Deep Reinforcement Learning for Energy-Efficient Task Scheduling in SDN-based IoT Network

Bassem Sellami*, Akram Hakiri†, Sadok Ben Yahia*, and Pascal Berthou‡
†University of Carthage, SYSCOM ENIT, ISSAT Mateur, Tunisia.
*University of Tunis El Manar, Faculty of Sciences, Dept of Computer Sciences.
‡CNRS, LAAS, UPS, 7 Avenue du colonel Roche, F-31400 Toulouse, France.
Email: sellami.bassem@gmail.com, akram.hakiri@enit.utm.tn
sadok.benyahia@fst.rnu.tn, pascal.berthou@laas.fr

***Abstract*—The growing demand and the diverse traffic patterns coming from various heterogeneous Internet of Things (IoT) systems place an increasing strain on the IoT infrastructure at edge network. Different edge resources (e.g. servers, routers, controllers, gateways) may illustrate different execution times and energy consumption for the same task. They should be capable of achieving high levels of performance to cope with the variability of tasks handling. However, edge nodes are often faced with issues to perform optimal resource distribution and energy-awareness policies in a way that makes effective run-time trade-offs to balance response time constraints, model fidelity, inference accuracy and task schedulability. To address these challenging issues, in this paper we present a dynamic task scheduling and resource management deep reinforcement learning approach for IoT traffic scheduling in SDN-based edge networks. First, we introduce the architectural design of our solution, with the specific objective of achieving high network performance. We formulate a task assignment and scheduling problem that strives to minimize the network latency, while ensuring energy efficiency. The evaluation of our approach offers better results compared against both deterministic and random task scheduling approaches, and show significant performances in terms of latency and energy consumption.**

***Index Terms*—Task scheduling; SDN; Fog Computing; Deep Reinforcing Learning; Internet of Things.**

I. Introduction

The Internet of Things (IoT) is increasingly connecting huge number of smart objects [1], which generate, gather, process, infer, and transmit massive amount of sensory data that should be processed at edge network 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 be able to coordinate their processing with neighboring helper IoT nodes by offloading their tasks in order to reduce tasks execution delay significantly. Specifically, such Fog nodes should be able to support the burst and unpredictable IoT traffic at different time scales, required 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, which makes it difficult to meet the increasingly growing demand and dynamic needs of IoT applications, and address the heterogeneous requirements smart objects that communicate over the Internet. Therefore, flexible resource management, intelligent network control, and efficient task scheduling algorithms play major roles to ensure fair and guaranteed performance.

Software Defined Network (SDN) is used to enable flexible and collaborative task offloading service orchestration in cloud-mobile edge computing (MEC) [3]. A service orchestration scheme is proposed to reduce network load along with a differentiated cloud-edge offloading decision algorithm have been proposed to improve cloud computation and energy consumption. Similarly, a SDN scheme for balancing Edge-Cloud traffic load and improving service response time has been introduced in [4]. A nature-inspired meta-heuristic schedulers [5] based on ant colony optimization has been introduced to effectively load balance IoT tasks between Fog nodes. Likewise, Cai et al. [6] proposed a framework for tasks and energy offloading in a fog-enabled IoT network, while minimizing tasks execution delay. Recently, Machine learning techniques have become a promising to bring intelligence to the SDN controller by performing data analysis, network optimization, and automated provision of network services [7].

However, these network provisioning approaches neither address the dynamicity of IoT applications nor care about resource utilization of fog-enabled IoT nodes. To address this issue, the network must be flexible enough to be reprogrammed in accordance with any change in IoT application needs. An additional trend reveals that fog computing devices should provide i) an on-demand basis resource allocation to support adaptive horizontal and vertical scaling of the network resources; ii) flexible infrastructure virtualization that exploits in-network programmability capabilities to operate inside a SDN-enabled virtualization platform and; iii) 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 [8].

To address those issues, this paper introduces a Deep Reinforcement Learning (DRL) energy-efficient task assignment and scheduling in SDN-based Fog IoT Network. 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 propose deep reinforcement learning algorithm to address task allocation and resource planning problem in dynamic and distributed IoT environment to improve latency minimization and reduce energy consumption. Our Deep RL uses intelligent agents that learn to make better decisions directly from experience interacting with the environment.

The remainder of this paper is organized as fellows: Section II compares our work with related research. In Section III, we depict the details of our architecture, and we describe the analytical model for our DL-based dynamic task scheduling and resource management. Section IV evaluates our solution to validate our claims of flexible data delivery and low latency communication overhead; and finally Section V presents concluding remarks alluding to lessons learned and future work.

