IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 1, 2020, accepted July 20, 2020, date of publication July 24, 2020, date of current version August 5, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3011705

Advanced Deep Learning-Based Computational Offloading for Multilevel Vehicular Edge-Cloud Computing Networks

MASHAEL KHAYYAT[1], IBRAHIM A. ELGENDY[2,3], AMMAR MUTHANNA[4], (Member, IEEE), ABDULLAH S. ALSHAHRANI[5], SOLTAN ALHARBI[6], AND ANDREY KOUCHERYAVY[4]

[1]Department of Information Systems and Technology, College of Computer Science and Engineering, University of Jeddah, Jeddah 23218, Saudi Arabia
[2]School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150090, China
[3]Department of Computer Science, Faculty of Computers and Information, Menoufia University, Shibin Al Kawm 32511, Egypt
[4]Department of Communication Networks and Data Transmission, St. Petersburg State University of Telecommunications, 193232 St. Petersburg, Russia
[5]Department of Computer Science and Artificial Intelligence, College of Computer Science and Engineering, University of Jeddah, Jeddah 23218, Saudi Arabia
[6]Department of Computer and Network Engineering, College of Computer Science and Engineering, University of Jeddah, Jeddah 23218, Saudi Arabia

Corresponding author: Ibrahim A. Elgendy (ibrahim.elgendy@hit.edu.cn)

This work was funded by the University of Jeddah, Saudi Arabia, under grant No. (UJ-02-016-ICGR). The authors, therefore, acknowledge the University's technical and financial support with thanks.

ABSTRACT The promise of low latency connectivity and efficient bandwidth utilization has driven the recent shift from vehicular cloud computing (VCC) towards vehicular edge computing (VEC). This paper presents an advanced deep learning-based computational offloading algorithm for multilevel vehicular edge-cloud computing networks. To conserve energy and guarantee the efficient utilization of shared resources among multiple vehicles, an integration model of computational offloading, and resource allocation is formulated as a binary optimization problem to minimize the total cost of the entire system in terms of time and energy. However, this problem is considered NP-hard and it is computationally prohibitive to solve this type of problem, particularly for large-scale vehicles, due to the curse-of-dimensionality problem. Therefore, an equivalent reinforcement learning form is generated and we propose a distributed deep learning algorithm to find the near-optimal computational offloading decisions in which a set of deep neural networks are used in parallel. Finally, simulation results show that the proposed algorithm can exhibit fast convergence and significantly reduce the overall consumption of an entire system compared to the benchmark solutions.

INDEX TERMS Computation offloading, vehicular edge-cloud computing, autonomous vehicles, 5G, resource allocation, deep reinforcement learning.

I. INTRODUCTION

IN recent years, the Internet of Things (IoT) and wireless sensors have become more popular in daily life. Additionally, with the emergence of 5G technology, the capabilities of communication are gradually improving, which will lead to a proliferation of new applications with advanced features such as autonomous vehicles, virtual/augmented reality, face recognition, and e-Health [1], [2]. However, the limited computational capacity and battery power of vehicles pose a large challenge to meeting these requirements and ensuring the required quality of service (QoS)level [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Huan Zhou.

To address the contradiction between the requirements of these applications and the limitations of resource-constrained vehicles, a computational offloading concept has been introduced in which resource-intensive computations are migrated from the vehicles to a resource-rich server for remote execution, and the results are returned [4]–[8]. Vehicular cloud computing (VCC) was initially developed to provide vehicles with flexibility in computing, storage and service capabilities, thus reducing power consumption and enhancing application performance. However, high latency is considered the main challenge for VCC, which makes it unsuitable for real-time and delay-sensitive applications [9], [10].

Vehicular edge computing (VEC) is considered a viable and promising new solution for addressing VCC challenges that has recently been proposed and received much

attention [11]–[14]. In VEC, the computational and storage capabilities of cloud servers are deployed at the edge of the radio access network (e.g., roadside units - RSUs), which is in proximity to vehicles and can offer high QoS and a cost-efficient solution with low latency [15]. Applications such as on-vehicle cameras and embedded sensors can benefit from VEC and provide efficient and safe transportation systems.

