Generation Radio Access Networks," IEEE Transactions on Green Communications and Networking, vol. 5, pp. 84-93, mar 2021.
- [28] WG2, "AI/ML Workflow Description and Requirements v1.03," tech. rep., O-RAN, 2021.
- [29] L. Bonati, S. D'Oro, M. Polese, S. Basagni, and T. Melodia, "Intelligence and Learning in O-RAN for Data-Driven NextG Cellular Networks," IEEE Communications Magazine, vol. 59, pp. 21-27, oct 2021.