Energy-aware task scheduling and offloading using deep reinforcement learning in SDN-enabled IoT network

Bassem Sellami, Akram Hakiri, Sadok Ben Yahia, Pascal Berthou

Energy-Aware Task Scheduling and Offloading using Deep Reinforcement Learning in SDN-enabled IoT Network

Bassem Sellami[a], Akram Hakiri[b], Sadok Ben Yahia[c], Pascal Berthou[d]

[a]*University of Tunis El Manar, Faculty of Sciences, Dept of Computer Sciences, Campus universitaire, BP 37, Tunis, 1002, Tunisia*
[b]*University of Carthage, SYSCOM ENIT, ISSAT Mateur, Route de Tabarka, Mateur, 7030, Tunisia*
[c]*Tallinn University of Technology, Department of Software Science , Akadeemia tee 15a, Tallinn, 12618, Estonia*
[d]*CNRS, LAAS, UPS, 7 Avenue du colonel Roche, BP 54200, Toulouse, F-31031, France*

Abstract

The fifth-generation (5G) mobile network services have made tremendous growth in the Internet of Things (IoT) network. A counters number of battery-powered IoT devices are deployed to serve diverse scenarios, e.g., smart cities, autonomous farming, smart manufacturing, to name but a few. In this context, energy consumption became one of the most critical concerns in interconnecting smart IoT devices in such scenarios. Additionally, whenever these IoT devices are distributed in space and time-evolving, they are expected to deliver high volume data scalably/predictably while minimizing end-to-end latency. Furthermore, edge IoT nodes often face the biggest hurdle of performing optimal resource distribution and achieving high-performance levels while coping with the variability of task handling, energy conservation, and ultra-reliable low-latency.

This paper investigates an energy-aware and low-latency oriented computing task scheduling problem in a Software-Defined Fog-IoT Network. First, we formulate the online task assignment and scheduling problem as an energy-constrained Deep Q-Learning process as a kickoff. The latter strives to minimize the network latency while ensuring energy efficiency by saving battery power under the constraints of application dependence. Then, given the task arrival process, we introduce a deep reinforcement learning (DRL) approach for dynamic task scheduling and assignment in the Software-Defined Networking (SDN)-enabled edge networks. We conducted comprehensive experiments and compared the introduced algorithm to three pioneering deep learning algorithms (i.e., deterministic, random, and A3C agents). Extensive simulation results demonstrated that our proposed solution outperforms these algorithms. Additionally, we highlight the characterizing feature of our design, energy-awareness, as it offers better energy-saving by up to 87% compared against the other approaches. We have shown that the offloading scheme could perform more task assignments with the available battery power by up to 50% less time delay. Our results back our claims that the solution we propose can readily be used to dynamically optimize task scheduling and assignment of complex jobs with task dependencies in distributed Fog IoT networks.

Keywords: Task scheduling, Deep Reinforcing Learning SDN, Fog Computing, Internet of Things
PACS: 0000, 1111
2000 MSC: 0000, 1111

1. Introduction

The Internet of Things (IoT) is rocketly connecting a considerable number of smart objects [1], which generate, gather, process, infer and transmit the massive amount of sensory data that should be processed at network edge nodes. Typical edge nodes, so-called *fog computing nodes*, often rely on discoverable, generic, forward-deployed servers and IoT gateways located in single-hop proximity of wireless mobile IoT devices [2]. With the acceleration of 5G commercial deployment, individual fog nodes should coordinate their processing with neighboring helper IoT nodes by offloading their tasks significantly to reduce task execution delay. Specifically, such fog nodes should support burst flow and unpredictable IoT traffic at different time scales. In addition, they are compelling for new types of delay-sensitive IoT services and applications, such as updating maps for self-driving cars or delivery drones, energy usage measurements from a smart grid, emergency monitoring, intelligent manufacturing, interactive multiplayer online games, and disaster relief.

However, wireless communication and computing resources (CPU, memory, storage) are usually highly limited and energy-consuming. Thus, it becomes more tricky to meet the increasingly growing demand and dynamic needs of IoT applications and address the heterogeneous requirements of smart objects that communicate over the Internet. Therefore, flexible resource management, intelligent network control, and efficient task scheduling algorithms are pivotal in ensuring fair and guaranteed steady performance.

Furthermore, millions of battery-powered IoT devices such as smart cameras, smartphones, home entertainment systems, smart TVs, environmental monitoring sensors, and smart meters are deployed to serve diverse scenarios such as smart cities, autonomous farming, smart buildings, and smart manufacturing; and these applications require delivering high volumes of data over ultra-reliable, low-latency wireless communication services [3]. The growth of IoT services and applications, along with the increasing traffic generated by IoT devices, raises the concerns for the increase of energy consumption needed to power, distribute and deploy IoT solutions [4].

In this respect, Software-Defined Networking (SDN) is used to enable flexible and collaborative task offloading service orchestration in cloud-Mobile Edge Computing (MEC) [5]. Furthermore, a service orchestration scheme is proposed to reduce network load, along with a differentiated cloud-edge offloading decision algorithm that has been submitted to improve cloud computation and energy consumption. However, the cost of using the cloud resources used in the above approach is often mistreated.

Similarly, an SDN scheme for balancing Edge-Cloud traffic load and improving service response time has also been introduced in [6]. A nature-inspired meta-heuristic scheduler [7], based on ant colony optimization, has been introduced to load balance IoT tasks between fog nodes effectively. Nonetheless, the increasing growth of network traffic generated by massive IoT applications and services makes it hard to forecast energy-saving computing offloading. For example, recent studies revealed that buildings account to nearly 40% of the global energy consumption [8] and more than 35% of CO2 emissions [9] in many countries.

Likewise, Cai et al. [10] proposed a framework for tasks and energy offloading in a fog-enabled IoT network while minimizing task's execution delay. Recently, machine learning techniques have become promising to bring proactive adaptability to an SDN controller by performing data analysis, network optimization, and automated provision of network services [11]. In fact, traditional task offloading schemes often assume that the mobile communication and signal processing between mobile edge devices and their radio base stations (i.e., eNodeB) are well modeled. This is, in contrast, very complicated due to the mobility and highly dynamic aspect of IoT devices, which is hard to predict. Furthermore, these network provisioning approaches neither consider the dynamicity of IoT applications nor care about resource utilization of fog-enabled IoT nodes. To curb both downsides, the network must be flexible enough to be reprogrammed in compliance with any change in IoT application needs.

The fog computing devices should provide : (*i*) An on-demand resource allocation to support adaptive horizontal and vertical scaling of the network resources; (*ii*) A flexible infrastructure virtualization that exploits in-network programmability capabilities to operate inside an SDN-enabled virtualization platform; (*iii*) A device-driven and human-driven intelligence to address the issues of energy efficiency and ultra-low latency requirements for future reliable and real-time IoT applications [12]. To address the issues mentioned earlier, we introduce, in this paper, a DRL energy-efficient task assignment and scheduling in SDN-based fog IoT Network. An SDN-fog computing model allows reducing network latency and traffic overhead by centralizing the network control and orchestration in a single SDN controller layer. We also introduce a DRL algorithm to tackle the task allocation and resource planning issues in dynamic and distributed IoT environments to improve latency minimization and reduce energy consumption. Our DRL relies on intelligent agents that learn to update better decisions directly from the experience of interacting with the environment, while ensuring optimization of consumable energy with minimization of latency.

The remainder of this paper is organized as follows: Section 2 scrutinizes the related work. Then, in Section 3, we depict the details of our architecture, and we describe the analytical model for our DL-based dynamic task scheduling and resource management. Section 4 describes our DRL algorithm to address the task offloading and scheduling issue and describes the implementation details of our solution. Section 5 evaluates our solution to validate our claims of flexible data delivery and low latency communication overhead. Finally, Section 6 presents concluding remarks, alluding to lessons learned and future work.

2. A scrutiny of the related work

This section sketches the research directions that paid close heed to on-task offloading and resource allocation problems using reinforcement learning in a fog-enabled network with SDN. In the following, we start by reviewing the task scheduling within SDN-enabled edge computing.

2.1. Task Scheduling SDN-enabled Edge Computing

SDN has been widely used to empower dynamic and effective resource allocation in diverse cloud [13] and data center networks [14]. Furthermore, it provided on-demand application and resource management in wireless sensor networks [15] and network edge [16]. For example, Wu et al. [17] introduced the UbiFlow framework that combines ubiquitous flow control and mobility management in heterogeneous urban networks. The latter adopts distributed SDN controller's pattern to split traffic scale among geographically distributed IoT network silos, where each controller can maintain network scalability, load balancing, and consistency.