With the development of computational offloading for VEC systems, numerous approaches and models have been recently proposed in which some systems handle a single objective and others handle multiple objectives [16], [17]. Additionally, conventional methods have mostly been used for solving these optimization models [13], [18], while different intelligent algorithms based on deep learning have been recently used [19], [20]. On the one hand, edge servers have a small scale with limited processing capacity in comparison with cloud servers, which may cause congestion and lead to a longer processing time delay for the computational offloading of a large number of vehicles if they offload their computational tasks to the same server [21]. On the other hand, finding an optimal solution in time-variant dynamic systems such as multiuser wireless VEC networks is challenging. Therefore, motivated by these considerations, a VEC network is considered in which our environment is composed of multiple edge servers that are connected to a single cloud server. In addition, there are sets of vehicles that can process their computational tasks remotely at one of the available edge servers or cloud servers. Furthermore, a distributed deep learning-based computational offloading algorithm is developed to handle the performance optimization. The main contributions of this study are summarized as follows:

- We formulate an integration model of computational offloading, and resource allocation as a binary optimization problem whose objective is to minimize the weighted sum cost of entire system in terms of the energy consumption and latency for multilevel vehicular edge-cloud computing networks.
- We transform the above problem into an equivalent reinforcement learning form and then, propose an efficient algorithm based on distributed deep learning to solve this problem and derive computational offloading decisions.
- Finally, simulation results are conducted to prove the effectiveness of our proposed model and algorithm on the overall performance of an entire system in terms of energy and time in comparison with other benchmark solutions.

The rest of the study is organized as follows. An overview of related works on computational offloading policies is presented in Section II. An introduction of our system model in terms of communication and computational models and the formulation of the optimization problem is presented in Section III. Next, we design a distributed deep Q learning algorithm for solving the optimization problem in Section III-D. Simulation experiments are conducted to demonstrate our offloading model and algorithm in Section V. Finally, Section VI concludes the paper.

II. RELATED WORK

Recently, numerous approaches and optimization models have been proposed for addressing the challenges of vehicles using mobile edge computing (MEC) by applying the computational offloading concept. Most of these optimization models can be solved using conventional methods [13], [18], whereas recently, deep learning algorithms have been recently adopted as an effective method for solving these models [19], [20]. In the following subsections, a brief overview of the common models based on the solving methods is presented.

A. CONVENTIONAL-BASED METHODS

Minimizing the weighted sum of the energy consumption and latency for vehicular users was considered the main goal of [22]. The authors jointly optimized the transmission power and uploading time, as well as the local CPU frequency and the offloading ratio for the computational and communication resources, respectively. In addition, an efficient algorithm was developed to derive the optimal computational offloading decision for this problem.

A cloud-mobile edge computing collaborative computation offloading approach was presented in [23], in which the resource allocation and computational offloading were jointly formulated as nonconvex optimization problem with the objective of maximizing the system utility. Then, game-theory and Lagrange multiplier methods were utilized to find the optimal resource allocation and offloading decision by designing a distributed computational offloading and resource allocation algorithm.

Xu *et al.* [24] proposed an edge computing-enabled computation offloading approach for addressing the privacy conflicts computational tasks for the internet of connected vehicles. First, the privacy conflicts between the computational tasks were formally analyzed. Then, the routing vehicles from the origin vehicle to the destination vehicle were acquired. Finally, an efficient nondominated sorting genetic algorithm was adopted to minimize the execution delay and reduce the energy consumption of edge computing devices. Zhou *et al.* [25] formulated a reverse auction-based incentive as an integer optimization problem whose objective was to maximize the mobile network operator revenue under delay constraints. In addition, two algorithms the greedy winner selection algorithm and the dynamic programming winner selection algorithm were developed to solve this problem in an efficient way.

Recently, a multi-user, multi-server vehicular edge computing environment was studied in [26], where the computational offloading and task scheduling were jointly formulated as a mixed-integer non-linear programming problem whose objective was to maximize system utility. In addition, a hybrid intelligent with a low-complexity algorithm was developed to obtain the near-optimal solution. However, the main draw-

back of [22]–[24], [26] was that they did not address the issue of obtaining the optimal policy in time-variant dynamic systems.

