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] *University of Tunis El Manar, Faculty of Sciences, Dept of Computer Sciences, Campus universitaire, BP 37, Tunis, 1002, Tunisia*
[d] *CNRS, LAAS, UPS, 7 Avenue du colonel Roche, BP 54200, Toulouse, F-31031, France*

Abstract

5G mobile network services have made tremendous growth in the IoT network. As a result, 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 task handling, energy conservation, and ultra-reliable low-latency variability.

This paper investigates an energy-aware and low-latency oriented computing task scheduling problem in a Software-Defined Fog-IoT Network. 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 SDN-enabled edge networks. We conducted comprehensive experiments and compared the presented 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 tasks with the available battery power by up to 50% more minor time delay. Our results support 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. A scrutiny of related work 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 sent by reviewing the task scheduling within an SDN-enabled edge computing.

1.1. Task Scheduling SDN-enabled Edge Computing

SDN has been widely used to empower dynamic and effective resource allocation in diverse cloud [1] and data center networks [2]. Furthermore, it provided on-demand application and resource management in wireless sensor networks [3] and network edge [4]. For example, Wu et al. [5] 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. [6] proposed an SDN-based heuristic model for offloading distributed computing resources in an ultra-dense network. They formulated the task offloading problem as a mixed-integer non-linear program to solve the task placement and resource allocation problem in mobile edge computing. Similarly, Pen et al. [7] introduced a mobile task offloading framework for device-to-device (D2D) fogging. They leveraged Lyapunov optimization D2D fogging methods for achieving 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. [8] 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. [9] 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 is to offer optimal transmission power for wireless fog-enabled mobile IoT nodes. Chalapathi et al. [10] proposed a Latency Aware Task Assignment (LATA) scheme for a multi-cloudlet network to optimize the latency monetary cost in computing the tasks of mobile devices by making optimal task assignments among the micro-clouds. The LATA model proposed an admission control policy to maintain optimality in high traffic conditions. Besides, the authors in [11] addressed the problem of task offloading in SDN-enabled network 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 an optimal decision on local or remote task computation, optimal fog node selection, and optimal path selection.

1.2. 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 [21]. Lei et al. [12] provided a comprehensive survey of automating and orchestrating IoT resources using reinforcement learning (RL) to achieve autonomy. Wan et al. [13] 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 networks. 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 is employed as expert knowledge to help with DRL agent training. Sen et al. [14] 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).

Article	Idea	Criteria			
		Energy effeciency	Scalability	Latency	Bandwidth
Lei et al. [12]	Automation and orchestration of IoT resources using RL & DRL in IoT Network	-	-	✓	✓
Wang et al. [13]	DRL-based planning for LTE cellular networks	-	✓	-	✓
Sen et al. [14]	Schedule application tasks in intelligent cognitive assistants distributed through machine learning	✓	✓	-	-
Hongzi et al. [15]	Introduced a DRL-based framework to automatically manage IOT resources from their own experience	-	✓	-	-
Dhoha et al. [16]	Improve resource sharing and task allocation using a cooperative process based on DRL	-	✓	-	-
Wang et al. [17]	Incremental approach based on DRL for task allocation strategies	-	✓	-	-
Ma et al. [18]	Task planning optimization scheme based on heuristic approaches	-	-	✓	-
H .Ye et al. [19]	Develop a decentralized resource allocation mechanism for V2V communications using multi-agent DRL	-	✓	✓	-
S. Liang et al. [20]	Introduce a mechanism to address the customized job scheduling problem in data centers using DRL	-	-	-	-
Our approach		✓	✓	✓	✓

Table 1: Reinforcing learning for Task Allocation

Hongzi et al. [15] introduced a DeepRM framework to build autonomous and intelligent systems that learn to manage resources from their own experience directly.

Likewise, the authors in [19] proposed a DRL approach for a decentralized resource allocation mechanism for 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. [16] 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. [17] 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 [20] introduced an RL approach for learning the scheduling policy automatically and reduce the estimation error on data centers. Similarly, Ma et al. [18] 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 of a workflow under deadline constraints in the infrastructure as a service (IaaS) model.

1.3. Paper's Contributions

Unlike the approaches above, which are mainly based on meticulously designed heuristics that ignore the patterns of incoming tasks, our approach uses 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, compared with existing researches for the 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. Furthermore, 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.