Chen et al. [18] proposed an SDN-based heuristic model for offloading distributed computing resources in an ultra-dense network. Yang et al. [19] introduce an online learning approach where mobile users can offload their computational tasks to neighborhood fog [20] and an extended approach that makes use of the adversarial multi-arm bandit framework to construct an online learning policy. The authors formulate a task offloading problem as a mixed-integer non-linear program to solve the task placement and the allocation of resources problems in mobile edge computing. Similarly, Pen et al. [21] introduced a mobile task offloading framework for device-to-device (D2D) fogging. They leveraged Lyapunov optimization D2D fogging methods to achieve energy-efficient task executions for network-wide users and reduce time-average task execution to avoid over-exploiting and free-riding behaviors. Furthermore, Kuang et al. [22] investigated a joint problem of partial offloading scheduling and resource allocation over multiple independent tasks in MEC networks. They formulated their framework as a non-convex mixed-integer optimization problem based on dual decomposition and Lagrangian relaxation to minimize the weighted sum of the execution delay and energy consumption while guaranteeing the transmission power constraint tasks. Finally, Zhang et al. [23] proposed a fair and energy-minimized task offloading algorithm based on the fairness scheduling metric. Their scheme considered task offloading energy consumption, historical average energy demand, and the FN priority to offer optimal transmission power for wireless fog-enabled mobile IoT nodes.

Chalapathi et al. [24] proposed a Latency Aware Task Assignment (LATA) scheme for a multi cloudlets network to optimize the latency monetary cost in computing the tasks of mobile devices by making optimal task assignments among the micro-clouds. Furthermore, the proposed LATA model achieves admission control policies to maintain optimal traffic conditioning when congestion occurs. Besides, the authors in [25] addressed the problem of task offloading in SDN-enabled networks by offering a computation scheme for multi-hop IoT access points (APs). The proposed scheme is formulated as an integer linear program (ILP) and greedy-heuristic-based approach to offer optimal local or remote task computation, optimal fog node selection, and optimal path selection.

The authors in [26] proposed a framework for task scheduling and resource allocation in cloud-hosted IoT applications. The scheduling is performed on cloud-hosted hosting multiple Virtual Machines (VMs). The authors discussed VMs placement in the cloud, where each

VM hosts several IoT tasks. Compared to this approach, our contribution relies on the edge-hosted nodes to perform resource allocation, where the SDN controller performs all task scheduling and resource allocation. We do not use VMs in the edge IoT network since they increase the load in the IoT infrastructure, in contrast to the logic to the Edge/Fog approach. Instead, we rely on lightweight IoT functions hosted on low-computation and energy-constrained devices.

2.2. Reinforcement Learning Task Allocation

Task scheduling problem in dynamic IoT environment is often one of the most challenging resource management problems. Indeed, it manifests as tricky online decision-making, where proactive solutions usually depend on the dynamic workload and the interaction with the surrounding environment [39]. Table 1 compares between related work that studied different criteria, including energy-efficiency, scalability, latency and bandwidth utilization. Specifically, Lei et al. [27] provided a comprehensive survey of automating and orchestrating IoT resources using reinforcement learning (RL) to achieve autonomy. Wan et al. [28] introduced DRL-based scheduling for cellular networks. They proposed two methods, i.e., learning from a dual AI (Artificial Intelligence) module and learning from the expert solution to perform link adaption, feedback, and scheduling mechanisms used in real LTE network. The former uses two independent agents to train and learn from each other. The latter relied on Proportional Fair (PF) scheduling algorithm, which is used as an expert knowledge to help with DRL agent training. Sen et al. [29] proposed a Machine Learning (ML) approach for scheduling application tasks in distributed Intelligent Cognitive Assistants (ICA). They introduced a heuristic method for solving task assignment problems between the three tiers in the edge computing system (i.e., remote cloud, fog, and edge devices). Hongzi et al. [30] introduced a DeepRM framework to build autonomous and intelligent systems that learn to manage resources from their own experience directly.

Likewise, the authors in [34] proposed a DRL approach for a decentralized mechanism for resource allocation in vehicle-to-vehicle (V2V) communications. They introduced a DRL agent that makes decisions to find optimal sub-band and power levels for transmitting V2V data. Doha et al. [31] proposed a cooperative DRL-based task allocation process that combines learning agents' capabilities to improve resource sharing and distributed task allocation. Finally, Wang et al. [32] proposed a DRL-based incremental approach for learning allocation strategies. They extracted diverse task patterns from the large volume of historical allocation data to improve learning efficiency. The authors in [35] introduced an RL approach for learning the scheduling policy automatically and reducing the estimation error on data centers. Similarly, Ma et al. [33] proposed an IoT-based deadline and cost-aware task scheduling optimization scheme to satisfy the Quality of Service (QoS) requirements in cloud-hosted IoT applications. The proposed algorithm uses heuristic approaches to minimize the execution cost under deadline constraints in the infrastructure as a service (IaaS) model.

Similarly, Liao et al. [36] propose a learning-based framework for channel selection with service reliability awareness, energy awareness, backlog awareness, and conflict awareness, by leveraging the combined power of machine learning, Lyapunov optimization, and matching theory. The authors provide rigorous theoretical analysis and prove that the proposed framework can achieve guaranteed performance with a bounded deviation from the optimal performance with global state information (GSI) based on only local and causal information. Furthermore, the authors extended their work in [38] by introducing a learning-based queue-aware task offloading scheme and an algorithm for resource allocation, so-called QUARTER. Specifically, the proposed algorithm can minimize energy consumption under the long-term constraint of queuing delay. They propose a queue-aware actor-critic-based task offloading algorithm to cope with dimensionality curse. A closer work has been introduced by Sun et al. [37], which developed a novel user-centric energy-aware mobility management (EMM) scheme, in order to optimize the delay due to both radio access and computation, under the long-term energy-consumption constraint of the user. Based on Lyapunov optimization and multi-armed bandit theories, EMM offers an online optimization approach, without future system state information, and effectively handles the imperfect system state information.

2.3. Paper's Contributions

Unlike the approaches mentioned above, which are mainly based on meticulously designed heuristics that ignore the patterns of incoming tasks, our approach uses

Article	Idea	Criteria			
		Energy effeciency	Scalability	Latency	Bandwidth
Lei et al. [27]	Automation and orchestration of IoT resources using RL & DRL in IoT Network	-	-	✓	✓
Wang et al. [28]	DRL-based planning for LTE cellular networks	-	✓	-	✓
Sen et al. [29]	Schedule application tasks in intelligent cognitive assistants distributed through machine learning	✓	✓	-	-
Hongzi et al. [30]	Introduced a DRL-based framework to automatically manage IOT resources from their own experience	-	✓	-	-
Dhoha et al. [31]	Improve resource sharing and task allocation using a cooperative process based on DRL	-	✓	-	-
Wang et al. [32]	Incremental approach based on DRL for task allocation strategies	-	✓	-	-
Ma et al. [33]	Task planning optimization scheme based on heuristic approaches	-	-	✓	-
H .Ye et al. [34]	Develop a decentralized resource allocation mechanism for V2V communications using multi-agent DRL	-	✓	✓	-
S. Liang et al. [35]	Introduce a mechanism to address the customized job scheduling problem in data centers using DRL	-	-	-	-
Liao et al. [36]	Propose a learning-based context-aware channel selection framework	✓	-	-	-
Sun et al. [37]	Develop a novel user-centric energy-aware mobility management (EMM) scheme to optimize the delay	✓	-	✓	-
Liao et al. [38]	Propose a learning-based queue-aware task offloading and resource allocation algorithm	✓	-	✓	-
Our approach		✓	✓	✓	✓

Table 1: Reinforcing learning for Task Allocation

SDN to enhance the control and management of fog-enabled IoT networks in terms of flexibility and intelligence. Our approach provides an intelligent IoT network communication system to create a single, coherent and unifying control framework for real-time fog-enabled IoT network design to resolve the task scheduling problem.

Furthermore, recent studies [40] indicate that over 35 billions IoT devices are installed worldwide in 2021, and we expect 50 billion by 2030. The IoT industry will result in an additional 53 TWh of fuel to power, distribute and deploy IoT solutions [4]. Therefore, to ensure mitigation strategies and adaptation efforts to combat the climate change, we consider in our IoT architecture renewable energy sources like solar, wind and small hydro; that charge IoT batteries by enabling sensors to harvest energy. Moreover, by enabling battery-powered IoT devices, we forecast future wireless battery charging for IoT devices that simplifies their design and shortens their time-to market, which in turn makes our contribution unique compared against [36], [37], and [38].