B. DEEP LEARNING-BASED METHODS

Currently, deep learning algorithms have been widely utilized in many aspects of life, such as natural language processing, gaming, IoT, computer vision, and speech recognition [27]. More specifically, reinforcement learning has been applied in a few recent studies of vehicular edge computing systems [20] to empirically cope with large-scale complex problems. For example, in [28], a novel decentralized resource allocation approach based on deep reinforcement learning was proposed for handling the latency constraints of vehicle-to-vehicle communications.

In [29], spectrum allocation and computing and storage resources were jointly studied for a multi-access edge computing-based vehicular network in which two architectures of mobile edge computing were formulated as multi-dimensional optimization problems. Then, an equivalent reinforcement learning form was derived from the above problems, and an efficient algorithm based on a deep deterministic policy gradient algorithm was developed to rapidly solve these problems and satisfy the quality-of-service requirements of vehicular applications.

Recently, Zhan *et al.* [30] and Luo *et al.* [31] investigated computational offloading and data scheduling for vehicular edge computing systems. In [30], Zhan *et al.* formulated computational offloading scheduling as an optimization problem whose objective was to minimize the long-term cost in terms of energy consumption and delay. Then, an equivalent Markov decision process was modelled to address the problem. Subsequentl, an efficient deep reinforcement learning-based algorithm was designed to solve this problem and obtain the optimal solution. In [31], communication and computational resources for data scheduling were jointly considered where the data bandwidth and computational offloading could be provided to the vehicles through a roadside unit across the road. Afterwards, computation, caching, communication, and collaborative computing were formulated as optimization problems to minimize the cost of data processing under latency constraints. Finally, deep reinforcement learning was utilized to derive an optimal data scheduling strategy. However, most of these works dealt with a one-level environment which colud cause congestion and lead to a longer processing time delay for computational offloading if a large number of vehicles offloaded their computational tasks to the same server. In addition, using conventional RL (e.g., Q-Learning) and other conventional methods is considered computationally prohibitive for large-scale environments.

III. SYSTEM MODEL

A. NETWORK MODEL

In this work, we consider a vehicle edge network (VEN) environment with a one-way road, a single cloud server, a set of $\mathbb{M}$ roadside units (RSUs) distributed across the road and a set of $\mathbb{N}$ vehicles moving along the road, as shown in Fig. 1. In addition, there is a vehicle edge computing server(VEC)[1] placed at each RSU which can provide with the storage and computational capabilities through a wireless channel. Furthermore, all the RSUs are connected with a cloud server through a backbone router that can manage and control the communication between them. We denote the set of vehicles as $\mathbb{N} = \{1, 2, \ldots, N\}$ in which each vehicle has an intensive computational task that can be completed by itself or be offloaded to and executed by the VEC server or the cloud server. In addition, we denote the set of RSUs as $\mathbb{M} = \{1, 2, \ldots, M\}$. Furthermore, we denote the set of computing servers as $\mathbb{K} = \{0, 1, 2, \ldots, M, M+1\}$, where 0 and $M+1$ denote the vehicle itself and the cloud server, respectively.

Let $x_{i,j} \in \{0, 1\}$ denotes the binary computational offloading decision for the computational task of vehicle i, which can be assigned to be executed at server j. More specifically, when $(x_{i,0} = 0)$, the computational task of vehicle i will be executed locally; when $(x_{i,M+1} = 1)$, the computational task of vehicle i will be offloaded and executed at the cloud server; when $(x_{i,j} = 1, \forall_j \in [1..M])$, the computational task of vehicle i will be offloaded and executed at one of the VEC servers. Overall, each computational task of vehicle i must be executed only one time by one of those servers (including server 0), while $\sum_{j=0}^{M+1} x_{i,j} = 1$.

In this paper, we assume that all vehicles are moving along the road and that their trajectories can be predicted using existing approaches [32], [33].

In the following subsections, the communication and computational models are presented within more detail, followed by the formulation of our optimization problem for multilevel vehicular edge-cloud computing networks.

B. COMMUNICATION MODEL

Let us begin with an introduction of the communication model in which our environment has M RSUs and N vehicles where each vehicle has an intensive computational task that needs to be completed by the vehicle itself or will be offloaded to and executed by the VEC server or cloud server. The computational task can be defined using a tuple $\{a_i, b_i, c_i\}$, where a_i describes the total data size of the computational task that can be offloaded (i.e., code and parameters), b_i is the total data size for the result that comes back from the server and c_i denotes the total number of CPU cycles required to accomplish the computational task. The values of a_i, b_i and c_i can be obtained through careful profiling of the task execution [34], [35].

