Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2024.045912

ARTICLE

Task Offloading in Edge Computing Using GNNs and DQN

Asier Garmendia-Orbegozo[1], Jose David Nunez-Gonzalez[1,*] and Miguel Angel Anton[2]

[1]Department of Applied Mathematics, University of the Basque Country UPV/EHU, Eibar, 20600, Spain

[2]TECNALIA, Basque Research and Technology Alliance (BRTA), San Sebastian, 20009, Spain

*Corresponding Author: Jose David Nunez-Gonzalez. Email: josedavid.nunez@ehu.eus

Received: 11 September 2023 Accepted: 04 December 2023 Published: 11 March 2024

ABSTRACT

In a network environment composed of different types of computing centers that can be divided into different layers (clod, edge layer, and others), the interconnection between them offers the possibility of peer-to-peer task offloading. For many resource-constrained devices, the computation of many types of tasks is not feasible because they cannot support such computations as they do not have enough available memory and processing capacity. In this scenario, it is worth considering transferring these tasks to resource-rich platforms, such as Edge Data Centers or remote cloud servers. For different reasons, it is more exciting and appropriate to download various tasks to specific download destinations depending on the properties and state of the environment and the nature of the functions. At the same time, establishing an optimal offloading policy, which ensures that all tasks are executed within the required latency and avoids excessive workload on specific computing centers is not easy. This study presents two alternatives to solve the offloading decision paradigm by introducing two well-known algorithms, Graph Neural Networks (GNN) and Deep Q-Network (DQN). It applies the alternatives on a well-known Edge Computing simulator called PureEdgeSim and compares them with the two default methods, Trade-Off and Round Robin. Experiments showed that variants offer a slight improvement in task success rate and workload distribution. In terms of energy efficiency, they provided similar results. Finally, the success rates of different computing centers are tested, and the lack of capacity of remote cloud servers to respond to applications in real-time is demonstrated. These novel ways of finding a download strategy in a local networking environment are unique as they emulate the state and structure of the environment innovatively, considering the quality of its connections and constant updates. The download score defined in this research is a crucial feature for determining the quality of a download path in the GNN training process and has not previously been proposed. Simultaneously, the suitability of Reinforcement Learning (RL) techniques is demonstrated due to the dynamism of the network environment, considering all the key factors that affect the decision to offload a given task, including the actual state of all devices.

KEYWORDS

Edge computing; edge offloading; fog computing; task offloading

1 Introduction

Various computing centers can be found in a local networking environment, with possible interconnections. In the Edge Computing paradigm, this interconnection facilitates the transmission

of information is of particular interest. In many cases, when resource-constrained devices are allocated to solve computationally expensive tasks, they can become overloaded and not powerful enough regarding processability and memory availability. In this way, weaker computers can alleviate their computational load by assigning different tasks to more powerful devices nearby. These devices can vary depending on their complexity and proximity to these end-user devices. This variation of possible destinations is appropriate to distinguish different layers in an architecture, dividing it into the cloud, fog/edge, and IoT (Internet of Things) layers. The cloud tier comprises remote network servers rich in general resources with the capacity to store, manage, and process data. This general computer is the richest regarding processability and resource availability and often acts as a network orchestrator. In contrast, at the lowest level, it can find sensors, gadgets, and other IoT devices equipped with restricted computing capabilities but offer immediate responses to users. Its function is to collect information from the environment and act on environmental changes, among others. Meanwhile, other layers can be defined, such as the Fog and Edge layers, with greater capacities than the previous ones but with less processing and memory capacity than the cloud, being a valuable alternative for different types of computations.

Management and decision-making tools based on Artificial Intelligence (AI) algorithms have great potential to offer new and more efficient services that improve people's living conditions. These services could be of various types, from the classification of multiple classes of land cover [1] to systems where pollution forecasts are made [2]. These tools are possible thanks to the collection of information from the physical environment in real-time (RT) and the subsequent use of this data in complex Machine Learning (ML) and Deep Learning (DL) models. The models require high computability for the training phase and a large amount of available memory to store their parameters for the last inference. Consequently, IoT devices cannot store such an amount of data or train deep models, so they need to adjust the data and model sizes or send these mappings to more powerful devices. This could alleviate the problem of lack of resources faced by IoT devices, but a drop in accuracy would be inevitable. When end-user devices intend to perform certain calculations but are not equipped with sufficient resources or are overloaded, they have the opportunity to transmit their assigned tasks to other devices over the network given the interconnectivity between different nodes. A proper offloading strategy is crucial to avoid situations where certain nodes in an architecture absorb all tasks from nearby end devices. As a result, load balancing between nodes must be ensured and all tasks must be executed successfully.

In addition, depending on the environment in which this paradigm is located or in which the application is intended to be used, some alternatives will be more beneficial. For example, if there are latency requirements for tasks, streaming to the nearest nodes will be more appropriate than streaming to the cloud, although cloud servers are unlimited in memory and offer the highest processing capabilities. The weakness of using this alternative is that transmitting information from end-user devices to the cloud involves some delay and possible loss of information over the network. This loss could result from connection loss or other erroneous message information. The information can be vulnerable to intrusion attacks and inaccurate or incomplete. Transferring confidential information to the cloud is not the right decision because the vulnerability grows with increased information exposure over the network.

In contrast, IoT devices do not expose information over the network when they perform a task, making them the most secure option. The latency required by many applications also prevents using the cloud as a final computing center due to increased delay. The essential requirement of RT computing is immediate response, which is impossible to achieve using cloud computing. On the other hand, end-user devices offer immediate feedback, but their limitations can lead to a lack of model accuracy.

Excessively reducing the size of the models and the data needed to represent them leads to a severe drop in the performance of the resulting models. For example, a simple actuator has to give a specific response depending on the values in the environment. In order to obtain the answer, it can be necessary to apply an ML model that would not be feasible to compute on the IoT device or not with its original structure. However, the interconnectivity between different computers at different layers offers the possibility of computing these models in other computing centers, alleviating the computational load of these tiny devices.

In order to solve the problems mentioned above, it is essential to establish an appropriate task offloading policy. This would indicate in each case if it is necessary to carry out the download process, and if so, what the most appropriate destination will be in each case.

There are many alternatives intended to help with the task offloading decision problem. Some researchers opted for optimization algorithms. Other studies have chosen methods based on AI. Other alternatives, such as population-based and control theory methods, are outside the research interest.

This study applies Graph Neural Networks (GNNs) and Deep Q-Networks (DQN) to decide if it is feasible to offload a task from an end-user device to a richer computing center in terms of processability and available memory in each situation and to determine which is the best destination available in the area. It represents the local network structure in which different types of devices can be found in the Graph structure where each node is a computer, and the edges are the interconnections between them. This architecture can be extrapolated to an area of local networks where different IoT devices are interconnected with each other and with more powerful Edge Datacenters that offer the possibility of offloading tasks from small devices to Datacenters such as a Smart Home with small gadgets and a central router. In addition, the agent learns the network environment in the DQN learning process, where each action will be a decision to offload the task to one of the potential destination computers surrounding the source computer originating the task. Each state will represent each situation where all the characteristics of computers will be reflected. In each scenario, this research establishes a general remote cloud server that will serve as the orchestrator of the offload strategy and a fixed number of Edge Computing data centers and end-user devices or Edge devices. The proposed methods are evaluated by observing the success rate of the generated tasks, workload balance, and energy consumption. Finally, This study analyzes which computing nodes are most suitable for downloading the success rates of each device type.

The main contributions made in this work are the following. We offer a novel alternative to establish a task offloading strategy in a local network environment. The network architecture is almost replicated in the GNN architecture and the quality of a network connection for download issues has been rated with a novel parameter called download rating. Furthermore, environment updates are fully considered in the DQN learning process. Our methodology offers an innovative way to offload tasks in a local network environment, ensuring load balancing and completion of tasks within the desired latency. An overview of the procedure is given in Fig. 1.

The rest of the paper is organized as follows. Section 2 reviews some of the most representative works published in the literature. Section 3 specifies the new algorithms proposed by this work. Section 4 presents the materials and methodology applied in this work. In Section 5, we carry out different experiments of the task offloading paradigm using a known simulator and the results are presented. In Section 6, these results are analyzed and conclusions are reached.