Compared with existing research on energy consumption in fog-enabled networks, which primarily focused on minimizing the overall energy consumed by the task offloading services, our approach introduces an online Deep Reinforcement Learning task assignment and scheduling scheme for optimizing IoT network performance, minimizing the energy demand and consumption—in the scenarios with battery-powered distributed IoT nodes, offering predictive behaviors on the network, and avoiding the impact of failures. Additionally, the SDN capabilities provided by the controller, e.g., logically centralized control, global view of the network, software-based traffic engineering, and dynamic updating of forwarding rules, make it straightforward to apply deep reinforcement learning for fully automated tasks assignments and scheduling in IoT network. Specifically, our approach offers a fully automated service deployment and resource/capacity planning mechanisms for fast-path forwarding across ultra-low latency SDN-enabled virtualized fog infrastructure. Our SDN-based solution offers a programmable analytic to the application layer through open interfaces to instantiate service intelligence at the network edge.

3. Model for Task Assignment and Scheduling problem

This section delves into the architectural details that enable us to support task assignment and scheduling, dynamic, and flexible resource management with our SDN-based framework, and presents the problem statement and the algorithms to instantiate service intelligence at the network edge.

3.1. System's Architecture

Figure 1 glances at the architectural design of our deep reinforcement assignment and scheduling solution to address the task scheduling problem in the IoT network. We added the task scheduler at the SDN controller level to find and select the best scheduling decision policy. The algorithm that describes the task scheduler consists of a queue containing task processing requests from mobile IoT applications, a learning-based input representative, and a planning decision-maker based on learning. Specifically, the dashed lines represent OpenFlow messages (i.e., FlowMod, OpenFlow Packet-IN and Packet-OUT, statistics such as OFPMP_FLOW that carry out Information about individual flow entries, as described by the OpenFlow Switch Specification [41]) exchanged between the Ryu SDN controller and the underlying Fog nodes, which are the data plane in our example. The messages are the continuous lines between the task scheduler and the SDN controller.

To usher the process, the SDN controller collaborates with our DRL algorithm to carry out intelligent network resource scheduling and management. The SDN controller uses a planner algorithm to manage task processing requests and create a historical dataset from incoming task requests. Figure 1 shows the internal controller modules: i) the path computing module assigns optimal route paths for different types of traffics generated by different IoT tasks and highly improves the QoS settings (bandwidth and delay); ii) the network monitoring module polls fog nodes to collect flow statistics to determine throughput, packet loss, and delay and; iii) the controller uses the flow scheduling module to exploit multiple paths to select the best QoS-aware path accordingly and uses queuing mechanisms to achieve optimal bandwidth utilization and supervise traffic activities. The SDN controller acts

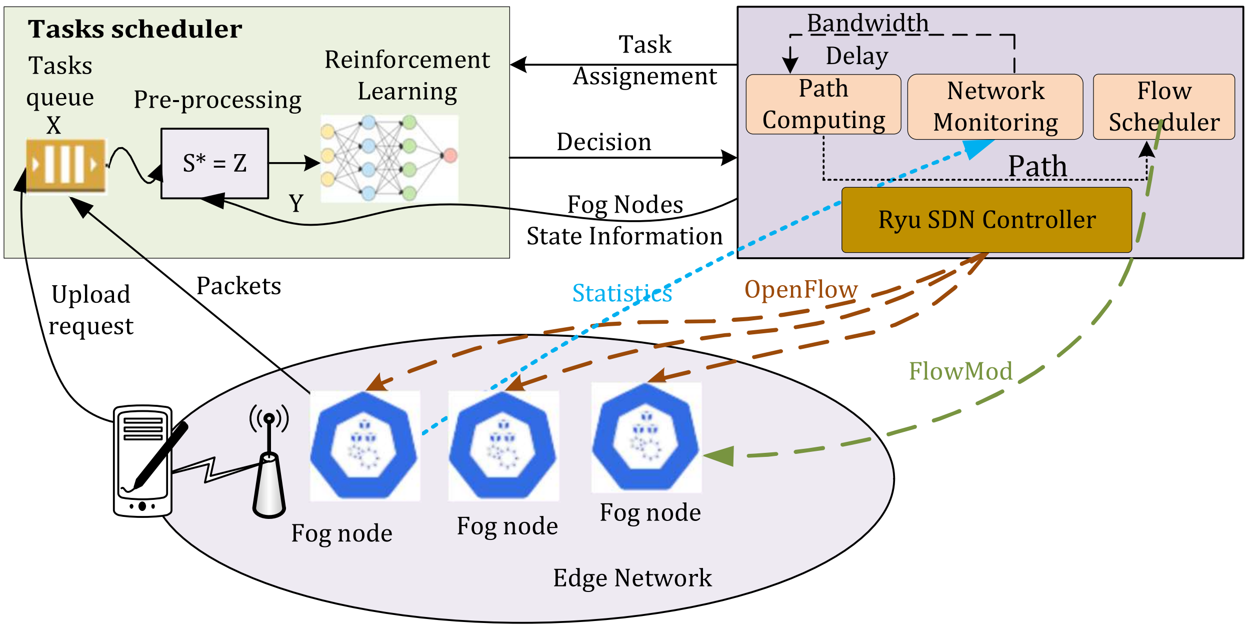

Figure 1: Task scheduling an architecture at a glance

as the brain of the network by controlling the underlying fog nodes through the OpenFlow secure channel.

Additionally, our agent in the controller SDN learns to use the dataset to represent the state information of all the fog nodes and the task requests to create a latent representation model. Once the information available on the dataset is well filtered and represented in a network graph, the SDN controller starts learning how to generate learning policies to input programming decision values (Q-value). Then, it selects the best fog node and sends the decision to OpenFlow rules for handling the tasks. Finally, the SDN controller assigns the fog nodes to process the task requests. After that, the mobile device can download data from the designated fog node.

3.2. *Problem Statement*

The task planning in fog computing is represented by N different tasks $\mathcal{T} = T_1, T_2, \ldots, T_N$, which will be assigned to distinct fog nodes $\mathcal{F} = F_1, F_2, \ldots, F_M$ to minimize energy consumption and time delay by optimizing the use of the transmission channel. Let us consider:

- $X_{ij}(t)$: denotes the assignment of task T_i on fog node F_j in respect of the time t.

$$X_{ij}(t) = \begin{cases} 1, & \text{if } T_i \text{ } runs \text{ } on \text{ } F_j \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

- Whenever a task is assigned to a fog node to run on it, it takes some latency time to execute in that node. We introduce the execution time of task T_i, when the task T_i is assigned to a fog node F_j, which can be obtained through the following equation.

$$TTC_{ij}(t) = D_i / C_j \tag{2}$$

in which, D_i stands for the number of instructions of a task T_i over time, and C_j is the CPU processing rate at fog node F_j.

The execution time cost in equation 2 describes the operating costs for the task assignment and indicates the complexity of the processing whenever the number of connected devices (i.e., requests) increase.

- The transmission time delay for task T_i on the fog node F_j is shown by equation 3:

$$TTR_{ij}(t) = \frac{D_i(t)}{r_{ij}(t)} ; \\ r_{ij}(t) = w_{ij}(t) \times \log(1 + \frac{h_{ij}(t) \times p_{ij}(t)}{\sigma}) \tag{3}$$

Where: the transmission time TTR_{ij} is the time required to transmit data in the uplink by a given IoT fog node; we assume that all nodes have the same bandwidth $w_{ij}(t)$; the uplinks quality of all fog nodes is given by SINR, which is computed based on parameters $w_{ij}(t)$, $h_{ij}(t)$, and σ in Equation (3), where: $h_{ij}(t)$ is the channel power-gain, $p_{ij}(t)$ is the transmission power, and σ is the noise power. The fog transmission service rate in equation 3 indicates the communication time of task offloading while taking into account the wireless path fading based on the channel characteristics and the noise power spectrum density, as well as the available bandwidth per fog node.

- The queuing delay for the task T_i is denoted by equation 4:

$$TTW_{ij}(t) = \frac{D_i(t)}{w_{ij}(t)} \frac{\tau}{1-\tau} ; \\ \tau = \frac{D_i(t) \times a_{ij}(t)}{w_{ij}(t)} \tag{4}$$

  Where $a_{ij}(t)$ is the average packet rate. The queuing delay, *aka.* residence time, indicates how many persistent tasks in the queue to offload to a neighboring node based on the remaining queue size and incoming tasks requests. τ stands for the traffic intensity, which is a measure of the average occupation of a task during a given period of time.

- The total time delay is given by equation 5:

$$TT_{ij}(t) = TTR_{ij}(t) + TTC_{ij}(t) + TTW_{ij}(t) \tag{5}$$

  Where $TTW_{ij}(t)$ is the queuing delay for task scheduling.