In this paper, we have considered that an orthogonal frequency is used for multi-vehicle transmissions in the same cell to mitigate the intracellular interference for uplink transmission [36], [37]. Regarding the Shannon law, the uplink and downlink data rate for the communication between vehicle i

[1] In this study, RSUs and VEC server are used interchangeably.

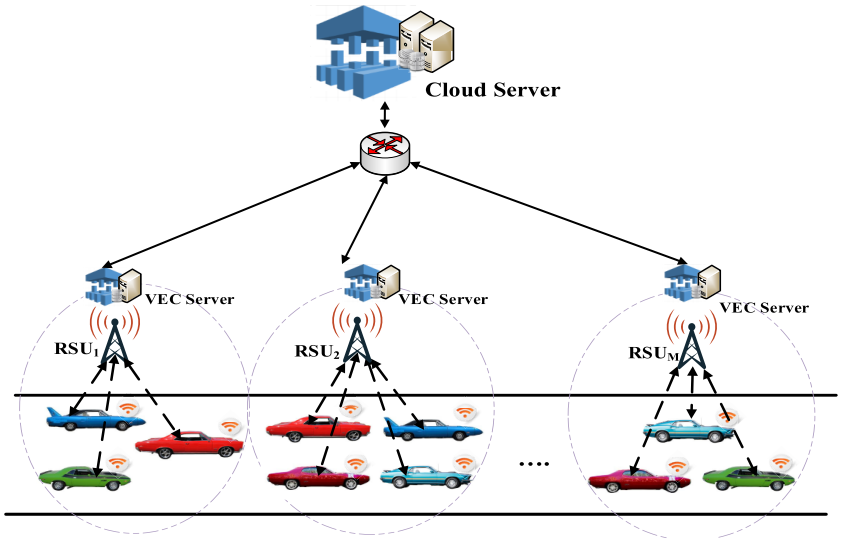


FIGURE 1. System model.

and connected RSU j is given as:

$$r_{i,j}^{U} = B_{i,j}^{U} log_2(1 + \frac{p_i^T G^2}{\omega B_{i,j}^{U}}) \quad (1)$$

$$r_{i,j}^{D} = B_{i,j}^{D} log_2(1 + \frac{p_j^T G^2}{\omega B_{i,j}^{D}}) \quad (2)$$

where $B_{i,j}^{U}$ and $B_{i,j}^{D}$ denote the uplink and downlink channel bandwidth, respectively, and p_i^T and p_j^T denote the transmission power of vehicle i and RSU j, respectively, whereas ω and G denote the density of noise power and the corresponding channel gain between the vehicle and the connected RSU, respectively.

Subsequently, as mentioned above, the computational task of vehicle i will be assigned for execution at the best place depending on the vehicle location one of the edge servers or the cloud server through the edge server with the maximum uplink and downlink transmission data rate and then the result will be returned. Based on that, where there is neither uplink nor downlink for local execution, we can assign it as $r_{i,0}^{U} = r_{i,0}^{D} = \infty$. In addition, the uplink and downlink for offloading and executing the computational task at the cloud server can be expressed as follows [40]:

$$r_{i,M+1}^{U} = \max_{j \in M} r_{i,j}^{U} \quad (3)$$

$$r_{i,M+1}^{D} = \max_{j \in M} r_{i,j}^{D} \quad (4)$$

Consequently, the total communication time and energy for executing all the computational tasks of vehicle i can be calculated as follows:

$$T_i^{Comm} = T_i^{U} + T_i^{D} + \nu x_{i,M+1} \quad (5)$$

$$E_i^{Comm} = p_i^T T_i^{U} + p_i^R T_i^{D} \quad (6)$$

where p_i^R and ν denote the reception power of vehicle i and the propagation delay for transferring the computational task between VEC nodes and the cloud server, respectively, which is a predetermined value regardless of the number of vehicles. T_i^{U} and T_i^{D} denote the time for uplink and downlink of the computation task, respectively, which can be expressed as:

$$T_i^{U} = \sum_{j=0}^{M+1} \frac{a_i}{r_{i,j}^{U}} x_{i,j} \quad (7)$$

$$T_i^{D} = \sum_{j=0}^{M+1} \frac{b_i}{r_{i,j}^{D}} x_{i,j} \quad (8)$$

C. COMPUTATIONAL MODEL

In the computational model, our system has N vehicles connected with M RSUs. We consider that each vehicle has an intensive computational task that needs to be executed either locally on the vehicle or remotely on one of the available VEC servers or on the cloud server. As a consequence, the computational time for local and remote execution are presented within more detail in the following subsections.