2. A scrutiny of related work 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 sent by reviewing the task scheduling within an 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 [1] and data center networks [2]. Furthermore, it provided on-demand application and resource management in wireless sensor networks [3] and network edge [4]. For example, Wu et al. [5] 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. [6] proposed an SDN-based heuristic model for offloading distributed computing resources in an ultra-dense network. They formulated the task offloading problem as a mixed-integer non-linear program to solve the task placement and resource allocation problem in mobile edge computing. Similarly, Pen et al. [7] introduced a mobile task offloading framework for device-to-device (D2D) fogging. They leveraged Lyapunov optimization D2D fogging methods for achieving 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. [8] 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. Zhang et al. [9] 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 is to offer optimal transmission power for wireless fog-enabled mobile IoT nodes. Chalapathi et al. [10] proposed a Latency Aware Task Assignment (LATA) scheme for a multi-cloudlet network to optimize the latency monetary cost in computing the tasks of mobile devices by making optimal task assignments among the micro-clouds. The LATA model proposed an admission control policy to maintain optimality in high traffic conditions. Besides, the authors in [11] addressed the problem of task offloading in SDN-enabled network 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 an optimal decision on local or remote task computation, optimal fog node selection, and optimal path selection.

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 [21]. Lei et al. [12] provided a comprehensive survey of automating and orchestrating IoT resources using reinforcement learning (RL) to achieve autonomy. Wan et al. [13] 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 networks. 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 is employed as expert knowledge to help with DRL agent training. Sen et al. [14] 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. [15] introduced a DeepRM framework to build autonomous and intelligent systems that learn to manage resources from their own experience directly.

Likewise, the authors in [19] proposed a DRL approach for a decentralized resource allocation mechanism for 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. [16] 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. [17] 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 [20] introduced an RL approach for learning the scheduling policy automatically and reduce the estimation error on data centers. Similarly, Ma et al. [18] 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 of a workflow under deadline constraints in the infrastructure as a service (IaaS) model.

2.3. Paper's Contributions

Unlike the approaches above, which are mainly based on meticulously designed heuristics that ignore the patterns of incoming tasks, the one that we introduce uses 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, compared with existing researches for the 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. Furthermore, the SDN capabilities

Article	Idea	Criteria			
		Energy effeciency	Scalability	Latency	Bandwidth
Lei et al. [12]	Automation and orchestration of IoT resources using RL & DRL in IoT Network	-	-	✓	✓
Wang et al. [13]	DRL-based planning for LTE cellular networks	-	✓	-	✓
Sen et al. [14]	Schedule application tasks in intelligent cognitive assistants distributed through machine learning	✓	✓	-	-
Hongzi et al. [15]	Introduced a DRL-based framework to automatically manage IOT resources from their own experience	-	✓	-	-
Dhoha et al. [16]	Improve resource sharing and task allocation using a cooperative process based on DRL	-	✓	-	-
Wang et al. [17]	Incremental approach based on DRL for task allocation strategies	-	✓	-	-
Ma et al. [18]	Task planning optimization scheme based on heuristic approaches	-	-	✓	-
H .Ye et al. [19]	Develop a decentralized resource allocation mechanism for V2V communications using multi-agent DRL	-	✓	✓	-
S. Liang et al. [20]	Introduce a mechanism to address the customized job scheduling problem in data centers using DRL	-	-	-	-
Our approach		✓	✓	✓	✓

Table 2: Reinforcing learning for Task Allocation

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 Task scheduler algorithm 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.

The SDN controller collaborates with our DRL algorithm to usher the intelligent network resource scheduling and management process. The SDN controller uses a planner algorithm to manage task processing requests and create the 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) trigger the flow scheduler 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 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. the latter helps to avoid any kind of noise, useless and less essential data just it will represent the relevant and helpful information when going to use it the second step to optimizing the problem-solving of scheduling tasks. 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 in OpenFlow rules for handling the tasks. Finally, the SDN controller assigns the fog nodes to decision for processing 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 T_1,T_2,...,T_N, which will be assigned to distinct fog nodesF_1,F_2,...,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

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

- Execution time to T_i task at fog node F_j

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

Where $D_i(t)$ standing for the data size for the task T_i, θ_i is the computing intensity and $C_j(t)$ is the computing resource on the fog node F_j.