- Similarly, the energy consumption is denoted by equation 6:

$$EC_{ij} = TTR_{ij}(t) \times p_{ir}(t) + TTC_{ij}(t) \times p_{ie}(t) \tag{6}$$

  Where $p_{ir}(t)$ is the transmission power, and $p_{ie}(t)$ is the idle power.

We model the task scheduling problem as a nonlinear multi-objective combinatorial optimization problem with several objectives. The objective function is multi-variables and multi-constraints. That is, it is tricky to find an optimal solution using a polynomial method. Thus, there is a compelling need to design a hybrid heuristic algorithm proposed in this section to build a task scheduling strategy. In simplifying the complexity of the problem into a single objective problem and reducing the difficulty of solving, we consider the following hypotheses:

- Each task is independent, and there is no constraint between the tasks.
- Each task can be assigned to only a fog node.
- No task can be allocated repeatedly.
- In the calculation process, the task doesn't consider the impact of the mobility of the terminal equipment.
- All nodes are static, and the current task cannot be interrupted.

The objective function of task scheduling in fog nodes is shown in equation 7, where both time delay and energy consumption constraints are formulated as follows:

$$f = \min \sum_{i=1}^{N} (W_{it} \sum_{j=1}^{M} [X_{ij}(t) \\ \times TT_{ij}(t)] + W_{ie} \sum_{j=1}^{M} [X_{ij}(t) \times EC_{ij}(t)]) ; \tag{7}$$

Subject to:

$$\sum_{i=1}^{N} \sum_{j=1}^{M} EC_{ij}(t) \leq EC_{max} ; \\ \sum_{i=1}^{N} \sum_{j=1}^{M} TT_{ij}(t) \leq TT_{max}$$

Where, EC_{max} is the maximum energy consumption of our system, which represents the sum of the batteries available, TT_{max} is the maximum delay of our system, W_{it} is the weight of delay, and W_{ie} is the weight of energy consumption. Both weighting factors emphasize the importance of each type of constraint. In other words, the choice of weights in such multi-objective optimization approaches alludes to the decision-maker's preference.

4. Deep Reinforcement Learning for resolving Task Scheduling Problem

The DRL learning algorithm consists of objective (goal) oriented training and learning technique. An agent learns the optimal policy actions to interact with the environment and rewards it for every action it takes. Specifically, we define an action a belonging to a set of actions $A = \{Serve, Forward, Encapsulate, Discard\}$, that corresponds to the action of serving the current request and send it to normal processing pipeline, forwarding it to another port for scheduling to another neighbor node, encapsulate and forward to controller, and discard the request whenever resources are no longer available. A fog node tries to maximize its cumulative rewards and adapt to its environment to achieve the goal. The observation space (i.e., the state space) representing the environment at a given step describes the state of task requests to access a given service, e.g., number of task requests, number of granted requests, and number of allocated resources for that task request.

Figure 2 illustrates our approach to tackle the issue scheduling problem using deep reinforcement learning. Because the task planning module contains a small amount of information about the future arriving tasks, such as arrival time and task size, the SDN controller uses historical tasks to build the deployment decisions. The DRL algorithms we implemented inside the controller can analyze the performance of all connected fog-enabled IoT nodes. In doing so, we build efficient scheduling to execute several simultaneous tasks and predict optimal scheduling on the network that meets both low-latency and efficient-energy requirements.

First, we train the SDN controller to represent the datasets of all tasks and fog nodes for performing task assignments using the best optimal way under the constraints mentioned above, minimizing the network latency and reducing the energy consumption. Therefore, we introduce our DRL algorithm to select and apply the best decision that distributes the tasks on available fog nodes. As shown in Figure 2, the compressed low-dimensional representation of the input is used to find a latent representation of the data between tasks that will be executed and fog node states ready to execute these tasks. Then, an auto-encoder model has been developed, which aims to find a latent representation Z from data X using an encoder and decoder networks. The main goal of this model is to compute the following function $g(x) = sg(Wx + b)$, where sg is an activation function as $sigmoid()$, W is the weight, and b is the bias. After that, a bottleneck layer $Z = g(x)$ filters the incoming data from the encoder layer. Then, a decoder function defined as follows $f(x) = sg(W'z + b')$ is used to reconstruct the input X from Z (representation of latent space). The auto-encoder model is trained using the mean squared error (MSE), which minimizes the reconstruction error between the input X and the reconstructed input (output) X'. Furthermore, the obtained latent representation Z, obtained by the encoder network, i.e., $S^* = Z = g(S)$, is used to train the SDN controller to assign the task T_i to the node F_j and generate the optimal decision to schedule the tasks.

4.1. Task Assignment

Algorithm 1 illustrates the task assignment we implemented inside the SDN controller, which collects information from the underlying SDN routers about the available fog node capacities, including their available energy. Thence, the algorithm receives a list of tasks and their characteristics, and then assigns them to the available fog nodes. The DRL algorithm selects the fog nodes based on their available energy and current occupation rates to reduce the delay in processing time. Once the controller has assigned tasks to their relevant nodes, it keeps track of a log dataset of the current node's processing and available energy.

Whenever a task is successfully assigned to a fog node, the controller increases the value of the local reward and selects the forthcoming action according to the expected reward. Then, to maximize the objective function (e.f., equation 7), the algorithm applies the *argmax* operator to find the maximum values that fulfill low-energy consumption constraints, and lower network latency.

4.2. Deep Q-Learning Policy Algorithm

First, we introduce our Deep Q-Learning policy using experience replay to learn a small data block to avoid bias-

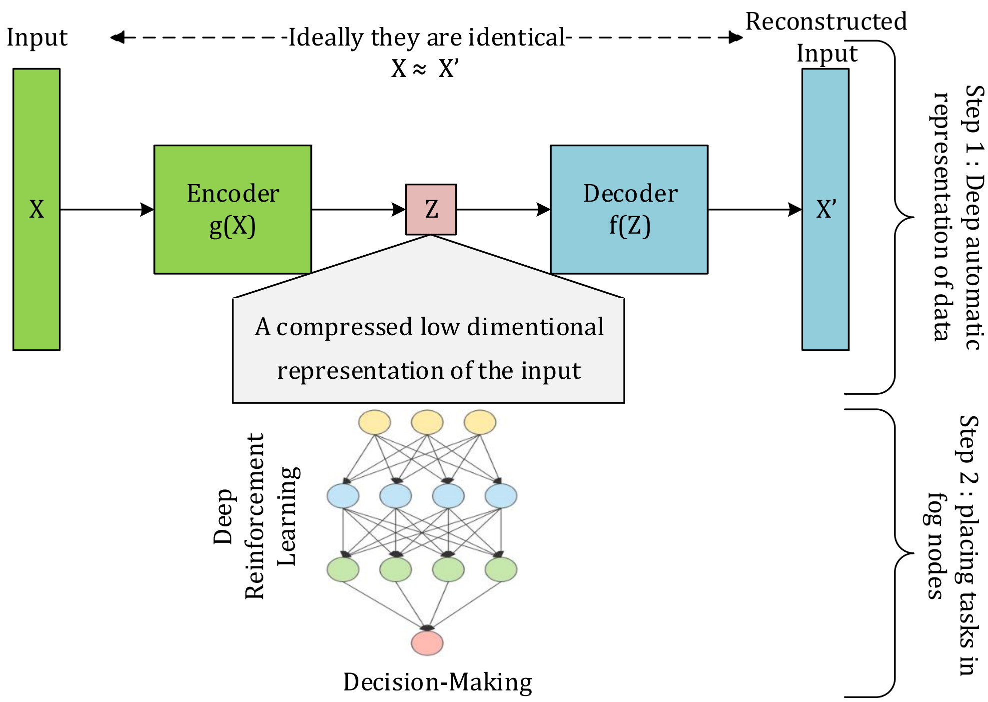

Figure 2: Deep Learning for Task Scheduling

ing the dataset distribution. Then, the algorithm approximates the Q-value function to explore all the states and determine all possible actions to reach the optimal solution. The algorithm uses continuous learning through the experience replay to update the algorithm parameters based on the previous actions. As described in algorithm 2, the SDN controller implements a Deep Q-learning algorithm to mimic a learning agent that maps states of the environment to actions. The agent considers these actions to move from one state to another to maximize a numerical reward over time.