1) LOCAL EXECUTION APPROACH

For local execution, we consider that different vehicles may have different computational capabilities. In addition, all vehicles execute the computational task locally.

Thus, the execution time and energy consumption for executing all the computational tasks locally on each vehicle i can be calculated as follows:

$$T_i^{l} = \frac{c_i}{f_i^{l}} \quad (9)$$

$$E_i^{l} = \vartheta_i c_i \quad (10)$$

where ϑ_i is a coefficient denoting the energy consumed per CPU cycle and f_i^{l} denotes the computational capability (CPU cycles per second) of vehicle i.

2) REMOTE EXECUTION APPROACH

For remote execution, the computational task of vehicle i will be offloaded and executed at one of the connected VEC servers via a wireless channel or be offloaded and executed at the cloud server through one of the connected VEC servers and the result will be returned. Thus, the execution time for executing the computational task of vehicle i remotely at one of the available VEC servers or at the cloud server can be respectively expressed as follows:

$$T_i^e = \frac{c_i}{f_i^e} \tag{11}$$

$$T_i^c = \frac{c_i}{f_i^c} \tag{12}$$

where f_i^e and f_i^c denote the computational capability of the VEC and cloud server assigned to vehicle i.

In this paper, we assume that the computational resources of each VEC server are equally shared among all vehicles that are connected. In addition, all VEC servers have equal resources.

Consequently, based on Eqs.(9, 10, 11 and 12), the total computational time and energy for executing all the computational tasks of vehicle i can be calculated as follows:

$$T_i^{Comp} = T_i^l x_{i,0} + T_i^e x_{i,M+1} + \sum_{j=1}^{M} T_i^e x_{i,j} \tag{13}$$

$$E_i^{Comp} = E_i^l x_{i,0} + \sum_{j=1}^{M+1} \eta x_{i,j} \tag{14}$$

where η is a constant denoting the energy consumed by a vehicle in an idle case (i.e., waiting for the results from the remote server).

Regarding the previous subsections, where communication and computational models are considered, the overall consumption for processing the computational task of vehicle i at server j can be calculated as follows:

$$O_i = w_i^t (T_i^{Comm} + T_i^{Comp}) + w_i^e (E_i^{Comm} + E_i^{Comp}) \tag{15}$$

where w_i^e and $w_i^t \in [0, 1]$ denote the weighting parameters of execution time and energy consumption for vehicle i's decision making, respectively. For example, if $w_i^e = 0$ and $w_i^t = 1$, the running application is time-sensitive. Consequently, different values of w_i^e and w_i^t are set for different objectives.

D. PROBLEM FORMULATION

In this section, we consider the issue of achieving efficient computational offloading for multilevel vehicular edge-cloud computing networks. Regarding the above communication and computational models, the computational offloading problem is formulated as the following constrained optimization formulation problem:

$$\min_x \sum_{i=1}^{N} O_i$$

$$\sum_{j=0}^{M+1} x_{i,j} = 1, \quad C1$$

$$x_{i,j} \in \{0, 1\} \quad C2 \tag{16}$$

The objective function of the optimization problem aims to minimize the weighted sum of system consumption in terms of time and energy through the deployment of task offloading. Constraint C1 guarantees that each computational task of vehicle i must be executed only one time. Constraint C2 guarantees that the computational offloading decision variable is binary.

The solution for this problem can be obtained by finding the best values of the task offloading decision x^*. However, because x is a binary variable, the feasible set and objective function of the problem is not convex which is difficult to solve, especially for a large number of vehicles because of the well-known curse-of-dimensionality problem, where the size of the problem grows exponentially with the number of vehicles [38]–[40]. Therefore, the deep reinforcement learning method is used to find the optimal values efficiently instead of using conventional optimization methods.