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

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

Where $w_{ij}(t)$ is the bandwidth, $h_{ij}(t)$ is the channel power gain, $p_{ij}(t)$ is the transmission power, and σ is the noise power.

- The total time delay is given by equation 4:

$$TT_{ij}(t) = TTR_{ij}(t) + TTC_{ij}(t) + TTW_{ij}(t) \tag{4}$$

Where $TTW_{ij}(t)$ is the waiting time for task scheduling.

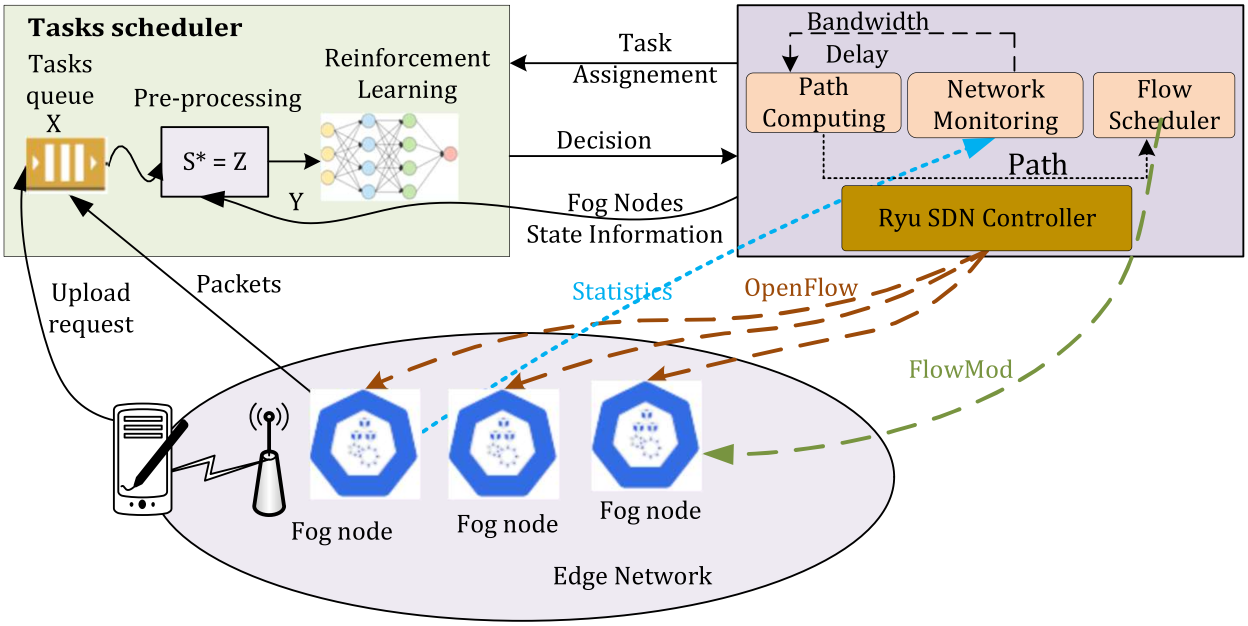

Figure 1: Task scheduling an architecture at a glance

- Similarly, the energy consumption is denoted by equation 5:

$$EC_{ij} = TTR_{ij}(t) \times p_{ir}(t) + TTC_{ij}(t) \times p_{ie}(t) \quad (5)$$

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, a compelling need is to design a hybrid heuristic algorithm to build a task scheduling strategy. In simplifying the complexity of the problem into a single problem, and reducing the difficulty of solving, we consider the hypotheses:

- Each task is independent, and there are no constraints between the tasks.
- Each task can be assigned to only a node fog.
- 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 6, where both time delay and energy consumption constraints are formulated as follows:

$$f = \min \sum_{i=1}^{n} (W_{it} \sum_{j=1}^{m} [X_{ij}(t) \times TT_{ij}(t)] + W_{ie} \sum_{j=1}^{m} [X_{ij}(t) \times EC_{ij}(t)]) \quad (6)$$

Where W_{it} is the weight of delay, and W_{ie} is the weight of energy consumption. Both weighting factors are set to 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-makers preference.