II. Related Work

This section draws on the research directions on task offloading and resource allocation problem using Reinforcement Learning and empowering cognitive and autonomic control and management IoT services in a fog-enabled network with SDN.

A. Reinforcement Learning for Task Allocation

Task scheduling problem in dynamic IoT environment is often one of the most challenging resource management problems as it often manifests as a difficult online decision-making where appropriate solutions usually depend on the dynamic workload and the interaction with the surrounding environment [9]. Lei et al [10] provides a comprehensive survey of automating and orchestrating IoT resources using reinforcement learning (RL) in order to achieve autonomy. Wan et al. [11] introduced a DRL-based scheduling for Cellular Networks. They proposed two methods, i.e., learning from a dual AI 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 uses Proportional Fair (PF) scheduling algorithms as method, the PF algorithm is employed as expert knowledge to help with DRL agent training. Sen et al. [12] 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 problem between the three tiers in the edge computing system (i.e., remote cloud, fog and edge devices). Hongzi et al. [13] introduced DeepRM framework to build autonomous and intelligent systems that learn to manage resources directly from their own experience.

Likewise, the authors in [14] proposed a DRL approach for 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 level for transmitting V2V data. Dhoha et al. [15] proposed a cooperative DRL-based task allocation process that combines learning agents capabilities to improve resource sharing and distributed task allocation. Wang et al. [16] 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 [17] introduced a reinforcement learning approach for learning the scheduling policy automatically and reduces 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.

B. Task Scheduling for SDN-enabled Edge Computing

Additionally, SDN has been widely used to empower dynamic and effective resource allocation in diverse cloud [19] and data centers [20] networks, and providing on-demand application and resource management in wireless sensor networks [21] and edge network [22]. For example, Wu et al. [23] introduced UbiFlow framework that combines ubiquitous flow control and mobility management in urban heterogeneous networks. UbiFlow adopts distributed SDN controllers pattern to divide traffic scale among geographically distributed IoT network islands or partitions, where each controller can maintain network scalability, load balancing and consistency. Chen et al. [24] proposed a SDN-based heuristic model for offloading distributed computing resource in ultra-dense network. They formulated the task offloading problem as a mixed integer non-linear program to solve task placement and resource allocation problem in mobile edge computing. Similarly, Pen et al. [25] introduced a mobile task offloading framework for device-to-device (D2D) Fogging. They leverage Lyapunov optimization D2D Fogging methods for achieving energy efficient task executions for network wide users and reduce time-average task execution in order to avoid over-exploiting and free-riding behaviors.

Furthermore, Kuang et al. [26] investigated a joint problem of partial offloading scheduling and resource allocation for Mobile edge computing (MEC) ti run multiple independent tasks. They formulated their framework as non-convex mixed-integer optimization problem based on Lagrangian dual decomposition in order to minimize the weighted sum of the execution delay and energy consumption while guaranteeing the transmission power constraint of the tasks. Zhang et al. [27] proposed a fair and energy-minimized task offloading algorithm based on a fairness scheduling metric. Their scheme considers 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. [28] proposed a Latency Aware Task Assignment (LATA) scheme for multi-cloudlet network to optimize the latency monetary cost in computing the tasks of mobile devices by making optimal task assignment among the micro-clouds. LATA model proposes an admission control policy to maintain optimality in high traffic conditions. Besides, the authors in [29] 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.

C. Paper Contribution

Unlike the aforementioned approaches, which are mostly based on meticulously designed heuristics which ignore the patterns of incoming tasks, our approach used SDN to enhance the control and management of For-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 supporting future real-time and time-sensitive Fog-enabled IoT network design, by combining smart connectivity, reinforcement learning, distributed SDN communication, automated and greatly cost reduced network operation.

Furthermore, compared with existing researches for the energy consumption in fog-enabled networks, which mostly 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 especially in the scenarios with battery-powered distributed IoT nodes, offering predictive behaviors on the network, and preventing the impact of failures. The SDN capabilities offered 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 programmable analytics to the application layer through open interfaces, instantiate service intelligence at the edge network.

III. Model for Task Assignment and Scheduling Problem

This section delves into the architectural details that enable 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 edge network .