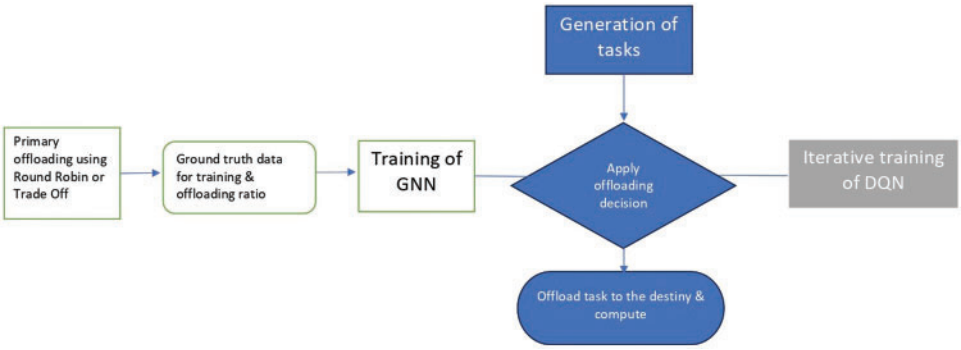

Figure 1: Overview of the entire process

2 State of the Art

Over the last decade, several researchers have found the task offloading paradigm a conflict of interest. The opportunity to transfer tasks from resource-constrained devices to resource-rich computing data centers can alleviate the computational load on end-user devices and complete tasks that were not feasible to complete at the source due to processing and memory constraints.

Those techniques have been widely used in different areas. Different techniques have been used in virtual reality (VR) applications, such as fog computing-based radio access networks (F-RAN) [3] or ML-based intelligent programming solutions [4]. In autonomous vehicle applications, task offloading techniques have been used to improve performance by reducing latency and transmission cost, as was done in [5]. Real-time traffic management was feasible by distributing decision-making tasks to Edge devices [6]. In the area of robotic task offloading, new paradigms have emerged, in [7], they presented an approach to simultaneous localization and mapping (SLAM) for RGB-D cameras like the Microsoft Kinect, and in [8], a novel Robot-Inference-and-Learning-as-a-Service (RILaaS) platform for low-latency and secure inference serving of deep models that could be deployed on robots was introduced. Nonetheless, there are already commercial solutions for offloading tasks in robotics [9–11]. Similarly, cloud-based solutions can be found in video streaming applications [12,13]. However, offloading Edge should improve performance as in [14,15], by enabling gateways and facilitating caching and transcoding mechanisms, respectively. The challenge of transferring computationally expensive tasks to Edge nodes has been addressed in [16–18] in the area of disaster management, but is still underexplored in this field. In the IoT field, task offloading has been of special interest since its inception, since these devices with limited resources often face this drawback. Due to the long delays involved in network transfer between IoT devices and the cloud, edge offloading needs to be considered. The collaboration between IoT devices and Edge devices could be useful in the area of smart health, being a good alternative to help paralyzed patients [19]. However, due to the growing number of IoT devices, the best option would be the collaboration between the Edge and Cloud servers as they did in [20] proposing a paradigm that foresees an IoT Cloud Provider (ICP)-oriented cooperation, which allows all devices that belong to the same public/private owner to participate in the federation process.

Different strategies have been proposed to solve the problem of task offloading. Optimization algorithms have become a very useful and frequently used solution for this paradigm. Mixed integer programming (MIP) has become a useful tool for resource allocation problems, addressing network synthesis and allocation issues [21]. In other words, they opted for greedy heuristic solutions [22–24] to

solve the task offloading problem. The main advantage of these is that they offer a low execution time, they do not require specialized optimization tools for their resolution and rather they can be expressed as pseudocode, easily implementable in any programming language. These become much more efficient when the task offloading problem is modeled as a nonlinear constrained optimization problem, or when the scale of the scenario is large enough [25]. In this case, a greedy heuristic could estimate the exact solution [13,22,24]. In other words, game theory was chosen, formulating the problem of partial task offloading in a multi-user infrastructure, Edge Computing and multi-channel wireless interference environment as an offloading game [26]. The Cloud-Edge game could be seen as an infrastructure game in which the players are the corresponding infrastructures [27]. Contract theory [28–30] and local search [31,32] are another type of optimization solutions for the task offloading problem.

Another interesting approach to solving the problem of task offloading is the use of methods based on AI. This branch includes all ML methods, including Supervised Learning, Unsupervised Learning, DL, and Deep Reinforcement Learning (DRL) methods. The download destination could be chosen following the simplest models, such as a regression model [33] or regression trees [34]. However, given the dynamism of network environments, modeling has been performed with the support vector regressor [35] and the nearest neighbor regressor [36] for future load prediction and energy efficient utilization of the Edge servers, respectively. In [37], a resource-aware offloading video analysis in Mobile Edge Computing and a resource-aware offloading (ROA) algorithm using the radial basis function networks (RBFN) method to improve reward were proposed under the resource deadline constraint. Taking into account unsupervised models, clustering models are useful tools to group resources depending on the distance between computing nodes [38] and task demands [39] and analyze the allocated resources [40].

DL can be an accurate tool for making task offloading decisions, based on the resource usage of the processing Edge nodes, the workload, and the quality of services (QoS) constraints defined in the Service Level Agreement (SLA) [41]. In [42], a new multi-objective strategy based on biogeography-based optimization (BBO) algorithm for Mobile Edge Computing (MEC) offloading was proposed to satisfy multiple user requirements (execution time, power consumption, energy, and cost). In [43], a task offloading model based on dynamic priority adjustment was proposed. Second, a multi-objective optimization model for task scheduling was constructed based on the task offloading model, which optimizes the time delay and energy consumption. In [44], they proposed an Improved Gorilla Troops Algorithm (IGTA) to offload dependent tasks in MEC environments with three objectives: minimizing the application execution latency, the power consumption of light devices, and the used cost of MEC resources. DL models have been used to minimize the computational load under dynamic network conditions and constrained computational resources [45]. A model that also considers the challenges of speed, power, and security, while satisfying QoS with dynamic needs, has been proposed to determine the combination of different computing nodes [46]. In [47], they developed a novel calibrated contextual bandit learning (CCBL) algorithm, where users learn the computational delay functions of micro base stations and predict the task offloading decisions of other users in a decentralized manner. At [48], they presented a novel federated learning framework for GAN, namely Collaborated g Ame Parallel Learning (CAP), which supports parallel training of data and models for GAN and achieves collaborative learning between edge servers, devices, and Cloud. Furthermore, they proposed a Mix-Generator (Mix-G) module that splits a generator into the sharing layer and the personalizing layer.

DRL techniques have emerged as an interesting alternative to typical task-offloading policies. Deep Q networks have been used to solve the task offloading problem [49] and have been optimized by introducing a short-term memory (LSTM) [50] into them. An intelligent partial offloading scheme was

proposed in [51], namely digital twin-assisted intelligent partial offloading (IGNITE), which combines the improved clustering algorithm with the digital twin (DT) technique, in which unreasonable decisions can be avoided by reducing the size of the decision space and finding the optimal offloading space in advance. In the same field, reference [52] proposed a mobility-dependent task offloading (MESON) scheme for urban vehicle edge calculation (VEC) and developed a DRL-based algorithm to train the offloading strategy. To improve the training efficiency, a vehicle mobility detection algorithm was further designed to detect the communication time between vehicles and Road Side Units (RSUs). In this way, MESON was able to avoid unreasonable decisions by reducing the size of the action space. Finally, the DRL algorithm was used to train the offloading strategy. In [53], they used the Markov decision process (MDP) that minimizes the total completion time. In [54], they considered a wireless MEC system that governs a binary offloading decision to execute the task locally on the Edge devices or on the remote server, proposing a Reinforcement Learning-based Intelligent Offloading online (RLIO) framework that adopts the optimal offloading policy.

Other approaches that differ from those mentioned above include population-based methods and control theory-based methods. Swarm Intelligence methods [55,56] and Evolutionary Algorithms [57,58] are the two variants of population-based methods that have been proposed to address the problem. Solutions based on control theory include optimal control [59,60], state feedback control [61] and Lyapunov optimization processes [62] among others.