Specifically, the SDN controller selects these actions during run-time, even if an agent doesn't complete the knowledge of rewards and state transition functions. In each state, the agent can choose between two types of behavior: (*i*) the controller can continue exploring the state space to find optimal decision policy; or (*ii*) it can leverage the information already given by the Q values defined by equation 8:

$$\begin{aligned} Q(S^*) &= R + \gamma \max Q(s', a') \\ Action\ a &= \arg\max_{a'} Q(s, a') \end{aligned} \tag{8}$$

The total reward is given by equation 9:

$$\begin{aligned} R = \sum_{i=1}^{n} (W_{it} \sum_{j=1}^{m} \beta[X_{ij}(t) \times TT_{ij}(t)] \\ + W_{ie} \sum_{j=1}^{m} \beta[X_{ij}(t) \times EC_{ij}(t)]) \end{aligned} \tag{9}$$

Where W_{it} is the delay weight and W_{ie} is the weight of energy consumption. The parameter β takes a positive sign if we execute the assigned task in the node, whereas a negative sign otherwise remains pending in the queue. We implemented the training algorithm that uses a regression loss-function to reduce the total training data error. Worthy of mention, the deep learning neural network loss

Algorithm 1: Task Assignment to Fog Nodes

Input:

1. Detection nodes $N = \{n_1, n_2, \ldots, n_j\}$ with their available energies,
2. Set of tasks $T = \{t_1, t_2, \ldots, t_i\}$ with their characteristics

Output: Assign a task t_i to a node n_j

while *true* **do** `// infinite loop`
`// learn according to cases`
Replay (n, t) ;
`// Predict the value of the reward`
act-values = predict (n, t) ;
`// Choose the action according to the expected reward`
$a = \arg\max$(act-values[0]) ;
Execute a `// Send t to n`

Algorithm 2: Deep Q-learning algorithm with experience replay

Input: Initialize replay memory M
Initialize action-value Q with random weights
Observe the initial state s
Output: A trained model to optimally assign a task to node

repeat
Select an action a with probability ϵ
Select a random action;
Otherwise select $a = \arg\max_{a'} Q(s, a')$
execute action a
observe reward r and new state s'
`// Store experience in memory`
store exp $< s, a, r, s' >$ in memory M
sample random transitions $< ss, aa, rr, ss' >$ from memory M
compute target for each minibatch transition
if ss' *is terminal state* **then**
$tt \leftarrow rr$
else $tt \leftarrow rr + \gamma \max'_a Q(ss', aa')$

Train the Q network using $(tt - Q(ss, aa))^2$ as loss
$s = s'$
until *terminated*

function, to predict the states of Q values, given by equation 10 with a hyper-parameter setting γ as a discount factor :

$$L = \frac{1}{2}[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]^2 \tag{10}$$

Algorithm 2 learns the allocation policy to provide an optimal decision regarding both the constraints mentioned above, i.e., in terms of latency and energy and performance of the system. First, to start the learning process, the algorithm initializes a decision matrix with weights (Q-values) of random policies and observes initial states of the SDN network. Then, nodes will be chosen with their smallest probability values from the matrix, which already filled and assigns the tasks randomly to its distinct fog nodes. Once the first step is completed, the controller can move to the following states, i.e., returning a reward and performing the calculation, and transitioning from one state to another. Each newly calculated step is saved in the matrix, and we compare the existing policy with the previous one. If the newer policy is better, it will be considered locally, optimal, and so on. Thus, we repeatedly operate until we get an optimal global assignment for each task. Then, the operation is repeated until all tasks in the waiting queue are processed and assigned to the best available fog nodes.

4.3. Random Learning Policy

We introduced dense random neural networks that randomly process a data block while simultaneously improving the learning policy's robustness and accuracy. Algorithm 3 represents the assignment of tasks using a random agent. The latter is based on a strategy that assigns the tasks one by one to the various available fog nodes randomly without taking into account the quality of service in terms of latency. It assigns each task to a randomly chosen fog node and updates its energies each time. In addition, it also updates the accumulated rewards.

4.4. Deterministic Learning Policy

We developed a deterministic learning policy to dictate what action to take given a particular state. Indeed, we consider a different situation where incoming IoT tasks are known to forecast the next event precisely from the

Algorithm 3: Agent Random

Input:
1. Detection nodes $N = \{n_1, n_2, \ldots, n_n\}$ with their available energies,
2. Set of tasks $T = \{t_1, t_2, \ldots, t_m\}$ with their characteristics

Output: Assign task t_i to node n_j

```
i ← 1
reward ← 0
totrewards ← 0
while i <= m do // m is the number of tasks
    action ← env.action_space.sample() // randomly choose an action

    ob, reward ← env.step(action)
    update(N,ob) // update of nodes

    totrewards+ = reward
    i + +
```

Algorithm 4: Agent deterministic

Input:
1. Detect nodes $N = \{n_1, n_2, \ldots, n_n\}$ with their available energies,
2. Set of tasks $T = \{t_1, t_2, \ldots, t_m\}$ with their characteristics

Output: Assign task t_i to node n_j

```
Sorted(N) i ← 1
assign ← false
reward ← 0
totrewards ← 0
while i <= m do // m is the number of tasks
    for j ← 1 to n do // n is the number of nodes
        if N[j]["energ"] > T[i]["energ"] then
            N[j]["energ"] ← N[j]["energ"] − T[i]["energ"]
            ob, reward ← Execute(T[i], N[j]) // execute task in node
            assign ← true
            break
    if !assign then
        ob, reward ← env.step()
    update(N,ob) // update of nodes
    Sort(N)
    totrewards+ = reward
    i + +
```

current event. The value of the state is the expected reward if we start from it and continue using the same policy. Nonetheless, the deterministic policy does not involve that the reward remains the same.

Algorithm 4 represents the assignment of tasks to the corresponding nodes using a deterministic agent. The latter will schedule the tasks one by one according to their order of arrival and assign them to nodes close to their minimum latency. At each iteration, an update will be performed on the energy state of the node that executed the task.

4.5. Asynchronous Actor-Critic Agent (A3C) Algorithm

We design Asynchronous Advantage Actor-Critic (A3C) Algorithm to involve global network optimization in parallel by using multiple agents. These agents have their own set of parameters to create different situations to interact with the fog nodes distributed in the environment. Each agent harvests a different learning experience and add it to the overall learning experience.

Algorithm 5 illustrates the task assignment and scheduling scheme using the A3C approach, which relies on multiple agents that likely explore different states and transitions. Each agent has its own network parameters and a copy of the environment.

These agents interact with their respective environments asynchronously while learning at each iteration. Thus, each agent gets its own copy of the environment processes the gathered data samples at their arrival. The main thrust of A3C is that the network controls each agent to gain more acknowledge and contribute to the complete knowledge of the network.

Algorithm 5 (*lines 20-24*) illustrates the policy update we developed using the A3C approach. As shown in *line 21*, the A3C agents select different actions in order to

Algorithm 5: ASYNCHRONOUS ADVANTAGE ACTOR-CRITIC AGENT (A3C)

Output: Model trained with workers to assign task to node.

1 **for** $i \leftarrow 1$ ***to*** n **do** `// n is the number of workers`
2 $\quad Wi$.run() `// Start worker thread`
3 $step \leftarrow 1$ `// ForEach worker Wi, initialize step counter`
4 $T \leftarrow 0$ `// Initialize episode counter`
5 **repeat**
6 $\quad d\theta \leftarrow 0; d\theta_v \leftarrow 0$ `// reset gradients`
7 $\quad \theta' \leftarrow \theta; \theta'_v \leftarrow \theta_v; t \leftarrow step$ `// Synchronize thread-specific parameters`
8 $\quad s \leftarrow s_t$ `// Initialize iteration` `// Get observation state`
9 $\quad$ **while** *s is not terminal and $step - t < t_{max}$* **do**
10 $\quad\quad$ Simulate action a_t according to $\pi(a_t|s;\theta)$
11 $\quad\quad$ Receive reward r_t and next state s_{t+1}
12 $\quad\quad step++;$
13 $\quad\quad T++;$
14 $\quad$ **if** *s_t is terminal state* **then**
15 $\quad\quad R \leftarrow 0$
16 $\quad$ **else** $R \leftarrow V(s_t, \theta'_v)$ `// Bootstrap from last state`
17
18 $\quad$ **for** $i \leftarrow step - 1$ ***to*** t **do**
19 $\quad\quad R \leftarrow r_i + \gamma R$
20 $\quad\quad d\theta \leftarrow d\theta + \Delta_{\theta'} log(\pi(a_i|si;\theta')(R - V(s_i;\theta'_v)))$ `// Accumulate gradients`
21 $\quad\quad d\theta_v \leftarrow d\theta_v + \frac{\partial((R-V(s_i;\theta'_v))^2)}{\partial\theta'_v}$
$\quad$ `// Perform asynchronous update of` θ `and` θ_v
22 $\quad \theta = \theta + d\theta$
23 $\quad \theta_v = \theta + d\theta_v$
24 **until** $T > T_{max}$

maximize the discounted reward R by updating the hyper-parameter settings such as discount factor γ. The agents try to maximize the immediate rewards by taking greedy actions using the policy function π and the Value function V to impact future global parameters vectors $d\theta$, as shown in *line 22*. The update operation is performed until reaching the maximum number of predefined iterations T_{max}.