IV. PROBLEM SOLUTION USING DEEP REINFORCEMENT LEARNING

In this section, an introduction of reinforcement learning is presented where the main key parts are formulated. Then, the deep reinforcement learning method is presented in detail to craft a computational offloading decision for multilevel vehicular edge-cloud computing networks.

A. REINFORCEMENT LEARNING

Reinforcement learning is a computational approach in which an agent interacts with an unknown dynamic environment and takes different actions to maximize the total amount of rewards, as shown in Fig. 2. More specifically, at each time t, the agent observes a state s_t from the state space S, and accordingly chooses an action a_t from the action space A that maps the agent from current state s_t to a new state s_{t+1} based on the policy $\pi(a_t|s_t)$. This policy can be determined using a Q-function in the Q-learning method or can be approximated in the deep learning method. Afterward, the environment transits to a new state s_{t+1} based on state transition probability $P(s_{t+1|s_t,a_t})$ and the agent obtains a reward r_t based on the reward function $R(s, a)$. Finally, this process is repeated until the agent reaches the terminal state in which the main goal is to maximize the expected cumulative rewards,

$$R_t = \Big[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \Big] \tag{17}$$

where $\gamma \in [0, 1]$ is the discount factor.

Regarding the main key parts of reinforcement learning, we consider a vehicular edge-cloud computing network in which the computational task' requirements are used to represent the state space S, which is defined as $s_t = \{(a_i, b_i, c_i)_t\} | i \in N$. In addition, the binary computational

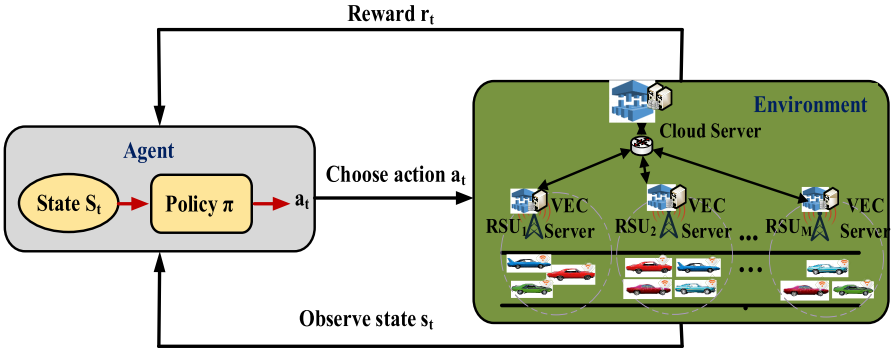


FIGURE 2. Reinforcement learning for vehicular edge-cloud computing networks.

offloading $x_{i,j}$ is used to represent the action space A which is defined as $a_t = \{(x_{i,j})_t\} | i \in N, j \in K$. Furthermore, at each time step t, the agent obtains a reward r_t based on a certain state s_t and after choosing an offloading action a_t following a policy $\pi(a_t|s_t)$. This process continues with the increase in the time index $t = 0, 1, 2, \ldots, T$. Moreover, the reward function should be related to the objective function of our optimization problem. Therefore, we aim to design a policy π that can efficiently generate an offloading action a_t for each system state s_t to minimize the expectation of the reward r_t, as:

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} r_t \tag{18}$$

where r_t denotes O_i in Eq. 15 which can be calculated using s_t and a_t.

B. DISTRIBUTED DEEP-Q-LEARNING

Deep Q learning is an effective reinforcement learning algorithm that combines reinforcement learning with deep neural networks [41]. Specifically, deep Q learning replaces the Q-table by a deep neural network (DNN) with parameter θ which tries to approximate the Q-values where θ is used to represent the trainable weights of the neural network. In addition, the system state is given as the input and the Q-value of all possible actions is generated as the output. Distributed deep Q learning has n DNNs that can work in parallel to find the optimal decision [42], [43], as shown in Fig. 3.