A. System Architecture

Figure 1 illustrates the architectural design of our deep reinforcement assignment and scheduling solution to address the task scheduling problem in 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 coming from mobile IoT applications, and learning-based input representative and a planning decision maker based on learning. First, the SDN controller uses a planner algorithm to manage

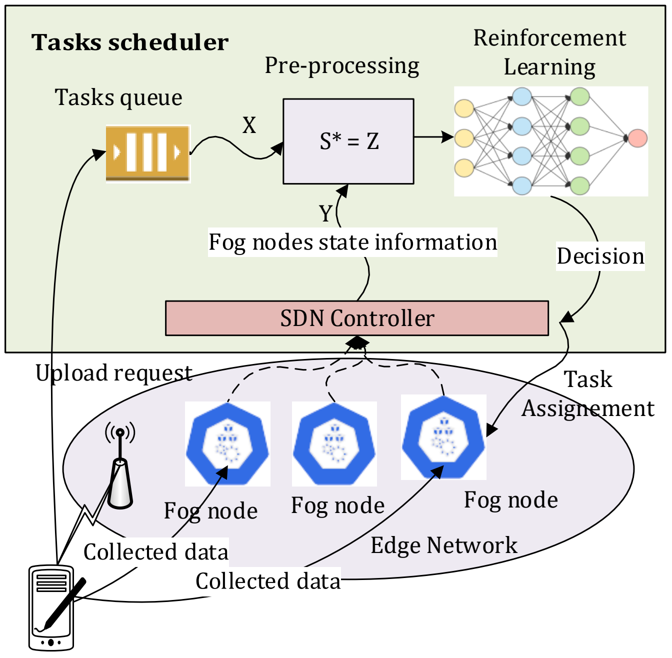

Fig. 1: Task Scheduling architecture

task processing requests and create historical data-set from incoming task requests. The controller learns a data-set to represent the state information of all the fog nodes and the task requests in order to create a latent representation model to avoid any kind of noise, useless and less important data. Once the information available on the dataset are well filtered and represented in form of network graph, the SDN controller starts the learning process to generate learning policies to input programming decision values (Q-value). Then, it selects the best fog node and sends the decision in form of OpenFlow rules for handling the tasks. Finally, the SDN controller assigns the fog nodes to decision for processing the tasks requests. Thereafter, the mobile device can download data from the assigned fog node.

B. 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 the different fog nodes F_1,F_2,...,F_m in order to minimize energy consumption and time delay by making the most of the use of transmission channel. Lets consider:

- X_{ij}(t): denotes the assignment of task T_i on fog node F_j

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

- Execution time to T_i task at fog node F_j

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

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

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

$$TTR_{ij}(t) = \frac{D_i(t)}{r_{ij}(t)} \quad ; \qquad r_{ij}(t) = w(t) * \log(1 + \frac{h(t) * p(t)}{\sigma}) \tag{3}$$

Where $w_(t)$ is the bandwidth, $h(t)$ is the channel power gain, $p(t)$ is the transmission power, and σ is the noise power.

- The total time delay is denoted by equation 4:

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

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

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

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

We model the tasks 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 becomes difficult to find an optimal solution using polynomial method, hence the need to design a hybrid heuristic algorithm is proposed in this section to build a task scheduling strategy. In order to simplify the complexity of the problem into a single objective problem and reduce the difficulty of solving, we consider the following hypotheses:

- Each task is independent and there are no constraints between the tasks.
- Each task can only be assigned to a node fog.
- No task can be allocated repeatedly.
- The task in the calculation process 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 is formulated as follows:

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

Where W_{it} is the weight of delay and W_{ie} is the weight of energy consumption.

C. Deep Reinforcement Learning for resolving Task Scheduling Problem

Figure 2 depicts our approach to resolve the task scheduling problem using deep reinforcement learning. Since 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. The DRL algorithms we implemented inside the controller can analyze the performance of all connected fog-enabled IoT nodes in order to build an efficient scheduling to execute several simultaneous tasks and predict optimal scheduling on the network that meets both the low-latency and efficient-energy requirements.