3.3. *Deep Reinforcement Learning for resolving Task Scheduling Problem*

Figure 2 glances 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, the SDN controller uses historical tasks to build the deployment decisions. In addition, 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 $Z = g(x)$ is used to filter 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 task T_i to node F_j and generate the optimal decision to schedule the tasks.

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
    $a = \arg\max(\text{act-values}[0])$ ;
    Execute $a$ // Send $t$ to $n$
end
```

Algorithm 1 illustrates the task assignment approach performed by 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 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 6), the algorithm the *argmax* operator to find the maximum values fulfilling the constraints of low-energy consumption and lower network latency.

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

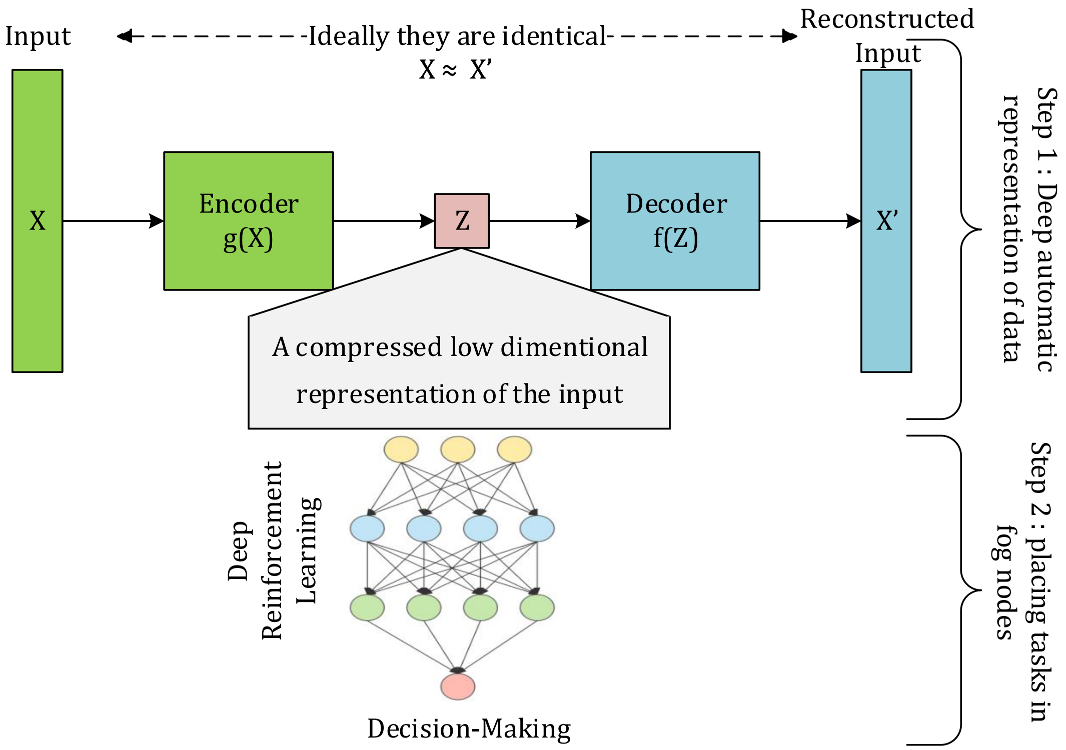
Input
-Ideally they are identical-
X ≈ X'
Reconstructed Input
X
Encoder
g(X)
Z
Decoder
f(Z)
X'
A compressed low dimentional representation of the input
Step 1 : Deep automatic representation of data
Deep Reinforcement Learning
Decision-Making
Step 2 : placing tasks in fog nodes

Figure 2: Deep Learning for Task Scheduling

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 the find optimal decision policy; or (ii) it can leverage the information already given by the Q values defined by equation 7:

$$\begin{aligned} Q(S^*) &= R + \gamma \max Q(s', a') \\ Action\ a &= \arg\max_{a'} Q(s, a') \end{aligned} \tag{7}$$

The total reward is given by equation 8:

$$R = \sum_{i=1}^{n} (W_{it} \sum_{j=1}^{m} \beta[X_{ij}(t) \times TT_{ij}(t)] + W_{ie} \sum_{j=1}^{m} \beta[X_{ij}(t) \times EC_{ij}(t)]) \tag{8}$$

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 be little of the total training data error. Worthy of mention, the deep learning neural network loss function, to predict the states of Q values, given by equation 9:

$$L = \frac{1}{2}[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]^2 \tag{9}$$

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: model trained to assign task to node
repeat
    Select an action a with probability $\epsilon$
    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
```

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 observed initial states of the SDN network. Then, nodes will be chosen with their smallest probability values from the filled matrix and randomly assign the tasks to its distinct fog nodes. Once the first step is completed, the controller can now 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 found to be 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.