Most of the mentioned studies implemented using an outdated methodology that has been surpassed by recent models such as deep models or Reinforcement Learning (RL) models, or those that chose to use these techniques are single objective and/and do not care about task' features and the actual workload of the destinations. In contrast, this study applies a simplistic approach that considers the nature of the tasks and the updated status of potential download destinations, facilitating user understanding while achieving high accuracy and competitive performance. It provides a methodology representing the network architecture in a graph and an RL technique that considers all the key factors when determining the optimal download decision. The research proposes a novel feature to evaluate the goodness of a download destination, which is a critical factor in determining whether a potential download route is valuable for a given task. Table 1 compares the latest and most relevant works, specifying the methodology proposed in each work.

Table 1: Comparative of current works on edge computing

Work	Proposed method	Field
[24]	Heuristic greedy offloading scheme	Multi-access mobile edge computing
[28]	Contract theory. Negotiation between task publisher and fog nodes as an optimization problem	Fog computing
[33]	Multi-task regression problem & Multi-task learning based feedforward neural network (MTFNN) model	Multi-access edge computing
[34]	Module placement method by classification and regression tree algorithm (MPCA) & Probability of network's resource utilization in the module offloading (MPMCP)	Mobile fog computing
[37]	Radial basis function networks-based resource-aware offloading	Video analytics in mobile edge computing

(Continued)

Table 1 (continued)

Work	Proposed method	Field
[38]	Balanced clustering and joint resources allocation (BCJRA)	Mobile fog computing
[39]	Dynamic mobile cloudlet cluster policy (DMCCP)	Fog computing
[40]	Server partitioning algorithm based on clustering. Multi-user game with Nash equilibrium	Mobile edge computing
[41]	Neural networks for mapping quality of service required levels and (expected) application workload to concrete resource demand	Edge computing
[42]	Multi-objective strategy based on the biogeography-based optimization (BBO) algorithm	Mobile edge computing
[43]	Task unloading model based on dynamic priority adjustment & Multi-objective optimization model	Task offloading & Real-time scheduling
[44]	Improved Gorilla troops algorithm (IGTA)	Multi-access edge computing
[46]	Deep learning-based dynamic task offloading in mobile cloudlet (DLDTO)	Mobile computing
[47]	Novel calibrated contextual bandit learning (CCBL) algorithm	Mobile edge computing into an ultra-dense network (UDN)
[48]	Collaborated game parallel learning (CAP) for GANs & Mix-generator module (Mix-G) that divides a generator into the sharing layer and personalizing layer	
[49]	Deep Q-learning approach for designing optimal offloading schemes, jointly considering selection of target server and determination of data transmission mode	Mobile edge computing
[50]	DRL & LSTM network layer and the candidate network set	Mobile edge computing
[51]	Digital twin-assisted intelligent partial offloading (IGNITE)	Vehicle edge computing
[52]	Mobility-aware dependent task offloading (MESON) scheme for urban VEC and a DRL-based algorithm to train the offloading strategy	Vehicle edge computing
[53]	Markov decision process	IoT & Edge computing
[54]	Reinforcement learning based intelligent online offloading (RLIO)	Mobile edge computing
[55]	Fuzzy clustering	Mobile edge computing

3 Proposed Algorithms

This study proposes a well-known DRL technique using GNN and DQN to solve the task offloading problem. Brief descriptions of both architectures are given in this section. Finally, the

training processes of both algorithms are explained utilizing Fig. 2, showing the complete architecture procedure.

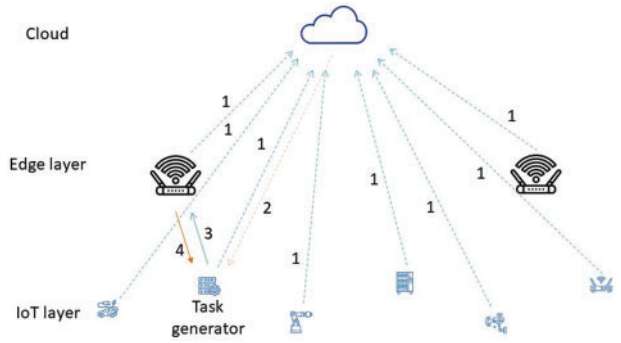

Figure 2: Training procedure of the entire architecture divided in different steps

3.1 Graph Neural Network

Graphs are a data structure representing a collection of elements (nodes) and their connections (edges). A GNN is a type of neural network (NN) that works directly with the graph's structure. In the used case, each node in the network represents a computing center that can be an IoT/Edge device, an Edge server, or a cloud server. Graphs are a data structure representing a collection of elements (nodes) and their connections (edges). The edges between these nodes represent the connections between the different computing centers, which can be the download paths of the tasks that must be completed to meet their requirements.

Each node represents each computing device, a potential destination for the download task in question. Furthermore, the edges represent the connection between these devices, whose characteristics are as follows. The characteristics of each node are determined by the computing device's available RAM, millions of instructions per second (MIPS), central processing unit and memory, and the desired task latency and file size in bits associated with the task. The characteristics of Edge are determined by the offload classification defined in this work, that is, the number of tasks successfully executed using the offload path divided by the total number of tasks offloaded using the path.

The output size of the network will be determined by the number of possible destinations of the task initially assigned to the IoT/Edge device. The number of output neurons will be equal to those possible destinations. The output would be binary, downloading/not downloading to each possible destination.

To train the network, we apply the real data produced by the architecture following two well-known offloading algorithms, Trade-Off and Round Robin. The download destinations for each task obtained following any of the mentioned algorithms would be the actual data used to train the network. Once the network is trained, the input would be the task with its characteristics and the output would be a binary decision of the possible download destinations. A brief description of our algorithm is provided in Algorithm 1.

Algorithm 1: GNN

Nodes ←Available computing centers
Edges ←Connection between computing centers
NodeFeatures ←RAM, Mips, CPU, latency, file size
for *NTasksExecuted Satisfactory* **do**
    $Destinynode \leftarrow TradeOff/Round - Robin(task)$
    $EdgesSuccessfullyExeceutedTasks \leftarrow EdgesSuccessfullyExeceutedTasks + 1$
end for
for NEdges **do**
    EdgeFeature ←EdgesSuccessfullyExeceutedTasks/NTasksOffloadedbyEdge
end for
for Ntasks **do**
    $Output \leftarrow GNN(task)$
    $Loss \leftarrow CrossEntropyLoss(Output, Destinynode)$
end for
for Ntasks **do**
    OffloadingDestiny ← GNN(task)
end for

3.2 Deep Q-Network

RL is a framework in which the agent attempts to learn from its environment by obtaining different rewards on each action performed in that environment. The agent's objective is to maximize the sum of rewards obtained by performing consecutive actions following its policy, and by optimizing this policy the problem in question is solved. After obtaining an observation of its environment (s_t) the agent acts a_t following its policy $\pi(a_t|s_t)$. Consequently, depending on the action performed in that observation, a reward and the next observation (s_{t+1}) are obtained.

DQN was developed by [63]. Deep neural networks (DNN) and replay techniques were used to optimize the Q-learning process. Q-learning is based on the function Q that measures the expected return or the discounted sum of rewards obtained from state s by taking action a first and following policy π. An optimal function Q^* is defined and, using the Bellman optimization equation (see Eq. (1)) as an iterative update, convergence of the function Q is guaranteed.

$$Q_{i+1}(s,a) = E[r + \gamma * max_{a'} Q_i(s', a')] \tag{1}$$

Representing the function Q by combining all possible actions and states is not the most practical option in most cases. For this reason, a function approximator is used for this. Using the NN approximation can be done using parameters θ and minimizing the loss function.

$$L_i(\theta) = E_{s,a,r,s'\rho}[(y_i - Q(s,a;\theta_i))^2] \qquad y_i = r + \gamma * max_{a'} Q(s', a'; \theta_{i-1}) \tag{2}$$

In the use case, the actions were the possible decision to download to each of the potential destinations on the network, given the state of the environment. The state of the environment will be determined by the task's characteristics and each device's state and capabilities. The properties of the task that conditioned the state of the environment were the maximum allowed latency and the size of the file in bits belonging to the task. Similarly, each computing device's available RAM, MIPS, central processing unit, and memory determined the rest of the state properties. If the task requirements were successfully met, the reward for downloading to a given computing center would

be 1, and −1 if the requirements for that action were not met. Following the technique above, the optimal download policy was obtained. Finally, the optimized policy would determine the optimal download destination for each task. Algorithm 2 summarizes the method.