In this study, we propose a distributed deep Q learning algorithm to approximately minimize the expectation of the total reward, which is presented in Eq.(18), in which D parallel DNNs are used to generate the binary computational offloading decision. More specifically, at each time slot t, the algorithm takes the system state s_t as input and each DNN generates an offloading action a_t^d based on $f_{\theta_t^d} : s_t \rightarrow a_t^d$, in which $f_{\theta_t^d}$ is a parameterized function representing the d^{th} DNN with parameters θ_t^d and $d \in D = \{1, 2, \ldots, D\}$ is the index of the DNN. Then, the offloading action with the lowest reward is selected as the output action using $a_t^* = arg \min_{d \in D} Q(s_t, a_t^d)$. Algorithm 1 presents the process of the distributed deep Q learning algorithm for the computational offloading of multilevel vehicular edge-cloud computing networks.

Algorithm 1 Distributed Deep Q Learning Algorithm

Input: Requirements for computation task of vehicles s_t

Output: Computation offloading decision a_t^*
Initialize all the DNNs with random weights $\theta_t^d, d \in D$.
Initialize replay memory Y with Size P
for $t = 1, 2, \ldots, G$ **do**
  Input the same state s_t to each DNN.
  Generate an offloading action from the DNNs $\{a_t^d\} = f_{\theta_t^d}(s_t)$.
  Select the offloading action with the lowest reward $a_t^* = arg \min_{d \in D} Q(s_t, a_t^d)$.
  Save transition (s_t, a_t^*) into the memory Y
  Extract a sample random mini-batch of transitions from the memory Y.
  Train the DNNs.
end for

V. SIMULATION RESULTS AND DISCUSSION

In this section, the experimental setup for our simulation is introduced. Afterwards, a discussion on the extensive simulation results is provided to evaluate the performance of our proposed model.

A. EXPERIMENTAL SETUP

In our simulation, we consider a 100 m one-way road where there are five RSUs randomly distributed. Each RSU is connected with a VEC server that can provide computational capabilities. In addition, there are 10 arriving vehicles on the road in which each vehicle has a computational task that needs to be accomplished. Furthermore, there is a single cloud server that is connected with the VEC servers through a backbone router. The CPU frequencies of the cloud server, VEC servers and vehicles are set to 1×10^{12} cycles/s, 10×10^{9}

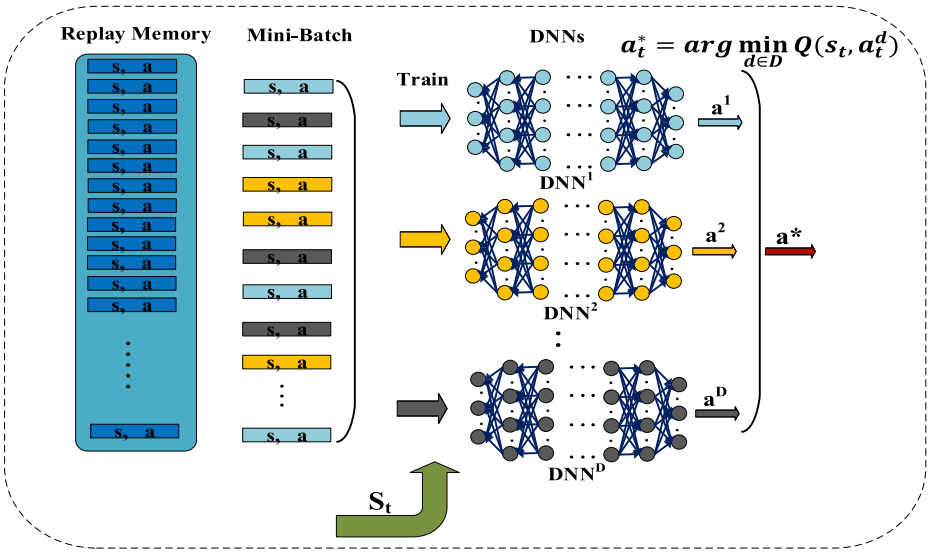


FIGURE 3. Distributed Deep-Q-Learning Architecture.

TABLE 1. Simulation parameters.