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 $n_j$
$i \leftarrow 1$
$reward \leftarrow 0$
$totrewards \leftarrow 0$
while $i <= m$ do // m is the number of
 tasks
    $action \leftarrow env.action_{s}pace.sample()$
        // randomly choose an action

    $ob, reward \leftarrow env.step(action)$
    update($N$,$ob$) // update of nodes

    $totrewards+ = reward$
    $i + +$
end
```

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 gives each task to a randomly chosen fog node and updates the fog node energies each time. In addition, it also updates the accumulated rewards.

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.

Algorithm 5 illustrates the task assignment and

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

```
Algorithm 5: Asynchronous Advantage Actor-Critic Agent (A3C)
Output: Model trained with workers to assign task to node.
for $i \leftarrow 1$ to $n$ do // $n$ is the number of workers
    $Wi$.run() // Start worker thread
end
$step \leftarrow 1$ // ForEach worker $Wi$, initialize step counter
$T \leftarrow 0$ // Initialize episode counter
repeat
    $d\theta \leftarrow 0$; $d\theta_v \leftarrow 0$ // reset gradients
    $\theta' \leftarrow \theta$; $\theta'_v \leftarrow \theta_v$; $t \leftarrow step$ // Synchronize thread-specific parameters
    $s \leftarrow s_t$ // Initialize iteration
    // Get observation state
    while $s$ is not terminal and $step - t < t_{max}$ do
        Simulate action $a_t$ according to $\pi(a_t|s;\theta\ )$
        Receive reward $r_t$ and next state $s_{t+1}$
        $step++$;
        $T++$;
    end
    if $s_t$ is terminal state then
        $R \leftarrow 0$
    else $R \leftarrow V(s_t, \theta'_v)$ // Bootstrap from last state