Algorithm 2: DQN

for N tasks **do**
  $s \leftarrow$RAM,Mips,CPU,memory,latency and file size
  for N possible destinies **do**
    $a \leftarrow$ Possible destiny
    Calculate $L(\theta_i)$
  end for
  $a \leftarrow max_a Q(s, a; \theta)$
  $Q_{i+1} \leftarrow Q(s, a)$
end for

3.3 Training Procedure & Orchestration of Tasks

In the case of GNN, it is first necessary to perform a training process following any of the two default methods available in the simulator. In each iteration, any devices that make up the IoT layer will randomly create a task. All devices will send a message to the cloud reporting their actual status, even if they have a task to solve (Step 1 in Fig. 2). In this scenario, the cloud will orchestrate the download action following the predetermined algorithm by sending a message to the device (Step 2 in Fig. 2), and this will be downloaded to the destination (Step 3 in Fig. 2). After all, if the task has been completed by meeting the requirements, that will be a positive result for the subsequent training of the GNN; otherwise, it will be negative. Once the entire training procedure of the default algorithm is completed, the GNN will use the download decisions and the output generated in the previous step as ground truth and perform the training process after defining the download rating for each network connection. As an edge feature. For both training procedures, the graph shape was determined by the network structure (influenced by the number of IoT devices), the learning rate was 0.001, the optimizer was Stochastic Gradient Descent (SGD), and 10000 was the number of epochs. The GNN training will be conducted through the cloud. Finally, the cloud will decide to download after each device sends the message with the information about its status (Step 1 in Fig. 2), and the cloud will return the message to the task-generating device informing about the download destination (Step 2 in Fig. 2). After sending the task to the target device (Step 3 in Fig. 2) and completing the task on this device, the results will be sent back to the source device (Step 4 in Fig. 2).

In the case of DQN, each device will send information about its status to the cloud (Step 1 in Fig. 2). There, taking the state of the environment based on the offloading policy, the optimal action must be taken. If the task was completed meeting the requirements, the reward will be 1, and 0 otherwise. In this way, an optimal offloading policy will be obtained after converging the Q function.

Finally, the Q function obtained will determine the optimal download destination in the cloud after each device sends its state to the cloud, and it returns the message to the task generator indicating where to download its assigned task (Step 2 in Fig. 2). After sending the task to the target device (Step 3 in Fig. 2) and completing the task on this device, the results will be sent back to the source device (Step 4 in Fig. 2).

4 Material & Methodology

This section explains the environment in which the methodologies presented in the previous section were applied in the experimental process. The software and hardware used in the experiments are also described.

For the experimental processes we chose to use a well-known Edge Computing simulator called PureEdgeSim [64]. The simulator offers high configurability through its modular design. In this way, by editing each module and adjusting it to the user's needs, it is simple and feasible to reproduce the desired environment in each case.

The hardware environment in which all development of our work took place is a ×64 Ubuntu 20.04.4 LTS Operating System equipped with an Intel Core i7-11850H working at 2.5 GHz ×16 and 32 GB DDR-4 RAM and a NVIDIA T1200 Laptop GPU (driver version: 510.47.03, CUDA version:11.6).

The study established between 10 and 30 end-user devices in this case, forming the IoT-Edge layer. It repeated the experiment 3 times and compared the results of applying the abovementioned algorithms to make task-offloading decisions. These devices were dynamic, and their range of motion was limited to 200 × 200 units. The Fog-Edge layer comprised four data centers, each located symmetrically in the coverage area. Each of these Edge Data Centers covered an area of 100 × 100 units. Finally, a resource-rich cloud platform offered greater computing and memory.

Each of the end-user devices was interconnected with each other. In this way, interconnections between them were feasible. Similarly, each of these end-user devices was connected to the nearest Edge Datacenters, and all were connected to the cloud.

The orchestrator of the decision to download was the cloud. It was equipped with 200 cores, 40,000 MIPS, 16 GB of RAM, and 1 TB of memory.

The Edge Datacenters were equipped with ten cores, 40,000 MIPS, 16 GB of RAM, and 200 GB of memory. Its idle power consumption was 100 Wh, with a maximum consumption of 250 Wh.

Finally, the number of Edge devices or end-user devices was 10, 20, and 30 in each experimental test. Their operating system was Linux and they had an architecture of ×86. These devices had dynamic behavior in some cases, with a speed of 1.8 m/s. The type of network connection used to interconnect with the rest of the devices was WiFi with a bandwidth of 1300 Mbits/s, with a latency of 0.005 s. There were 5 different types of Edge devices and their characteristics are summarized in Table 2.

Table 2: Characteristics of different types of Edge devices

Device type	Type 1	Type 2	Type 3	Type 4	Type 5
Speed (m/s)	1.8	0	0	0	1.8
Pause duration (m/s)	100–400	0	0	0	100–400
Mobility duration (m/s)	60–100	0	0	0	60–100
Battery powered	Yes	No	Yes	No	No
Battery capacity (Wh)	18.75	–	56.2	–	–
Initial battery (%)	100	–	100	–	–
Idle energy consumption (Wh)	0.2	3.8	1.7	0.4011	0.4011
Max. energy consumption (Wh)	5	5.5	23.6	0.436	0.436
Cores	8	4	8	0	0

(Continued)

Table 2 (continued)

Device type	Type 1	Type 2	Type 3	Type 4	Type 5
MIPS	25000	16000	110000	0	0
RAM (GB)	4	4	8	0	0
Storage (MB)	256	128	256	0	0
Percentage	18	11	11	28	32

Each device could spawn any applications or tasks whose specifications are summarized in Table 3. Container size refers to the size of the application in kB. The request size refers to the download request sent to the orchestrator and then to the device where the task will be downloaded in kB. Result size refers to the downloaded task results in kB.

Table 3: Characteristics of different types of tasks

Application type	Hard real-time	Soft real-time	Non real-time
Generation rate/s	20	30	3
Latency (s)	0.02	0.5	300
Task length (Millions of instructions)	500	5000	30000
Container size (kB)	20	1500	2200
Request size (kB)	20	1500	2500
Results size (kB)	20	50	200
Percentage	20	30	50

This study applied the offload decision algorithms against the default methods provided by the simulator, Round Robin, and Trade-Off. It introduced different options regarding possible download destinations by including all devices, Edge devices only, Edge Data Centers only, Edge Data Centers and cloud-only, Edge devices and cloud-only, and Edge devices and Edge Data Centers only.

In total, there were 6 offloading configurations $\times$ 3 number of Edge device possibilities $\times$ 4 algorithms $=$ 72 simulation configurations. Each simulation time was established at 200 s.

5 Experiments & Results

In the experimental process, we considered the following parameters: energy consumption, tasks executed in each layer, and success rate. Additionally, we considered the distribution of the workload among different devices. Task failure could be due to different reasons, such as lack of available memory, violation of latency constraints, or network traffic congestion. His explanation is given below:

- Success rate: The ratio of the number of successfully executed tasks divided by the total number of tasks.
- Energy consumption: The power consumed by all devices of each type during each experimentation process.

- Workload distribution: Refers to the number of tasks distributed by each type of device in each experimentation process.

5.1 Tests with 10 Edge Devices

First, only ten devices were placed in the end-user layer, and these devices were divided into various types following the percentages shown in Table 2. These randomly generated the three types of tasks following the percentages and generation rates listed in Table 3. The success rate results are depicted in Table 4.

Table 4: Success rate of different algorithms including different types of destiny devices (10 Edge devices)

Algorithm	All devices	Edge & Edge DC	Edge & Cloud	Edge DC & Cloud	Edge	Edge DC
Trade Off	99.9194%	100%	66.0484%	99.9194%	54.5161%	100%
Round Robin	99.8387%	100%	47.1774%	99.8387%	55.4839%	100%
GNN	99.9194%	100%	80.4032%	99.9194%	59.1935%	100%
DQN	99.9194%	100%	82.0968%	100%	65.5645%	100%

As can be seen, the most critical environments were when there were no Edge Data Centers available as possible download destinations. This could be because the Edge devices were not equipped with sufficient capabilities to computationally support the rest of the devices' tasks. Likewise, the cloud was too far from these end-user devices, so latency requirements were not met in most cases where the cloud was the download destination. Those problems were solved when tasks were offloaded to Edge Data Centers, which are computationally less powerful than the cloud platform but still have high capabilities. In the same way, being located close to these Edge devices, task latency was not an issue.