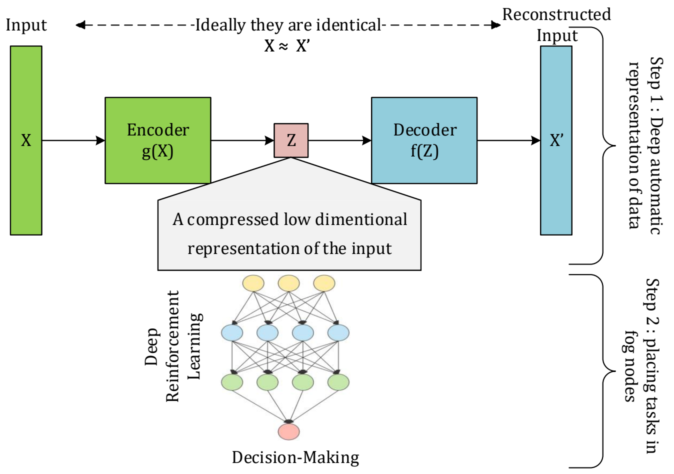
Input
Ideally they are identical
X ≈ X'
Reconstructed Input
X
Encoder g(X)
Z
Decoder f(Z)
X'
A compressed low dimentional representation of the input
Step 1 : Deep automatic representation of data
Deep Reinforcement Learning
Step 2 : placing tasks in fog nodes
Decision-Making

Fig. 2: Deep Learning for Task Scheduling

First, we should train the SDN controller to represent the data-sets of all tasks and fog-enabled nodes intelligently in order to perform tasks assignments using the best optimal way under the aforementioned constraints of minimizing the network latency and reduce the energy consumption. Therefore, we introduced the DRL algorithm to choice and apply the best decision that places the tasks on available fog nodes. As shown in Figure 2, the A compressed low-dimensional representation of the input are used to the find a latent representation of the data between tasks that will be executed and fog node states ready to execute these tasks. Then, the encoder later, which model as a function $g(x) = sg(Wx + b)$, is used to reduce the input X in a representation of latent space Z. Thereafter, a bottleneck layer $Z = g(x)$ is used to filter the incoming data from the encoder layer. Then, a decoder function $f(x) = sg(W'z + b')$ is used to reconstruct X (input) from Z (representation of latent space). The latent representation Z obtained at the output of the encoder, i.e. $S^* = Z = g(S)$, is then 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 illustrates the task assignment approach performed by the SDN controller, which collects information from the underlying SDN routers about the available fog nodes capacities, including their available energy. Then, the algorithm receives a list of tasks along with their characteristics and assign them to available fog nodes. The DRL algorithms selects fog nodes based on their available energy and their current occupation rates in order to reduce the processing time delay. Once the controller assigned tasks to their relevant nodes, it keeps an historical dataset of the current node's processing and available energy. Each time a tasks is successfully assigned to a fog node, the the controller increase the value of local reward and select the next action according to the expected reward. Then, to maximize the objective function (see equation 6), the algorithm apply *argmax* operation to find the maximum values satisfying the constraints of low-energy consumption and lower network latency.

```
Algorithm 1: Tasks Assignment to Fog Nodes
Input: Detection nodes $N = \{n_1, n_2, \ldots, n_j\}$ with their
       available energies, Set of tasks
       $T = \{t_1, t_2, \ldots, t_i\}$ with their characteristics
Output: Assign task $t_i$ to node $n_j$
1 while 1 do // infinite loop
      // learn according to cases
2     Replay (n, t)
      // Predict the value of the reward
3     act-values = predict (n, t)
      // Choose the action according to
         the expected reward
4     $a$ = argmax(act-values[0])
5     Send $t$ to $n$
6 end
```

As described in algorithm 2, the SDN controller implements a Deep Q-learning algorithm) in form of a learning agent that maps states of the environment to actions the agent can take to move from one state to another, in order to maximize a numerical reward over time. Specifically, 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: either the controller can continue exploring the state space to find optimal decision policy, or its can exploit the information already present in 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:

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

Where W_{it} is delay weight and W_{ie} is weight of energy consumption.

We implemented the training algorithm that uses a regression loss function to minimize the total training data error. The deep learning neural network loss function to predict the states of Q values is defined by equation 9:

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

The algorithm 2 learns allocation policy to provide an optimal decision with respect to both the aforementioned constraints (i.e. in terms of latency and energy) and performance of the system. First, in order 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

```
Algorithm 2: DEEP Q-LEARNING ALGORITHM WITH EXPERIENCE REPLAY
Input: Initialize replay memory M ; Initialize
       action-value $Q$ with random weights ; Observe
       initial state $s$
Output: model trained to assign task to node
1  repeat
2      select an action a with probability $\varepsilon$
3      select $a$ random action otherwise select $a$ =
         $\arg\max_{a'} Q(s, a')$
4      execute action $a$
5      observe reward $r$ and new state $s'$
6      store experience $< s, a, r, s' >$ in memory M
7      sample random transitions $< ss, aa, rr, ss' >$ from
         memory $M$
8      calculate target for each minibatch transition
9      if $ss'$ is terminal state then
10         $tt \leftarrow rr$
11     else $tt \leftarrow rr + \gamma \max'_a Q(ss', aa')$
12
13     train the Q network using $(tt - Q(ss, aa))^2$ as loss
14     $s = s'$
15 until terminated
```