5. Performance Analysis

This section describes the testbed setup and shows the evaluation of our solution. Specifically, we describe the results for commonly used evaluation metrics, such as latency, energy efficiency, and network scalability.

5.1. Testbed Setup

We implemented our framework using an emulated SDN environment comprising Mininet [42] as our network emulators along with OpenFlow SDN switches for creating different IoT scenarios. Furthermore, we extended Mininet to support Open AI Gym toolkit [43] for reinforcement learning and deployed IoT nodes in the form of virtualized micro-services using Docker containers in emulated Kubernetes clusters. We also implemented the SDN northbound application using Python-based Ryu [44] SDN controller, which performs global traffic management, load balancing, global topology discovery, and monitoring. We developed our solution using the TensorFlow python interface for interacting with our SDN environment. In the latter, we used to run tests on more than 100 nodes, each running over $1,000$ tasks simultaneously. We assessed our solution versus deterministic algorithms, random, and A3C approaches. The deterministic agent plans tasks according to their order of arrival and assigns them to the nearest nodes regarding their minimum latency. Whereas the random agent assigns tasks to the available nodes in a stochastic order, i.e., it assigns tasks to the available nodes' strategy less. That is, if a selected node does not have the required capacity to execute an incoming task, then the random agent leaves it in-hold state in the waiting queue and assigns tasks to alternative nodes. Thus, the A3C approach uses multiple agents who independently learn a policy from their environment and then collaborate with other agents to create global knowledge to choose the best decision. The detailed simulation hyperparameters are given in Table 2.

Hyperparameter	Value
Learning rate	0.001
Gamma	0.95
Batch size	32
Exploration Max	1.0
Exploration Min	0.01
Exploration Decay	0.995

Table 2: Hyperparameter values in simulation

5.2. *Pre-processing*

The first step we performed on our datasets comprises pre-processing input data to carry out the training of our SDN controller (i.e., the Ryu controller). Carrying out data refinement allows properly representing and preparing data for our deep Q-learning model to perform task assignments and scheduling. Therefore, we implemented different techniques to reduce the dimension of our datasets to find the best representation of our data.

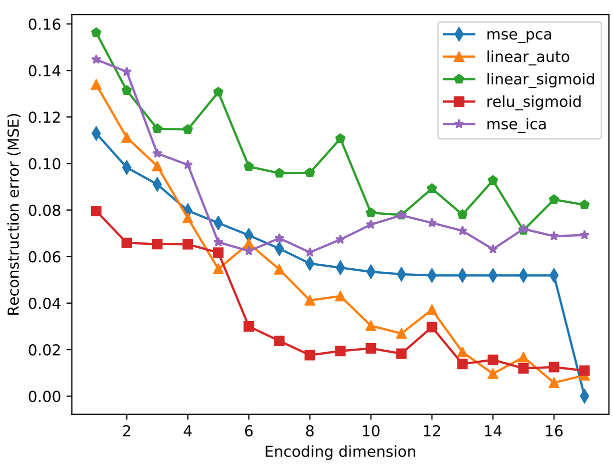

Figure 3: Data Pre-Processing

Figure 3 illustrates the Mean Square Error (MSE) regression loss function. This yield was obtained for different refinement techniques, including Principal Component Analysis (PCA), Independent Component Analysis (ICA), Deep auto-encoder with Sigmoid function, rectified linear activation function (reLU), and linear function. As underscored by Figure 3, the Deep auto-encoder with Sigmoid activation function (i.e., the Relu-Sigmoid) performs better filtering and refinement results while keeping the MSE error minimum. We create multiple local minima to find optimal task assignment strategies. The sigmoid function makes the loss function non-convex, rather than creating a single global minimum for our training.

5.3. *Discounted Cumulative reward*

The SDN agent in run-time collects states from the environment and sends back information to the controller. By trying different actions, the agent learns to optimize the reward from the environment. The controller can either decide to take the current policy as the best decision to place tasks on the selected fog-enabled nodes, or continue learning from the available distributed nodes to find a better candidate to place the current task requests. All agents are driven by the same goal, to maximize the expected discounted return.

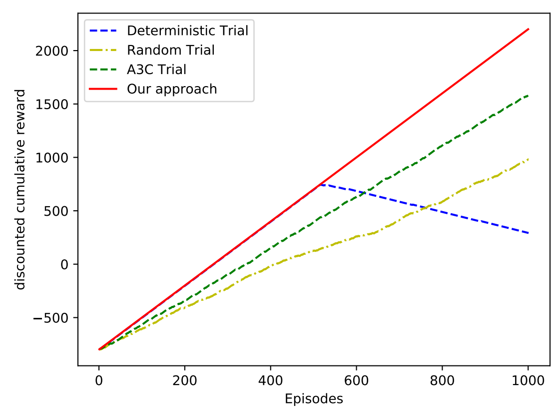

Figure 4: Cumulative Rewards for Our approach against the three other approaches.

Figure 4 compares the accumulated reward got by our approach versus the deterministic, random, and A3C approaches. After the deterministic agent increases to almost 550 earned rewards, it decreases and starts losing rewards. It ushers, losing its computation power and its ability to complete the planning of newer coming tasks. As a result, his cumulative reward curve slowly increases for the random agent, which means some tasks have not been assigned, and we must put them on hold state. For the A3C agent, its cumulative rewards slowly increase compared to our approach. Our approach performs better cumulative rewards than the deterministic and random case, selecting the available node based on the energy level. For the A3C trial agent, it has a lower accumulated reward compared to our approach. The update of the cumulative reward curve of our agent is increasing rapidly compared to other agents. Thus, the agent has a very optimal investment strategy where the action is selected each time. It motivates it to maximize the rewarded return.

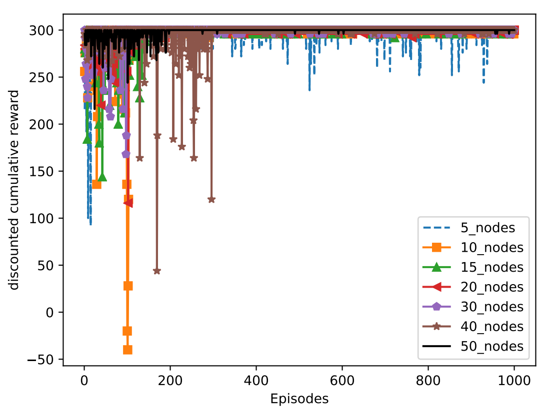

Figure 5: Local Rewards with Increasing number of nodes

To evaluate the stability and the scalability of our approach, we increased the number of fog nodes up to 50, as shown in Figure 5. We observed that our SDN-enabled decision-maker agent could quickly learn from the SDN topology network to make optimal decisions. As a result, the local reward, i.e., optimal local assignment, rapidly reaches almost 300 in a few dozen episodes as illustrated in Figure 5, which means: that optimal local minimum, i.e., local optimization, can be performed rapidly. We also experimented with our approach with over 100 in other scenarios (none shown in Figure 5), and we observed the same behavior. Thus, we claim that our deep learning approach is gainful to implement both local, optimal and global tasks assignments and schedules for SDN-enabled IoT networks while respecting QoS constraints.

5.4. Energy-Efficiency

We aim to reduce the energy consumption on running fog-enabled IoT nodes as we described in equation 6 and perform better energy efficiency as we considered all nodes as batteries-powered ones. To evaluate the energy efficiency of our SDN-enabled solution, the SDN controller trained the agent by $1,000$ episodes. As a result, the agent should be able to plan 100 tasks to energy-constrained fog nodes. Each fog node has a limited power capacity, i.e., their battery level during this planning process is close to $5,000wh$.