The energy consumption of Edge devices and Edge Data Centers are shown in Tables 5 and 6, respectively. Energy consumption was higher in cases where there were no Edge Data Centers available as possible download destinations, and the algorithm chosen to decide the download destination was GNN or DQN. In these cases, because the best download destinations were not available, the Edge devices that can perform the tasks consume more power. However, in the default algorithms, this is not the case. Most of the tasks could have been offloaded to the cloud, which hurts the success rate, as seen in Table 4. Edge Data Centers show a similar consumption pattern for all algorithms, slightly lower for all download policies when they were not potential download destinations.

Table 5: Energy consumption in Wh of Edge devices for different algorithms including different types of destiny devices (10 Edge devices)

Algorithm	All devices	Edge & Edge DC	Edge & Cloud	Edge DC & Cloud	Edge	Edge DC
Trade Off	3.1947	3.2359	3.9221	3.1246	3.5854	3.1246
Round Robin	3.1032	3.1246	3.1184	3.1246	3.6223	3.1246
GNN	3.1745	3.2707	4.3367	3.1246	3.6223	3.1246
DQN	3.1487	3.1902	4.5019	3.1246	3.6406	3.1246

Table 6: Energy consumption in Wh of Edge Datacenters for different algorithms including different type of destiny devices (10 Edge devices)

Algorithm	All devices	Edge & Edge DC	Edge & Cloud	Edge DC & Cloud	Edge	Edge DC
Trade Off	34.4092	34.7847	33.8889	34.5972	33.6111	34.9722
Round Robin	34.3193	34.9722	33.8889	34.5966	33.8889	34.9722
GNN	34.3577	34.6034	33.8889	35.0087	33.8889	34.9722
DQN	34.4162	34.8765	33.8889	35.0125	33.8889	34.9722

5.2 Tests with 20 Edge Devices

We repeated the experiment from the previous subsection by changing the number of Edge devices to 20. The results of the success rates are shown in Table 7.

Table 7: Success rate of different algorithms including different type of destiny devices (20 Edge devices)

Algorithm	All devices	Edge & Edge DC	Edge & Cloud	Edge DC & Cloud	Edge	Edge DC
Trade Off	99.8641%	100%	61.8478%	99.8913%	97.1739%	100%
Round Robin	99.8370%	99.7011%	96.30434%	99.8370%	59.9185%	100%
GNN	99.9185%	100%	84.4837%	99.9185%	97.1739%	100%
DQN	99.9457%	100%	84.3478%	99.9728%	97.1739%	100%

The success rate trend continued when we doubled the number of Edge devices. However, as the number of free Edge devices must have been higher than in the previous experiment the success rates were higher when Edge devices were involved and not Edge Data Centers. In cases where Edge Data Centers were possible destinations, rates are 100% or close to it. In this test, when Edge Datacenters were not potential destinations, performance degradation was observed when Edge devices were the only potential destinations for the Round Robin algorithm and when the cloud was also a potential offload destination for the Trade-Off algorithm. In these cases, the Edge devices would not be sufficient to respond to the tasks and the cloud would not respond within the desired latency, respectively.

Tables 8 and 9 show the energy consumption of edge devices and Edge Data Centers, respectively.

Table 8: Energy consumption in Wh of Edge devices for different algorithms including different types of destiny devices (20 Edge devices)

Algorithm	All devices	Edge & Edge DC	Edge & Cloud	Edge DC & Cloud	Edge	Edge DC
Trade Off	8.3796	7.7330	8.7102	7.0386	9.3050	7.0386
Round Robin	7.6535	6.9917	8.6272	7.0386	8.0294	7.0386
GNN	7.6113	7.2317	8.7088	7.0386	9.3050	7.0386
DQN	6.6980	7.5004	8.7049	7.0386	9.3286	7.0386

Table 9: Energy consumption in Wh of Edge Datacenters for different algorithms including different type of destiny devices (20 Edge devices)

Algorithm	All devices	Edge & Edge DC	Edge & Cloud	Edge DC & Cloud	Edge	Edge DC
Trade Off	34.9217	35.4306	33.8889	36.1160	33.8889	36.6806
Round Robin	34.9298	35.9590	33.8889	35.9284	33.8889	36.6806
GNN	34.9240	36.6409	33.8889	36.1788	33.8889	36.6806
DQN	34.3268	35.7012	33.8889	36.2064	33.8889	36.6806

There is no evident variation in the power consumption of Edge Data Centers when doubling the number of Edge devices generating tasks. However, the power consumption of the Edge devices was almost double that in the previous case since more tasks were generated while the number of Edge Data Centers was fixed. Since computing platforms with higher capabilities were the same for a larger number of tasks, more tasks were offloaded to devices with limited resources. As in the previous experiment, Edge device consumption was slightly higher for cases where Edge Data Centers do not receive any tasks.

5.3 Tests with 30 Edge Devices

Finally, we replicated the experiment from the previous subsections by changing the number of Edge devices to 30. The results of the success rates are shown in Table 10.

Table 10: Success rate of different algorithms including different types of destiny devices (30 Edge devices)

Algorithm	All devices	Edge & Edge DC	Edge & Cloud	Edge DC & Cloud	Edge	Edge DC
Trade Off	99.9324%	99.8986%	70.2196%	99.8986%	89.0541%	100%
Round Robin	99.1892%	99.4595%	91.5541%	99.8311%	99.3986%	100%
GNN	99.9493%	100%	92.3142%	99.9324%	93.9696%	100%
DQN	99.9662%	100%	94.0372%	99.9662%	94.5777%	100%

Overall the success rates are better than with 10 Edge devices but comparable to the case of 20 devices.

The power consumption of the Edge devices and Edge Data Centers are shown in Tables 11 and 12, respectively. The increase was proportional to previous cases, with a low variation in the consumption of Edge Data Centers and a significant variation in the consumption of Edge devices. Apart from the lineal increase in consumption due to the higher number of tasks computed in these types of devices, in the case when Edge devices and Cloud where potential destinies the highest energy consumptions were reported when DQN and GNN were the applied algorithms, as consequence of a higher number of tasks offloaded to Edge devices than to the cloud. As in the previous cases, when a greater proportion of tasks were offloaded to Edge devices (higher energy consumption) the success rates were higher, due to the latency violation that occurred when the cloud was in charge of performing the task.

Table 11: Energy consumption in Wh of Edge devices for different algorithms including different types of destiny devices (30 Edge devices)

Algorithm	All devices	Edge & Edge DC	Edge & Cloud	Edge DC & Cloud	Edge	Edge DC
Trade Off	14.1875	11.6458	12.4942	10.6673	13.8637	10.6673
Round Robin	10.7567	12.1466	12.9142	10.6673	13.8015	10.6673
GNN	12.9003	12.0004	13.2908	10.6673	13.8015	10.6673
DQN	11.9993	11.8096	13.4561	10.6673	13.8015	10.6673

Table 12: Energy consumption in Wh of Edge Datacenters for different algorithms including different types of destiny devices (30 Edge devices)

Algorithm	All devices	Edge & Edge DC	Edge & Cloud	Edge DC & Cloud	Edge	Edge DC
Trade Off	34.8949	36.0532	33.8889	37.1575	33.8889	38.0972
Round Robin	35.6297	36.4035	33.8889	36.9680	33.8889	38.0972
GNN	35.1027	36.3284	33.8889	37.0880	33.8889	38.0972
DQN	35.9075	36.1982	33.8889	37.2137	33.8889	38.0972

5.4 Task Distribution with Varying Edge Devices

Next, this study established all types of computing devices as possible offloading destinations, and by varying the number of Edge devices between 10 and 30, as in the previous tests, this research observed the distribution of the download destinations in each case. The algorithms considered were GNN and DQN.

Fig. 3 indicates that the incremental trend toward computing tasks on Edge devices remains for both algorithms as the number of Edge devices grows and, consequently, the number of generated tasks does as well. This agrees with the increase in energy consumption observed in the previous sections. For both algorithms, the Edge Data Centers cannot attend to more tasks in the case of 30 Edge devices, relegating the rest of the tasks to the Edge devices, consequently having them attend to more tasks.