network. Then, assign tasks to random nodes (i.e. fills matrix with calculated policies in real-time) chosen with their smaller probability. Once the first step is completed, the controller can now move to new next states i.e. returning a reward and continue performing the calculation and transitioning from one state to another. Each newly calculated step is saved in the matrix, and every the existing policy is compared against the previous one, if the newer policy is better than it will be considered as optimal (i.e. local optimization) and so on. The operation is performed repeatedly until a global optimal assignment is obtained for each task. Then, the operation is repeated until all tasks in the waiting queue are satisfied and assigned to the best available fog nodes.

IV. Performance Analysis

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

A. Testbed Setup

We implemented our framework using an emulated SDN environment comprising Containernet [30] and Mininet [31] as our network emulators with OpenFlow virtual switches used for creating different IoT scenarios where Docker containers are used as IoT nodes in emulated Kubernetes clusters. We also implemented the framework using Python-based Ryu [32] SDN controller to help manage traffic flows. For the blockchain development, we developed our solution using TensorFlow python interface for interacting with our SDN environment, which we used to run tests more than 100 nodes, each running over 1000 tasks simultaneously. We evaluated our solution against deterministic tasks placement and random tasks placements approaches. For the former, a deterministic agent plans tasks according to their order of arrival and assigns them to their nearly nodes in respects to their minimum latency. The random agent assigns tasks to available nodes in a stochastic order, i.e. it assigns tasks to available node with the non-strategy. That is, if a selected node doesn't have the capacity to execute an incoming task, the random agent leave it in hold state in the waiting queue and assigns tasks to other nodes.

B. Pre-processing

The first step we performed on our data-sets consists on pre-processing input data in order to realize the apprenticeship of our SDN controller (Ryu). Carrying out data refinement allows properly representing and preparing data for our deep Q-learning model to perform tasks assignments and scheduling. Therefore, we implemented different techniques to reducing the dimension of our datasets to find the best representation of our data.

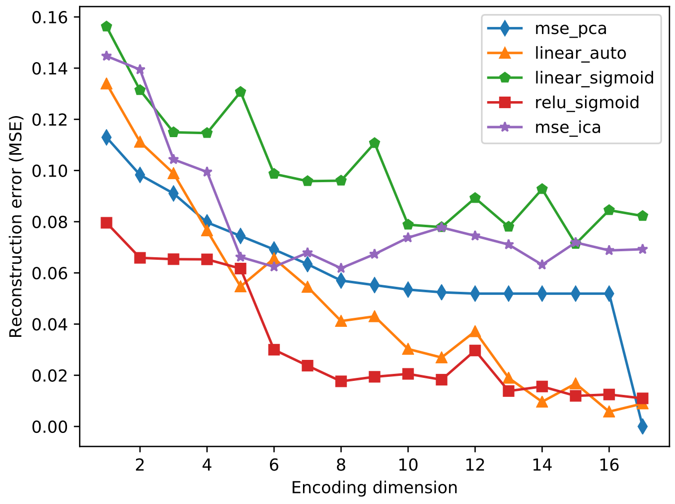

Fig. 3: Data Pre-Processing

Figure 3 illustrates the Mean Square Error (MSE) regression loss function we obtained for different refinement techniques, including Principal Component Analysis (PCA), Independent Component Analysis (ICA), Sigmoid function, and reLU piecewise function. As expected from Figure 3, the Sigmoid function perform better filtering and refinement results while keeping the MSE error minimum. Because the sigmoid function makes the loss function non-convex, rather than creating a single global minimum for our training, we creates multiple local minima to find optimal tasks assignment strategies.

C. Cumulative reward

The SDN agent performs deep learning in run-time, collects states from the environment and send 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 better candidate to place the current tasks requests.