    for $i \leftarrow step - 1$ to $t$ do
        $R \leftarrow r_i + \gamma R$
        $d\theta \leftarrow d\theta + \Delta_{\theta'} log(\pi(a_i|si;\theta')(R - V(s_i;\theta'_v)))$
        // Accumulate gradients
        $d\theta_v \leftarrow d\theta_v + \frac{\partial((R-V(s_i;\theta'_v))^2)}{\partial\theta'_v}$
    end
    // Perform asynchronous update of $\theta$ and $\theta_v$
    $\theta = \theta + d\theta$
    $\theta_v = \theta + d\theta_v$
until $T > T_{max}$
```

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. 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 full 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 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}.

4. 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.

4.1. Testbed Setup

We implemented our framework using an emulated SDN environment comprising Mininet [22] as our network emulators along with OpenFlow SDN switches for creating different IoT scenarios. Furthermore, we extended Mininet to support Open AI Gym toolkit [23] for reinforcement learning and deployed IoT nodes in the form of virtualized microservices using Docker containers in emulated Kubernetes clusters. We also implemented the SDN northbound application using Python-based Ryu [24] 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.

4.2. Pre-processing

The first step we performed on our data sets comprises pre-processing input data to carry out the training of our SDN controller (i.e., 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.

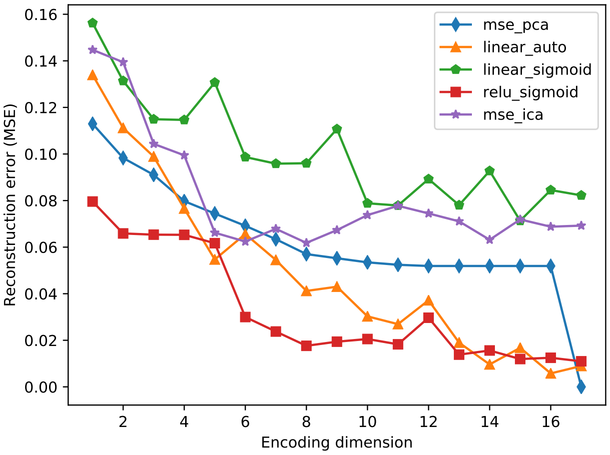

Figure 3: Data Pre-Processing

Figure 3 illustrates the Mean Square Error (MSE) regression loss function we got 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 (Relu-Sigmoid) performs better filtering and refinement results while keeping the MSE error minimum. Because the sigmoid function makes the loss function non-convex, then rather than creating a single global minimum for our training, we create multiple local minima to find optimal task assignment strategies.

4.3. 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 that he gets 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.

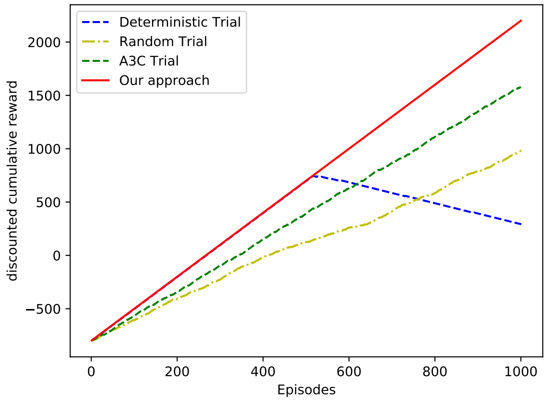

Figure 4: Cumulative Rewards for Our approach against the three other approaches.

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. His cumulative reward curve slowly increases for the random agent, which means some tasks have not been assigned, and we have to put them on hold state. For the A3C agent, its cumulative rewards were slowly increasing compared to our approach. Our approach performs better cumulative rewards than the deterministic and random case, which selects 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 each time the action is selected. It motivates it to maximize the rewarded return.

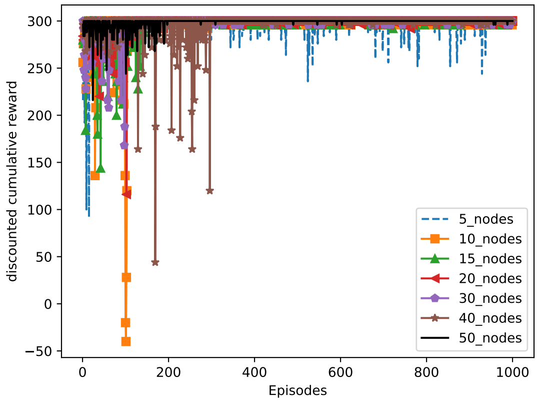

Figure 5: Local Rewards with Increasing number of nodes

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. 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. We based claim that our deep learning approach is gainful to implement both local, optimal and global tasks assignments and to schedule for SDN-enabled IoT networks while fulfilling the respect of QoS constraints.