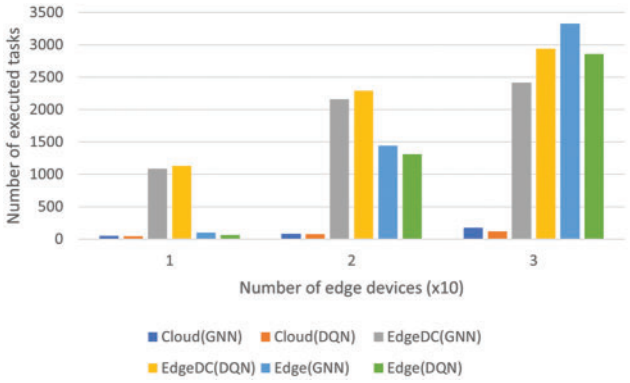

Figure 3: Task distribution with different algorithms and number of Edge devices

5.5 Success Rate of Different Layers

Finally, we established as possible offloading destinies all types of computation devices and varied the number of Edge devices between 10 and 30 as in previous tests we observed the success rate of different layers, to determine the optimal destination for computing the tasks generated by end-user devices. The algorithms regarded were GNN and DQN.

In this case, the worst results were given by the cloud platform. Although in terms of computational capabilities, it is the best option compared to the rest of the devices, the latency requirements were more difficult to meet due to the long time required to cross the entire network. This was expected. However, Edge devices were as good as Edge Datacenters in terms of accuracy for 10 and 20 devices. In the latter case, where 30 Edge devices were generating tasks, the Edge Data Centers were fully occupied, so more tasks were offloaded to resource-constrained devices. In isolated cases, the Edge devices were not able to complete the task, which is the reason why in the last case the Edge devices did not reach 100% accuracy for both algorithms. Fig. 4 shows the success rates for each layer & algorithm with different numbers of Edge devices.

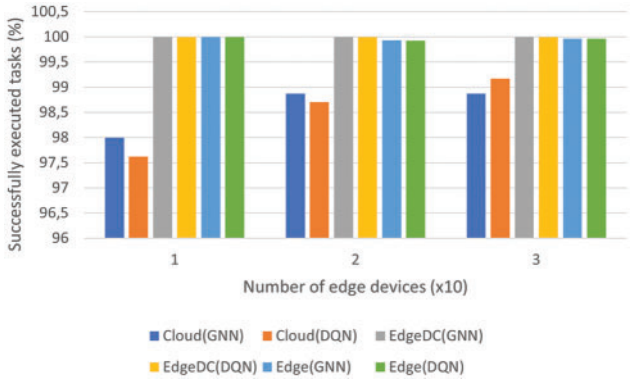

Figure 4: Success rates with different algorithms and number of Edge devices

6 Discussion & Conclusion

Section 5 presents the results obtained in the experimental process and observes different parameters.

Regarding success rate, there is a clear trend in favor of cases where Edge Data Centers were included in the download destinations. This is because they had sufficient computability and were located close to the Edge devices from where the tasks were generated. The worst results were obtained when only the cloud was available as a powerful computing center. In this situation, where network traffic congestion will have caused longer delays in task responses, meeting latency requirements will have been challenging. Tasks that had been offloaded to other Edge devices cannot be executed due to the lack of resources. As the number of Edge devices grew, the success rates of cases where Edge Data Centers were excluded improved significantly due to the increased number of free Edge devices. In this case, the number of Edge devices with sufficient computability grew, and fewer tasks had to be transmitted to the cloud.

Considering the differences between the different algorithms regarding success rate, there was a slightly favorable trend toward DQN, especially when the number of Edge devices was significant. In this case, there were possible offloading destinations, that is, more possible actions given the state of the environment. By learning an optimal policy, it will be more feasible to reach the optimal download

destination with this algorithm. GNN also outperformed the two default simulator methods when the number of Edge devices was 20 and 30. This was because the graph was complex, and although optimizing the network would be more difficult, the decision to offload was closer to optimal.

In terms of energy efficiency, there was no big difference between the first 3 tests. Obviously, in the case of 20 and 30 devices, the power consumption of Edge devices grew linearly from ∼3 to ∼10 Wh and ∼13 Wh, respectively, due to the larger amount of generated and downloaded tasks to these devices. On the contrary, the energy consumption of the Edge Data Centers almost remained at ∼34 Wh, even though the number of generated tasks increased. This was because the Edge Data Centers were full and other types of devices were needed to handle the rest of the tasks. This would have caused a reduction in the success rate, especially in the case of the 2 default algorithms and when the cloud and Edge Data Centers were included. In this situation, the rest of the tasks that were not attended to by the Edge Data Centers would have been offloaded to the cloud, meeting the problems mentioned in the previous paragraphs.

Regarding the distribution of tasks between different types of devices, we observed that when the number of Edge devices was not too high (10 or 20 devices), the Edge Data Centers were the destinations for most tasks. In contrast, when the number of devices grew to 30, they did not have enough free memory space, or their processors were busy. As a result, more tasks were offloaded to Edge devices, and a slight increase in the number of tasks offloaded to the cloud was also observed. In the experiment, this study compared only the proposed algorithms since they had the best performance in terms of success rate. Among them, DQN decided to download more tasks to Edge Data Centers, becoming a better alternative due to the better performance when Edge Data Centers were included in the possible download destinations.

Finally, the success rates of different types of devices were carefully compared. This study established all types of devices as possible destinations for the two proposed algorithms. It changed the number of Edge devices to between 10 and 30. There was a clear difference between the performance of the cloud and the rest of the devices. As mentioned in this section, the violation of the latency requirement is responsible for such performance degradation, given the high delay caused when traversing the network to transfer the task to the cloud and return the results to the Edge devices. Between Edge devices and Edge Data Centers, the latter had the best success rate. The larger capacities and larger memory were superior in computability compared to Edge devices. However, with an algorithm good enough to orchestrate all tasks between all possible destinations, offloading the less demanding tasks to the weakest computation centers, the success rate can be preserved with a higher number of generated tasks. That is why the proposed algorithms outperform the two default algorithms: they can offload less demanding tasks to weaker devices and more complex ones to Edge Data Centers. In this way, the task load was balanced between all available devices, meeting latency requirements.

We saw that our proposed algorithms outperform the default PureEdgeSim simulator methods in terms of success rate and load balancing. For example, for the case in which 30 Edge devices generated tasks, GNN and DQN achieved an improvement of 22.1% and 23.8% respectively concerning the Trade-Off when the Edge Datacenters were not included as potential destinations. However, GNN achieved an average improvement of 3.6% concerning Trade-Off and Round Robin, and DQN achieved an average improvement of 4.1% concerning Trade-Off and Round Robin. In other works, such as [53], they achieved an average improvement of 20.48%, 16.28%, and 12.36% concerning random download, higher data rate download (HDR), and the largest computing device (HCD), respectively. In [51], they achieved a 20% reduction in total computation delay and a 25% reduction in average computation delay compared to the GK-means DQN-based offloading policy. In our case, the

rest of the offloading policies analyzed offered a better result since they offered decent behavior in most cases. However, our methods significantly improved the success rates of the mentioned algorithms offer quite similar energy consumption, and have more to do with the distribution of tasks in different layers. Our network environments and experimental setup are completely different compared to those used in the works just mentioned. Therefore, the comparison cannot be made directly between different works. The distribution of tasks was different in our two algorithms, with a larger number of tasks being offloaded to the Edge Data Centers when DQN was applied. This resulted in a slight improvement in the success rate due to the greater capabilities of this type of computing center. Between both types of algorithms, the best results were offered by DQN with a slight variation. The ability to obtain the optimal policy increased when the number of Edge devices and, consequently, the number of generated tasks was larger. The same was true for GNN: by having more nodes and a broader network structure, the algorithm was able to reach a near-optimal offloading decision.

These algorithms could be a useful tool to provide proper orchestration in an environment where many IoT devices are requested to solve complex tasks and the characteristics of the environment are constantly updated. For example, in a Smart Building, several sensors can be located that detect different parameters and have to react by activating any other system based on the readings they obtain. Deciding what action to take may require the use of ML or DL techniques to take the optimal action. In this situation, these small devices could alleviate the computational burden of these deep models by offloading them to other powerful devices such as Edge Data Centers.

In this research, the introduction of GNN and DQN to the paradigm of task-offloading in a local network environment involving IoT, Edge, and Cloud layers is carried out. The similarity between the architectures of a graph and a local network involving the just mentioned devices favored the use of GNN to satisfactorily solve the task offloading paradigm. The offloading ratio used as an edge feature in this study is a good predictor of how good a potential target can be at accomplishing a task. Furthermore, the use of DQN slightly improved the results obtained with GNN. The learning process of the latter favors the consideration of the constant updates of an environment. The novelty offered using the proposed methodology in a local networking environment is the consideration of constant network updates and the scoring of network connections using the novel offloading rating parameter, being both GNNs and DQN powerful tools to impose an optimized offloading strategy in an environment made up of resource-constrained devices.