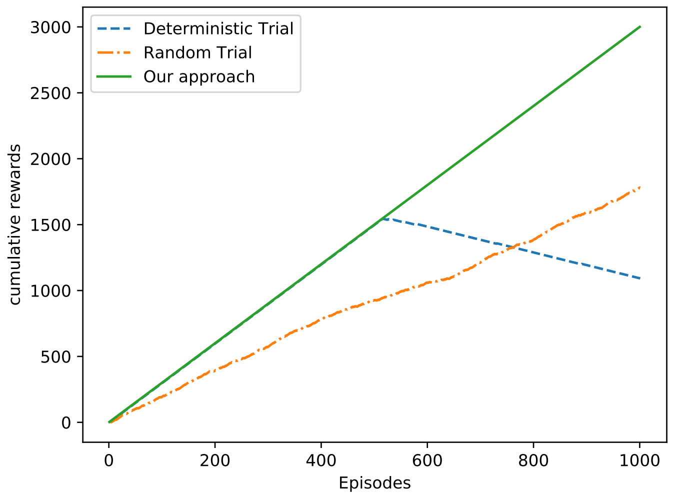

Fig. 4: Cumulative Rewards for three approaches.

Figure 4 compares the cumulative reward obtained by our approach against the deterministic and random approaches. After the deterministic agent increases to almost 550 accumulated rewards, it decreases and starts losing rewards. It starts losing its computation power and its ability to complete the planning of newer coming tasks. For the random agent, his cumulative reward curve slowly increases, which means there are tasks that have not been assigned, and they are put on hold state. Our approach performs better cumulative rewards compared to both the deterministic and random case, which selects the available node based on the energy level. The cumulative reward curve increases very quickly compared to other agents, which means that the agent has a very optimal placement strategy. To evaluate the stability and the scalability

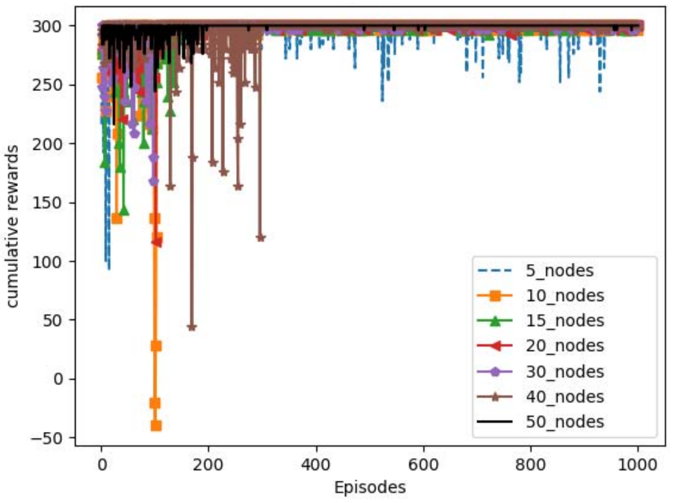

Fig. 5: Local Rewards with Increasing number of nodes

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 can quickly learn from the SDN topology network to make optimal decisions. The local reward, i.e. local optimal assignment, rapidly becomes close to 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 our approach with over 100 in other scenarios (not shown in Figure 5) and we observed the same behavior. That is, we argue that our deep learning approach successfully can help to implement both local optimal and global optimal tasks assignments and scheduling for SDN-enabled IoT network, while ensuring the respect of QoS constraints.

D. Energy-Efficiency

Our objective is 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 are batteries-powered. In order to evaluate the energy efficiency of our SDN-enabled solution, the SDN controller trained the agent by 1000 episodes during which the agent should be able to plan 100 tasks to energy-constrained fog nodes. That is, each fog node has a limited power capacity, i.e. their battery level during this planning process is close to 5000wh. Figure 6

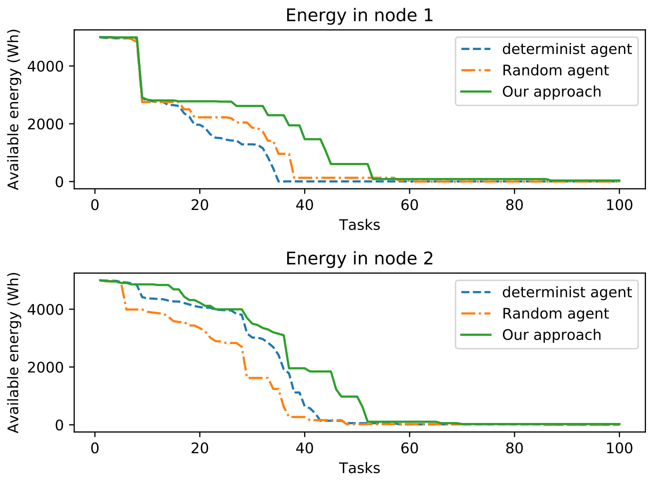

Fig. 6: Energy Consumption in two tasks' executing Fog Nodes

illustrates the energy consumption of a two available Fog nodes, each running up to 100 tasks simultaneously, using our training approach against both deterministic and random training agents. It is clear that throughout the planning strategy of these tasks, the DRL agent in our approach keeps better battery level in both nodes compared to deterministic and random agents.