4.4. Energy-Efficiency

We aim to reduce the energy consumption on running fog-enabled IoT nodes as we described in equation 5 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 $5,000wh$.

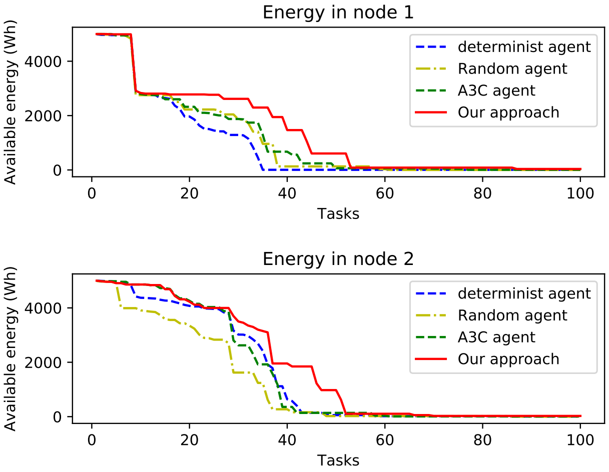

Figure 6: Energy Consumption in both tasks' executing fog Nodes

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 major objective is to minimize the overall energy consumption of our SDN-enabled fog network, as we described in equation 5. 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%.

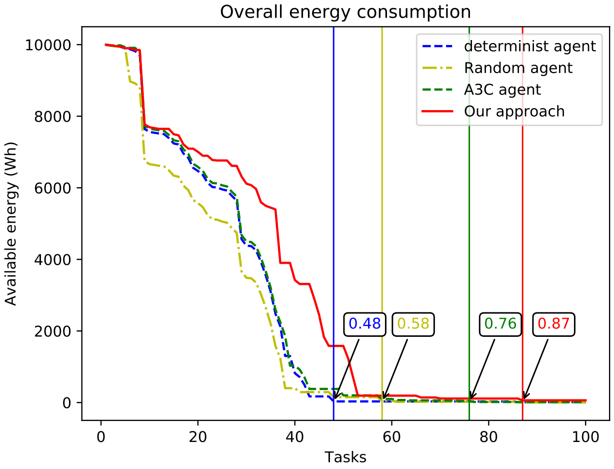

Figure 7: Energy Saving and Efficiency for all available Fog Nodes

Our results confirm 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.

4.5. Assessing the Latency performance

As our optimization approach aims at minimizing the network latency for available nodes during the task executions, as we described in equation 4. We gauged the total time delay expected by available fog-enabled nodes to processing the current task's requests and communicate the results to remote IoT senders.

To handle the pending task requests, we measured the total delay expected by the available fog nodes. 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, is shown in Figure 8.

The latency results of the different approaches have been summarized in Table 3. As expected, the deter-

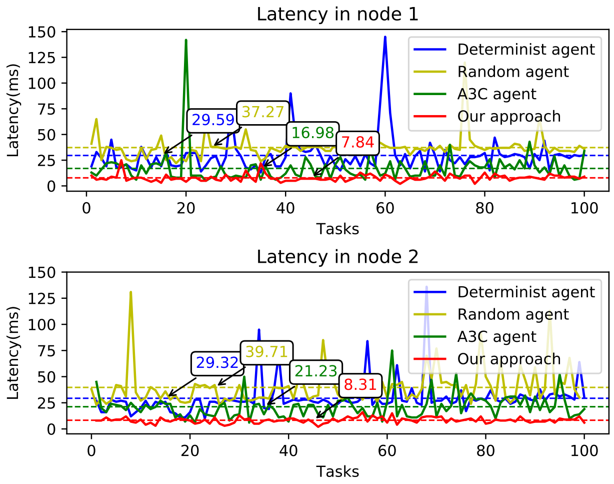

Figure 8: Evaluating Latency during Task Scheduling

ministic 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 ensures 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

4.6. 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 3.3. 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 surpassed 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, our IoT network means the deterministic algorithm will stick to the current task assignment state if 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 the better local optimum solution. Similarly, the randomized exploration approach (randomized policy) will explore different states based on a particular probability distribution, making it slightly slower.

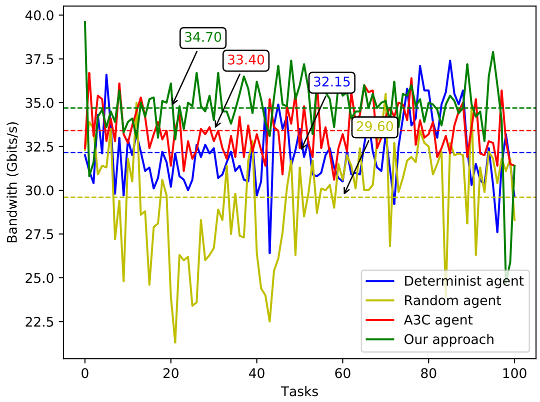

Figure 9: Evaluating The bandwidth utilization during task Scheduling

Additionally, while the A3C approach uses several parallel workers (agents) that interact asynchronously with separate instantiations of the environment;,its bandwidth usage is close to *33.4* Gbits/s; it is however defeated by our approach which reaches *34.70* Gbits/s. The A3C technique tend 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.

5. 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, ensured 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 focus on developing 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 central model parameters and distributing them back to all edge nodes.

Acknowledgments

The Tunisian Ministry of Higher Education partially funded this work 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] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

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

[24] R. project team, RYU SDN FRAMEWORK, FIXME, 2014.