Among the limitations found during this research work, it is worth highlighting the difficulties in reproducing other methodologies provided by the literature using the PureEdgeSim simulator. The complexity of the simulator was an advantage in adjusting the properties of the network environment to the needs. On the contrary, the reproduction of any algorithm has a high complexity. At the same time, as other environmental properties can directly impact the state of the network, such as vandalism attacks, natural disasters, or intrusion attacks, these must be considered in the simulation, applying a random appearance factor to them.

In future work, more algorithms can be implemented using the simulator to compare them with those presented in this study. Until now, the only default implementable algorithms for the simulator in question were tried and tested against our methods, and due to the complexity of the simulator, no others were implemented. Other types of network structures can be interesting for research and applicable using the methodology proposed in this work. Furthermore, combining RL techniques with the graph will open up an exciting research area.

Acknowledgement: The authors wish to express their appreciation to the reviewers for their helpful suggestions which greatly improved the presentation of this paper. This work is partially supported

by the project a Optimization of Deep Learning algorithms for Edge IoT devices for sensorization and control in Buildings and Infrastructures (EMBED) funded by the Gipuzkoa Provincial Council and approved under the 2023 call of the Guipuzcoan Network of Science, Technology and Innovation Program with File Number 2023-CIEN-000051-01.

Funding Statement: This work has received funding from TECNALIA, Basque Research and Technology Alliance (BRTA). This work is partially supported by the project a Optimization of Deep Learning algorithms for Edge IoT devices for sensorization and control in Buildings and Infrastructures (EMBED) funded by the Gipuzkoa Provincial Council and approved under the 2023 call of the Guipuzcoan Network of Science, Technology and Innovation Program with File Number 2023-CIEN -000051-01.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: A. Garmendia-Orbegozo, J. D. Nunez-Gonzalez and M. A. Anton; data collection: A. Garmendia-Orbegozo, J. D. Nunez-Gonzalez and M. A. Anton; analysis and interpretation of results: A. Garmendia-Orbegozo, J. D. Nunez-Gonzalez and M. A. Anton; draft manuscript preparation: A. Garmendia-Orbegozo, J. D. Nunez-Gonzalez and M. A. Anton. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: All data used in the experimental process was generated using the PureEdgeSim simulator, which is perfectly reproducible following the instructions of Section 4.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Yadav, C. S., Pradhan, M. K., Gangadharan, S. M. P., Chaudhary, J. K., Singh, J. et al. (2022). Multi-class pixel certainty active learning model for classification of land cover classes using hyperspectral imagery. *Electronics, 11(17),* 2799.
2. Haq, M. A. (2022). SmoteDNN: A novel model for air pollution forecasting and AQI classification. *Computers, Materials & Continua, 71(1),* 1403–1425.
3. You, D., Doan, T. V., Torre, R., Mehrabi, M., Kropp, A. et al. (2019). Fog computing as an enabler for immersive media: Service scenarios and research opportunities. *IEEE Access, 7,* 65797–65810.
4. ComÅa, I. S., Muntean, G. M., Trestian, R. (2021). An innovative machine-learning-based scheduling solution for improving live UHD video streaming quality in highly dynamic network environments. *IEEE Transactions on Broadcasting, 67(1),* 212–224.
5. Fraedrich, E., Cyganski, R., Wolf, I., Lenz, B. (2016). User perspectives on autonomous driving a use-case-driven study in Germany. https://core.ac.uk/download/pdf/31023753.pdf (accessed on 21/07/2023).
6. Wang, X., Ning, Z., Wang, L. (2018). Offloading in Internet of Vehicles: A fog-enabled real-time traffic management system. *IEEE Transactions on Industrial Informatics, 14(10),* 4568–4578.
7. Song, D., Tanwani, A. K., Goldberg, K., Siciliano, B. (2019). *Networked-, cloud- and fog-robotics*. Springer. Robotics Goes MOOC, Springer Nature MOOCs, Bruno Siciliano (Editor).
8. Tanwani, A. K., Anand, R., Gonzalez, J. E., Goldberg, K. (2020). RILaaS: Robot inference and learning as a service. *IEEE Robotics and Automation Letters, 5(3),* 4423–4430.
9. A.W. Services, AWS robomaker (2021). https://aws.amazon.com/robomaker/ (accessed on 21/07/2023).
10. Google, cloud robotics core (2021). https://googlecloudrobotics.github.io/core/ (accessed on 21/07/2023).

11. Rapyuta robotics (2021). https://www.rapyuta-robotics.com (accessed on 21/07/2023).
12. Papagianni, C., Leivadeas, A., Papavassiliou, S. (2013). A cloud-oriented content delivery network paradigm: Modeling and assessment. *IEEE Transactions on Dependable and Secure Computing, 10,* 287–300.
13. Bilal, K., Erbad, A., Hefeeda, M. (2017). Crowdsourced multi-view live video streaming using cloud computing. *IEEE Access, 5,* 12635–12647.
14. Fajardo, J. O., Taboada, I., Liberal, F. (2015). Improving content delivery efficiency through multi-layer mobile edge adaptation. *IEEE Network, 29,* 40–46.
15. Tran, T., Pandey, P., Hajisami, A., Pompili, D. (2017). Collaborative multi-bitrate video caching and processing in mobile-edge computing networks. *2017 13th Annual Conference on Wireless On-Demand Network Systems and Services (WONS)*. Jackson Hole, Wyoming, USA.
16. Kim, K., Hong, C. S. (2019). Optimal task-UAV-edge matching for computation offloading in UAV assisted mobile edge computing. *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*.
17. Chen, S. C., Lin, C. P., Hsu, H. C., Shu, J. H., Liang, Y. et al. (2019). Serum bilirubin improves the risk predictions of cardiovascular and total death in diabetic patients. *Clinica Chimica Acta, 488,* 1–6.
18. Avgeris, M., Spatharakis, D., Dechouniotis, D., Kalatzis, N., Roussaki, I. et al. (2019). Where there is fire there is smoke: A scalable edge computing framework for early fire detection. *Sensors, 19(3),* 639.
19. Jacob, S., Alagirisamy, M., Menon, V. G., Kumar, B. M., Jhanjhi, N. Z. et al. (2020). An adaptive and flexible brain energized full body exoskeleton with IoT edge for assisting the paralyzed patients. *IEEE Access, 8,* 100721–100731.
20. Farris, I., Militano, L., Nitti, M., Atzori, L., Iera, A. (2016). MIFaaS: A mobile-IoT-federation-as-a-service model for dynamic cooperation of IoT cloud providers. *Future Generation Computer Systems, 70,* 126–137.
21. Lee, E. K., Lewis, D. P. (2006). Integer programming for telecommunications. In: Resende, G. C., Pardalos, P. M. (Eds.), *Handbook of optimization in telecommunications*, pp. 67–102. Boston, MA, USA: Springer. https://doi.org/10.1007/978-0-387-30165-5_3
22. Ketyko, I., Kecskes, L., Nemes, C., Farkas, L. (2016). Multi-user computation offloading as multiple knapsack problem for 5G mobile edge computing. *2016 European Conference on Networks and Communications (EuCNC)*. Athens, Greece.
23. Bilal, K., Erbad, A., Hefeeda, M. (2017). Crowdsourced multi-view live video streaming using cloud computing. *IEEE Access,* vol. 5, pp. 12635–12647.
24. Guo, H., Liu, J., Zhang, J. (2018). Computation offloading for multi-access mobile edge computing in ultra-dense networks. *IEEE Communications Magazine, 56(8),* 14–19.
25. Zhao, Y., Hu, W., Yang, Z. (2015). High-resolution transmission electron microscopy study on reversion of al2cumg precipitates in alâcuâmg alloys under irradiation. *Micron, 76,* 1–5.
26. Chen, X., Jiao, L., Li, W., Fu, X. (2016). Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking, 24(5),* 2795–2808.
27. Liu, Y., Xu, C., Zhan, Y., Liu, Z., Guan, J. et al. (2017). Incentive mechanism for computation offloading using edge computing: A Stackelberg game approach. *Computer Networks, 129,* 399–409.
28. Zeng, M., Li, Y., Zhang, K., Waqas, M., Jin, D. (2018). Incentive mechanism design for computation offloading in heterogeneous fog computing: A contract-based approach. *2018 IEEE International Conference on Communications (ICC)*. Kansas, MO, USA.
29. Du, J., Gelenbe, E., Jiang, C., Zhang, H., Ren, Y. (2017). Contract design for traffic offloading and resource allocation in heterogeneous ultra-dense networks. *IEEE Journal on Selected Areas in Communications, 35(11),* 2457–2467.
30. Zhang, Y., Pan, M., Song, L., Dawy, Z., Han, Z. (2017). A survey of contract theory-based incentive mechanism design in wireless networks. *IEEE Wireless Communications, 24(3),* 80–85.