Recall that our main 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 performs better energy-efficiency, i.e. up to 87% compared against both deterministic agent, which performs 48% of energy efficiency, and random agent which performs 58% of energy efficiency, too.

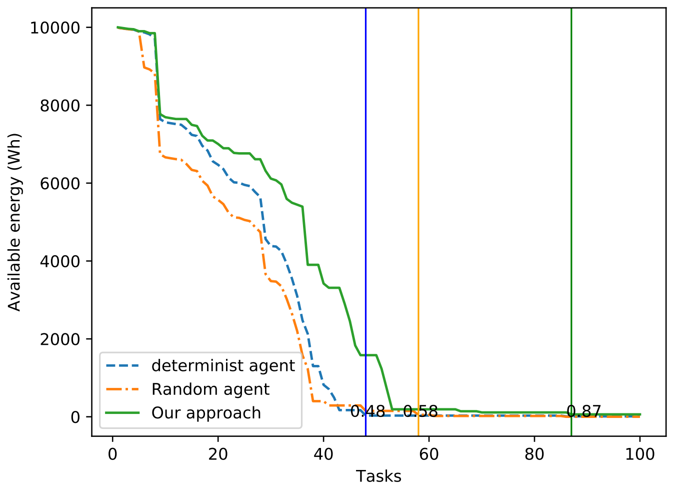

Fig. 7: Energy Efficiency for all available Fog Nodes

E. Evaluating the Latency

As our optimization approach aims at minimizing the network latency for available nodes during the tasks executions as we described in equation 4. We measured the total time delay expected by available Fog-enabled nodes to processing the current tasks requests and communicate the results to remote IoT senders. Figure 8 compares the time delay,

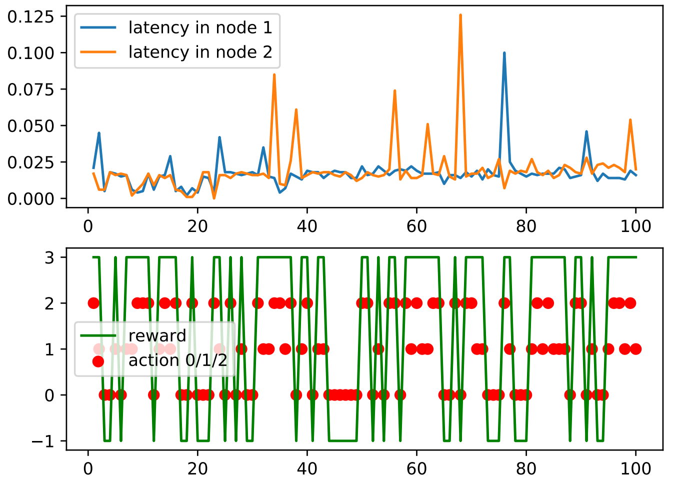

Fig. 8

i.e. latency, of our approach in two batteries-powered nodes and shows the instantaneous reward obtained by our task scheduling algorithm during the SDN controller agent training. During the planning of tasks by the SDN-enabled agent we implemented, the latency in all fog nodes remains lower when they have been selected (action) by our agent. We repeated the experiments five times, and we find the latency is close to 12.5 milliseconds. Therefore, our agent ensures to keep a minimum latency of all the fog nodes of our network.

V. Conclusion

In this paper we presented a Deep Reinforcement Learning (DRL) approach to create an intelligent SDN-enabled Fog IoT network. By blending DRL and SDN, our solution allows training DRL agents to select optimal allocation decision policies for tasks assignments and scheduling in real-time. The performance evaluation shows the effectiveness of the proposed solution to perform both local and global optimization, ensure lower-latency communication, and increase energy efficiency.

Our future work will focus on developing a Federated Machine Learning (FedML) approach to solve the issue of data ownership and privacy, by training statistical security models inside Fog nodes, while keeping data samples localized inside Fog nodes. It becomes increasingly appealing to make IoT devices generating and keeping their data locally and push the network computation to edge. This promising undertaking will greatly 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

This work was funded by the Tunisian Ministry of Higher Education and Scientific Research (MES) under the Young Researchers Incentive Program (19PEGC09-04). 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.