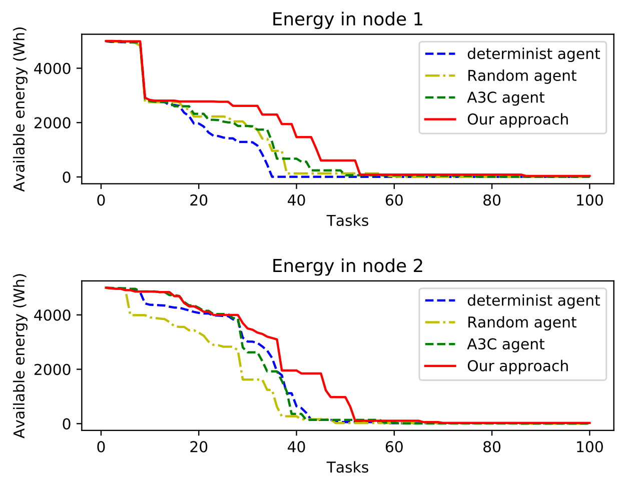

Figure 6: Energy Consumption in both tasks' executing fog nodes

Figure 6 illustrates the energy consumption of two available fog nodes, each running up to 100 tasks simultaneously, using our training approach against deterministic, random, and A3C training agents. Throughout the planning strategy of these tasks, the DRL agent in our approach keeps a better battery level in both nodes compared to the three other algorithms.

Recall that our primary objective is to minimize the overall energy consumption of our SDN-enabled fog network, as we described in equation 6. Figure 7 shows that our approach flags out better energy efficiency, i.e., up to 87% compared against both deterministic agents, which perform 48%, and the random agent that performs 58%, and A3C agent, which performs 76%.

Our results confirm our claims that the solution we propose can readily be used to optimize task scheduling and dynamically assignment of complex jobs with task dependencies in distributed fog IoT networks.

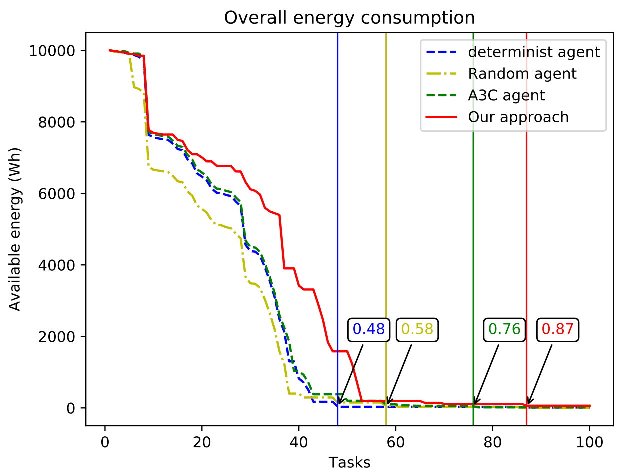

Figure 7: Energy Saving and Efficiency for all available Fog nodes

5.5. *Assessing the Latency performance*

Our optimization approach aims at minimizing the network latency for available nodes during the task executions, as we described in equation 5. We gauged the total time delay expected by available fog-enabled nodes to process the current task's requests and communicate the results to remote IoT senders.

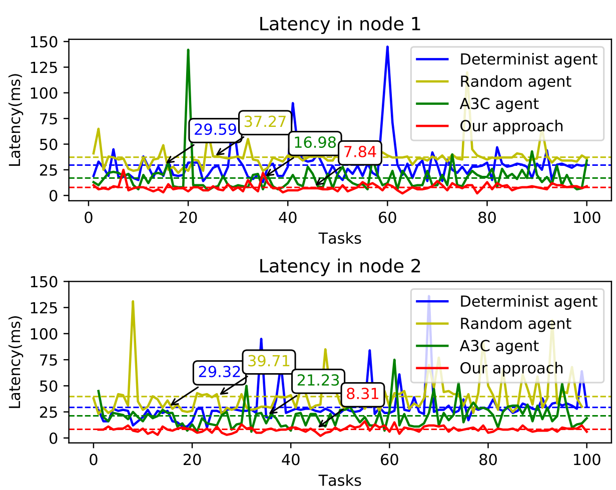

Figure 8: Evaluating Latency during Task Scheduling

We measured the total delay expected by the available fog nodes to handle the pending task requests. The set of latency values collected during task scheduling of our approach, as well as with other approaches to scheduling tasks in two battery-powered nodes, as shown in Figure 8.

The latency results of the different approaches have been summarized in Table 3. As expected, the deterministic approach performed a time latency of 29.59 ms in node 1 and 39.32 ms in node 2. Likewise, the random algorithm carried out 37.27 ms in node 1 and 39.71 ms in node 2. Finally, the A3C algorithm presents an average latency of 16.98 ms at node 1 and 21.23 ms at node 2. We repeated these experiments multiple times, and we find the average latency expected by our approach is close to 7.84ms in node 1 and 8.31 ms in node 2. Therefore, our approach ensured a minimum latency of all fog nodes and showed significant latency and energy consumption performances.

Agent	node 1	node 2
A3C	16.98	21.23
Random	37.27	39.71
Deterministic	29.59	39.32
Our approach	7.84	8.31

Table 3: Average latency

5.6. *Evaluating the Bandwidth performance*

To assess the performance of our approach, we studied the bandwidth usage of our IoT network during the task scheduling process. Figure 9 illustrates the bandwidth usage during the scheduling by the different approaches described in section 4. According to Figure 9, our approach outperforms all the other approaches during task scheduling. Compared against both random and deterministic approaches, which performed *29.6* Gbits/s and *32.15* Gbits/ respectively, our approach outperformed both of them. The reason is that the deterministic approach uses a deterministic greedy policy to make the locally optimal choice at each stage without exploration. Furthermore, in our IoT network, the deterministic algorithm will stick to the current state during that task assignment step, when it is better than the observed states. Nonetheless, the deterministic greedy policy will find itself trapped in a local optimum, failing to explore certain other states, which may hold a better local optimum solution. Similarly, the randomized exploration approach (randomized policy) will explore different states based on a specific probability distribution, making it slightly slower.

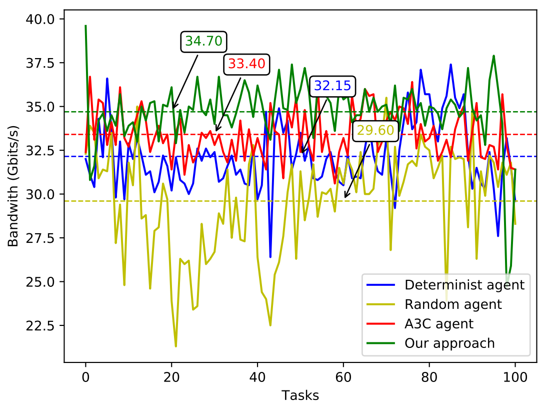

Figure 9: Evaluating The bandwidth utilization during task Scheduling

Unsurprisingly, our approach reaches *34.70* Gbits/s, which outperforms the *33.4* Gbits/s obtained with the A3C approach, even when we use many parallel workers (agents) that interact asynchronously with separate instantiations of the environment. The A3C technique tends to suffer when faced with more complex tasks, as it takes long delays between actions and relevant reward signals, i.e., known as Partially Observable Environments.

6. Conclusion

This paper showed the feasibility of developing task assignment and scheduling mechanisms for SDN-enabled IoT networks using Deep Reinforcement Learning. We formulated a task assignment and scheduling problem that minimizes network latency while ensuring energy efficiency. The evaluation of our solution against deterministic placement algorithm, random, and A3C strategies showed it outperforms these algorithms in selecting optimal allocation decision policies for task assignments and scheduling in real-time. Furthermore, our approach performed both local and global optimization, ensuring lower-latency communication and increased energy efficiency.

We claim we can extend our DRL algorithm to support intelligent multi-access Ultra-Dense Edge Computing (UDEC) to utilize multiple 5G resources efficiently. Our future work will develop a Federated Machine Learning (FedML) approach to solve data ownership and privacy by training statistical security models in UDEC nodes while keeping data samples localized inside fog nodes. This promising undertaking will expand the capacity of federated learning to keep individual data sets localized inside fog nodes while updating main model parameters and distributing them back to all edge nodes.

Acknowledgments

This work was partially funded by the Tunisian Ministry of Higher Education and Scientific Research (MES) under the Young Researchers Incentive Program (19PEJC09-04) and the NGI Explorers Program under the Horizon 2020 Research and Innovation Framework (H2020), Grant Agreement number: 825183, Call identifier: H2020-ICT-31-2018. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of MES, NGI or H2020.

References

[1] F. Shan, J. Luo, J. Jin, W. Wu, Offloading delay constrained transparent computing tasks with energy-efficient transmission power scheduling in wireless iot environment, IEEE Internet of Things Journal 6 (3) (2019) 4411–4422.

[2] M. Qin, N. Cheng, Z. Jing, T. Yang, W. Xu, Q. Yang, R. R. Rao, Service-oriented energy-latency tradeoff for iot task partial offloading in mec-enhanced multi-rat networks, IEEE Internet of Things Journal (2020) 1–1.

[3] H. Hejazi, H. Rajab, T. Cinkler, L. Lengyel, Survey of platforms for massive iot, in: 2018 IEEE International Conference on Future IoT Technologies (Future IoT), IEEE, New York, NY, USA, 2018, pp. 1–8.