31. Gendreau, M., Potvin, J. Y. (2010). *Handbook of metaheuristics*, vol. 2. Springer.
32. Wang, Y., Breedveld, S., Heijmen, B., Petit, S. (2016). Evaluation of plan quality assurance models for prostate cancer patients based on fully automatically generated pareto-optimal treatment plans. *Physics in Medicine and Biology, 61,* 4268–4282.
33. Yang, B., Cao, X., Bassey, J., Li, X., Kroecker, T. et al. (2019). Computation offloading in multi-access edge computing networks: A multi-task learning approach. *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. Shanghai, China.
34. Rahbari, D., Nickray, M. (2020). Task offloading in mobile fog computing by classification and regression tree. *Peer-to-Peer Networking and Applications, 13,* 104–122.
35. Hu, R., Jiang, J., Liu, G., Wang, L. (2013). CPU load prediction using support vector regression and Kalman smoother for cloud. *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*. Philadelphia, PA, USA.
36. Farahnakian, F., Pahikkala, T., Liljeberg, P., Plosila, J. (2013). Energy aware consolidation algorithm based on K-nearest neighbor regression for cloud data centers. *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*. Dresden, Germany.
37. Appadurai, J., Prabaharan, S., Venkateswaran, N., Roseline, S., Rama, B. (2023). Radial basis function networks-based resource-aware offloading video analytics in mobile edge computing. *Wireless Networks.* https://doi.org/10.1007/s11276-023-03420-7
38. Cheng, H., Xia, W., Yan, F., Shen, L. (2019). Balanced clustering and joint resources allocation in cooperative fog computing system. *2019 IEEE Global Communications Conference (GLOBECOM)*. Big Island, Hawaii, USA.
39. Li, Y., Anh, N. T., Nooh, A. S., Ra, K., Jo, M. (2018). Dynamic mobile cloudlet clustering for fog computing. *2018 International Conference on Electronics, Information, and Communication (ICEIC)*. Honolulu, Hawaii, USA.
40. Li, G., Lin, Q., Wu, J., Zhang, Y., Yan, J. (2019). Dynamic computation offloading based on graph partitioning in mobile edge computing. *IEEE Access, 7,* 185131–185139.
41. Bouras, I., Aisopos, F., Violos, J., Kousiouris, G., Psychas, A. et al. (2023). Mapping of quality of service requirements to resource demands for IAAS. *Proceedings of the 9th International Conference on Cloud Computing and Services Science CLOSER*, pp. 263–270. Heraklion, Crete, Greece. https://doi.org/10.5220/0007676902630270
42. Li, H., Zheng, P., Wang, T., Wang, J., Liu, T. (2022). A multi-objective task offloading based on BBO algorithm under deadline constrain in mobile edge computing. *Cluster Computing, 26,* 4051–4067.
43. Dai, Z., Ding, W., Min, Q., Gu, C., Yao, B. et al. (2023). M-E-AWA: A novel task scheduling approach based on weight vector adaptive updating for fog computing. *Processes, 11(4),* 1053.
44. Hosny, K., Ibrahim Awad, A., Khashaba, M., Rushdy, E. (2023). New improved multi-objective gorilla troops algorithm for dependent tasks offloading problem in multi-access edge computing. *Journal of Grid Computing, 21,* 21.
45. Yu, S., Wang, X., Langar, R. (2017). Computation offloading for mobile edge computing: A deep learning approach. *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. Montreal, QC, Canada.
46. Rani, D., Muthukumar, P. (2021). Deep learning based dynamic task offloading in mobile cloudlet environments. *Evolutionary Intelligence, 14,* 499–507.
47. Zhang, R., Cheng, P., Chen, Z., Liu, S., Vucetic, B. et al. (2022). Calibrated bandit learning for decentralized task offloading in ultra-dense networks. *IEEE Transactions on Communications, 70(4),* 2547–2560.
48. Zhang, J., Zhao, L., Yu, K., Min, G., Al-Dubai, A. et al. (2023). A novel federated learning scheme for generative adversarial networks. *IEEE Transactions on Mobile Computing,* 1–17. https://doi.org/10.1109/lTMC.2023.3278668

49. Zhang, K., Zhu, Y., Leng, S., He, Y., Maharjan, S. et al. (2019). Deep learning empowered task offloading for mobile edge computing in urban informatics. *IEEE Internet of Things Journal, 6(5),* 7635–7647.
50. Lu, H., Gu, C., Luo, F., Ding, W., Liu, X. (2019). Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning. *Future Generation Computer Systems, 102,* 847–861.
51. Zhao, L., Zhao, Z., Zhang, E., Hawbani, A., Al-Dubai, A. et al. (2023). A digital twin-assisted intelligent partial offloading approach for vehicular edge computing. *IEEE Journal on Selected Areas in Communications, 41,* 3386–3400.
52. Zhao, L., Zhang, E., Wan, S., Hawbani, A., Al-Dubai, A. et al. (2023). MESON: A mobility-aware dependent task offloading scheme for urban vehicular edge computing. *IEEE Transactions on Mobile Computing*.
53. Maray, M., Mustafa, E., Shuja, J., Bilal, M. (2023). Dependent task offloading with deadline-aware scheduling in mobile edge networks. *Internet of Things, 23,* 100868.
54. Mustafa, E., Shuja, J., Bilal, K., Mustafa, S., Maqsood, T. et al. (2022). Reinforcement learning for intelligent online computation offloading in wireless powered edge networks. *Cluster Computing, 26,* 1053–1062.
55. Liu, J., Wei, X., Wang, T., Wang, J. (2019). An ant colony optimization fuzzy clustering task scheduling algorithm in mobile edge computing. *Security and Privacy in New Computing Environments: Second EAI International Conference, SPNCE 2019*. Tianjin, China, Springer.
56. Hussein, M. K., Mousa, M. H. (2020). Efficient task offloading for IoT-based applications in fog computing using ant colony optimization. *IEEE Access, 8,* 37191–37201.
57. Zhang, D., Haider, F., St-Hilaire, M., Makaya, C. (2019). Model and algorithms for the planning of fog computing networks. *IEEE Internet of Things Journal, 6(2),* 3873–3884.
58. Al-habob, A. A., Dobre, O. A., Garcia Armada, A. (2019). Sequential task scheduling for mobile edge computing using genetic algorithm. *2019 IEEE Globecom Workshops (GC Wkshps)*. Big Island, Hawaii, USA.
59. Li, Y. (2017). *Edge computing-based access network selection for heterogeneous wireless networks (Ph.D. Thesis)*. Université de Rennes 1, France.
60. Avgeris, M., Dechouniotis, D., Athanasopoulos, N., Papavassiliou, S. (2019). Adaptive resource allocation for computation offloading: A control-theoretic approach. *ACM Transactions on Internet Technology, 19(2),* 23. https://doi.org/10.1145/3284553
61. Kalatzis, N., Avgeris, M., Dechouniotis, D., Papadakis-Vlachopapadopoulos, K., Roussaki, I. et al. (2018). Edge computing in IoT ecosystems for UAV-enabled early fire detection. *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*. Taormina, Sicily, Italy.
62. Pu, L., Chen, X., Xu, J., Fu, X. (2016). 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),* 3887–3901.
63. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J. et al. (2015). Human-level control through deep reinforcement learning. *Nature, 518,* 529–533.
64. Mechalikh, C., Taktak, H., Moussa, F. (2020). PureEdgeSim: A simulation framework for performance evaluation of cloud, edge and mist computing environments. *Computer Science and Information Systems, 18(1),* 43–66.