References

[1] F. Shan, J. Luo, J. Jin, and W. Wu, "Offloading delay constrained transparent computing tasks with energy-efficient transmission power scheduling in wireless iot environment," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4411–4422, 2019.

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

[3] M. Huang, W. Liu, T. Wang, A. Liu, and S. Zhang, "A cloud–mec collaborative task offloading scheme with service orchestration," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5792–5805, 2020.

[4] Y. Liu, Z. Zeng, X. Liu, X. Zhu, and 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*, vol. 7, no. 7, pp. 5922–5933, 2020.

[5] M. K. Hussein and M. H. Mousa, "Efficient task offloading for iot-based applications in fog computing using ant colony optimization," *IEEE Access*, vol. 8, pp. 37 191–37 201, 2020.

[6] P. Cai, F. Yang, J. Wang, X. Wu, Y. Yang, and X. Luo, "Jote: Joint offloading of tasks and energy in fog-enabled iot networks," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3067–3082, 2020.

[7] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 393–430, 2019.

[8] Q. D. La, M. V. Ngo, T. Q. Dinh, T. Q. Quek, and H. Shin, "Enabling intelligence in fog computing to achieve energy and latency reduction," *Digital Communications and Networks*, vol. 5, no. 1, pp. 3 – 9, 2019.

[9] A. Haj-Ali, N. K. Ahmed, T. Willke, J. Gonzalez, K. Asanovic, and I. Stoica, "A view on deep reinforcement learning in system optimization," 2019.

[10] L. Lei, Y. Tan, K. Zheng, S. Liu, K. Zhang, and X. Shen, "Deep reinforcement learning for autonomous internet of things: Model, applications and challenges," *IEEE Communications Surveys Tutorials*, vol. 22, no. 3, pp. 1722–1760, 2020.

[11] J. Wang, C. Xu, Y. Huangfu, R. Li, Y. Ge, and 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.

[12] T. Sen and 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.

[13] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," 2016, p. 50–56.

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

[15] D. B. Noureddine., A. Gharbi., and 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.

[16] J. Wang, J. Cao, S. Wang, Z. Yao, and W. Li, "Irda: Incremental reinforcement learning for dynamic resource allocation," *IEEE Transactions on Big Data*, no. 01, pp. 1–1, apr 2020.

[17] S. Liang, Z. Yang, F. Jin, and 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*, vol. 12085, pp. 906–917, Apr 2020.

[18] X. Ma, H. Gao, H. Xu, and 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*, vol. 2019, no. 1, p. 249, Nov 2019.

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

[20] L. Yang, X. Liu, J. Cao, and Z. Wang, "Joint scheduling of tasks and network flows in big data clusters," *IEEE Access*, vol. 6, pp. 66 600–66 611, 2018.

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

[22] C. Lin, G. Han, X. Qi, M. Guizani, and L. Shu, "A distributed mobile fog computing scheme for mobile delay-sensitive applications in sdn-enabled vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5481–5493, 2020.

[23] D. Wu, X. Nie, E. Asmare, D. I. Arkhipov, Z. Qin, R. Li, J. A. McCann, and K. Li, "Towards distributed sdn: Mobility management and flow scheduling in software defined urban iot," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 6, pp. 1400–1418, 2020.

[24] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.

[25] L. Pu, X. Chen, J. Xu, and 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*, vol. 34, no. 12, pp. 3887–3901, 2016.

[26] Z. Kuang, L. Li, J. Gao, L. Zhao, and A. Liu, "Partial offloading scheduling and power allocation for mobile edge computing systems," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6774–6785, 2019.

[27] G. Zhang, F. Shen, Z. Liu, Y. Yang, K. Wang, and M. Zhou, "Femto: Fair and energy-minimized task offloading for fog-enabled iot networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4388–4400, 2019.

[28] S. C. G., V. Chamola, C.-K. Tham, G. S., and N. Ansari, "An optimal delay aware task assignment scheme for wireless sdn networked edge cloudlets," *Future Generation Computer Systems*, vol. 102, pp. 862–875, Jan 2020.

[29] S. Misra and N. Saha, "Detour: Dynamic task offloading in software-defined fog for iot applications," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1159–1166, 2019.

[30] M. Peuster, J. Kampmeyer, and H. Karl, "Containernet 2.0: A rapid prototyping platform for hybrid service function chains," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, 2018, pp. 335–337.

[31] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," 2010.

[32] R. project team, *RYU SDN FRAMEWORK*, February 2014.