[4] 6gworld, Sustainability in new and emerging technologies (April 2021).

[5] M. Huang, W. Liu, T. Wang, A. Liu, S. Zhang, A cloud–mec collaborative task offloading scheme with service orchestration, IEEE Internet of Things Journal 7 (7) (2020) 5792–5805.

[6] Y. Liu, Z. Zeng, X. Liu, X. Zhu, M. Z. A. Bhuiyan, A novel load balancing and low response delay framework for edge-cloud network based on sdn, IEEE Internet of Things Journal 7 (7) (2020) 5922–5933.

[7] M. K. Hussein, M. H. Mousa, Efficient task offloading for iot-based applications in fog computing using ant colony optimization, IEEE Access 8 (2020) 37191–37201.

[8] E. I. A. (EIA), International energy outlook (September 2019).
URL `https://bit.ly/2CFN9QK`

[9] E. E. Agency, Progress on energy efficiency in europe (Feb 2019).
URL `https://bit.ly/2OygVJN`

[10] P. Cai, F. Yang, J. Wang, X. Wu, Y. Yang, X. Luo, Jote: Joint offloading of tasks and energy in fog-enabled iot networks, IEEE Internet of Things Journal 7 (4) (2020) 3067–3082.

[11] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, Y. Liu, A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges, IEEE Communications Surveys Tutorials 21 (1) (2019) 393–430.

[12] Q. D. La, M. V. Ngo, T. Q. Dinh, T. Q. Quek, H. Shin, Enabling intelligence in fog computing to achieve energy and latency reduction, Digital Communications and Networks 5 (1) (2019) 3 – 9.

[13] H. Yuan, J. Bi, M. Zhou, K. Sedraoui, Warm: Workload-aware multi-application task scheduling for revenue maximization in sdn-based cloud data center, IEEE Access 6 (2018) 645–657.

[14] L. Yang, X. Liu, J. Cao, Z. Wang, Joint scheduling of tasks and network flows in big data clusters, IEEE Access 6 (2018) 66600–66611.

[15] J. Zhou, H. Jiang, J. Wu, L. Wu, C. Zhu, W. Li, Sdn-based application framework for wireless sensor and actor networks, IEEE Access 4 (2016) 1583–1594.

[16] C. Lin, G. Han, X. Qi, M. Guizani, L. Shu, A distributed mobile fog computing scheme for mobile delay-sensitive applications in sdn-enabled vehicular networks, IEEE Transactions on Vehicular Technology 69 (5) (2020) 5481–5493.

[17] Z. Wu, B. Li, Z. Fei, Z. Zheng, Z. Han, Energy-efficient robust computation offloading for fog-iot systems, IEEE Transactions on Vehicular Technology 69 (2020) 4417–4425.

[18] M. Chen, Y. Hao, Task offloading for mobile edge computing in software defined ultra-dense network, IEEE Journal on Selected Areas in Communications 36 (3) (2018) 587–597.

[19] M. Yang, H. Zhu, H. Qian, Y. Koucheryavy, K. Samouylov, H. Wang, Peer offloading with delayed feedback in fog networks, IEEE Internet of Things Journal 8 (17) (2021) 13690–13702. doi:10.1109/JIOT.2021.3067919.

[20] M. Yang, H. Zhu, H. Wang, Y. Koucheryavy, K. Samouylov, H. Qian, An online learning approach to computation offloading in dynamic fog networks, IEEE Internet of Things Journal 8 (3) (2021) 1572–1584. doi:10.1109/JIOT.2020.3015522.

[21] L. Pu, X. Chen, J. Xu, X. Fu, D2d fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted d2d collaboration, IEEE Journal on Selected Areas in Communications 34 (12) (2016) 3887–3901.

[22] Z. Kuang, L. Li, J. Gao, L. Zhao, A. Liu, Partial offloading scheduling and power allocation for mobile edge computing systems, IEEE Internet of Things Journal 6 (4) (2019) 6774–6785.

[23] G. Zhang, F. Shen, Z. Liu, Y. Yang, K. Wang, M. Zhou, Femto: Fair and energy-minimized task offloading for fog-enabled iot networks, IEEE Internet of Things Journal 6 (3) (2019) 4388–4400.

[24] S. C. G., V. Chamola, C.-K. Tham, G. S., N. Ansari, An optimal delay aware task assignment scheme for wireless sdn networked edge cloudlets, Future Generation Computer Systems 102 (2020) 862–875.

[25] S. Misra, N. Saha, Detour: Dynamic task offloading in software-defined fog for iot applications, IEEE Journal on Selected Areas in Communications 37 (5) (2019) 1159–1166.

[26] F. Al-Turjman, M. Z. Hasan, H. Al-Rizzo, Task scheduling in cloud-based survivability applications using swarm optimization in iot, Transactions on Emerging Telecommunications Technologies 30 (8) (2019) e3539.

[27] L. Lei, Y. Tan, K. Zheng, S. Liu, K. Zhang, X. Shen, Deep reinforcement learning for autonomous internet of things: Model, applications and challenges, IEEE Communications Surveys Tutorials 22 (3) (2020) 1722–1760.

[28] J. Wang, C. Xu, Y. Huangfu, R. Li, Y. Ge, J. Wang, Deep reinforcement learning for scheduling in cellular networks, in: 2019 11th International Conference on Wireless Communications and Signal Processing (WCSP), 2019, pp. 1–6.

[29] T. Sen, H. Shen, Machine learning based timeliness-guaranteed and energy-efficient task assignment in edge computing systems, in: 2019 IEEE 3rd International Conference on Fog and Edge Computing (ICFEC), 2019, pp. 1–10.

[30] H. Mao, M. Alizadeh, I. Menache, S. Kandula, Resource management with deep reinforcement learning, in: FIXME, 2016, p. 50–56.

[31] D. B. Noureddine., A. Gharbi., S. B. Ahmed., Multi-agent deep reinforcement learning for task allocation in dynamic environment, in: Proceedings of the 12th International Conference on Software Technologies - Volume 1: ICSOFT,, 2017, pp. 17–26.

[32] J. Wang, J. Cao, S. Wang, Z. Yao, W. Li, Irda: Incremental reinforcement learning for dynamic resource allocation, IEEE Transactions on Big Data 01 (01) (2020) 1–1.

[33] X. Ma, H. Gao, H. Xu, M. Bian, An iot-based task scheduling optimization scheme considering the deadline and cost-aware scientific workflow for cloud computing, EURASIP Journal on Wireless Communications and Networking 2019 (1) (2019) 249.

[34] H. Ye, G. Y. Li, Deep reinforcement learning for resource allocation in v2v communications, in: 2018 IEEE International Conference on Communications (ICC), 2018, pp. 1–6.

[35] S. Liang, Z. Yang, F. Jin, Y. Chen, Data centers job scheduling with deep reinforcement learning, Advances in Knowledge Discovery and Data Mining: 24th Pacific-Asia Conference, PAKDD 2020, Singapore, May 11–14, 2020, Proceedings, Part II 12085 (2020) 906–917.

[36] H. Liao, Z. Zhou, X. Zhao, L. Zhang, S. Mumtaz, A. Jolfaei, S. H. Ahmed, A. K. Bashir, Learning-based context-aware resource allocation for edge-computing-empowered industrial iot, IEEE Internet of Things Journal 7 (5) (2019) 4260–4277.

[37] Y. Sun, S. Zhou, J. Xu, Emm: Energy-aware mobility management for mobile edge computing in ultra dense networks, IEEE Journal on Selected Areas in Communications 35 (11) (2017) 2637–2646.

[38] H. Liao, Z. Zhou, X. Zhao, Y. Wang, Learning-based queue-aware task offloading and resource allocation for space–air–ground-integrated power iot, IEEE Internet of Things Journal 8 (7) (2021) 5250–5263.

[39] A. Haj-Ali, N. K. Ahmed, T. Willke, J. Gonzalez, K. Asanovic, I. Stoica, A view on deep reinforcement learning in system optimization (2019). arXiv:1908.01275.

[40] H. Tankovska, Number of internet of things (iot) connected devices worldwide in 2018, 2025 and 2030 (October, 26 2019).

[41] O. N. F. Open Network Foundation, OpenFlow Switch Specification (April 2013).

[42] B. Lantz, B. Heller, N. McKeown, A network in a laptop: Rapid prototyping for software-defined networks, in: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, Hotnets-IX, Association for Computing Machinery, New York, NY, USA, 2010, p. 6.

[43] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, Openai gym (2016). arXiv:1606.01540.

[44] R. project team, Ryu SDN Framework, FIXME, 2014.