Parameter	Value
System Bandwidth	10 MHz
Background noise	-110dBm
Vehicle transmission and reception Power	24dBm
Input data size	$(10, 20)$MB Unif-Dist
CPU cycles to accomplish task	1900 cycles/bit
Computation capability of vehicle	0.6×10^9
VEC server capability	10×10^9
Cloud server capability	1×10^{12}
Cloud Propagation Delay	15 ms
Episode Size	20000
Replay Memory	1024
Mini-batch Size	32
Learning Rate	0.01

cycles/s and 0.6×10^9 cycles/s, respectively. The transmission and reception power of all the vehicles is 24 dBm. The system bandwidth of each RSU is 10 MHz. For each computational task, the input data size is uniformly distributed within the range $(10, 20)$ MB, the output data size is set to 0.2 of the input data size, and the required number of CPU cycles is set to 1900 cycles/byte. For the distributed deep learning algorithm, the episode, memory and mini-batch size are set to 20000, 1024 and 32, respectively. The learning rate and discount factor are set to 0.01 and 0.99, respectively. A Python-based simulator is used in our simulation, in which the computer is equipped with an Intel® Core(TM) i7-4770 CPU with a 3.4 GHz frequency and 16 GB RAM capacity running a Windows 10 Professional 64-bit platform. Additionally, TensorFlow and NumPy libraries are used to handle our algorithm in which two-hidden-layer DNNs are used with 120 and 80 neurons, respectively [44]. In addition, GEKKO Python-based package is used to solve our model using the conventional-based method [45].The other simulation parameters for communication and computation are summarized in Table 1.

B. EXPERIMENTAL RESULTS

1) CONVERGENCE PERFORMANCE

In this subsection, we study the convergence performance of our algorithm, in which different parameter values are applied and the appropriate parameter is selected for the next simulation. For better comparison, the reward ratio between the result of our algorithm and the result of the optimal approach[2] is used.

In Fig. 4, we show the effects of different numbers of DNNs on the convergence performance. It is observed from the figure that the convergence process of our algorithm accelerates as the number of DNNs increases. In addition, the reward ratio of only 3 DNNs can reach 0.96 after 2000 learning steps. However, with a small number of DNNs (e.g., DNNs=1), our algorithm falls into a local optimum and cannot converge well. Therefore, the number of DNNs is set to 3 in the following simulations.

In Fig. 5, we show the effects of different batch size values on the convergence performance, where the batch size is used to denote the number of experience samples that are trained at each interval. From the figure, the convergence rate for mini-batch size 32 is faster than 64, 128 and 512. This is because that the gradient descent direction becomes steeper as the mini-batch is smaller; therefore, the neural network weight will be updated faster. Thus, in the following

[2]All the binary offloading decision combinations ($2^{N(M+2)}$) are applied and then the optimal solution is obtained

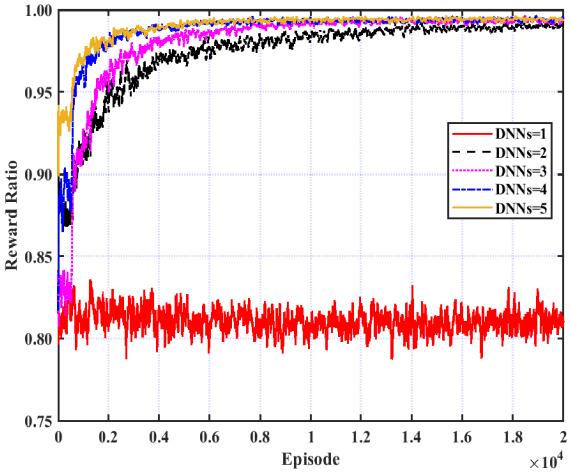


FIGURE 4. Convergence performance under different number of DNNs.

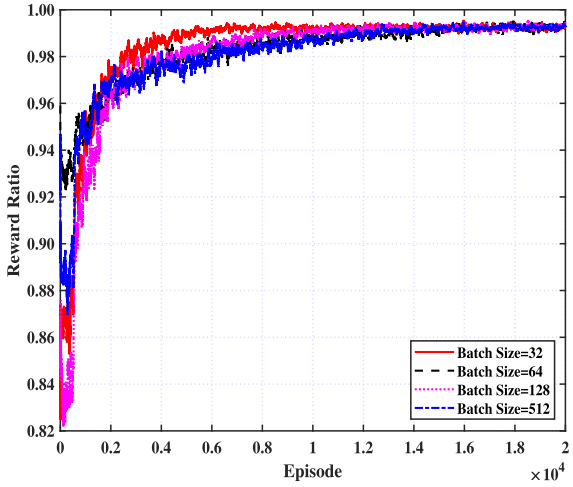


FIGURE 5. Convergence performance under different values of batch size.

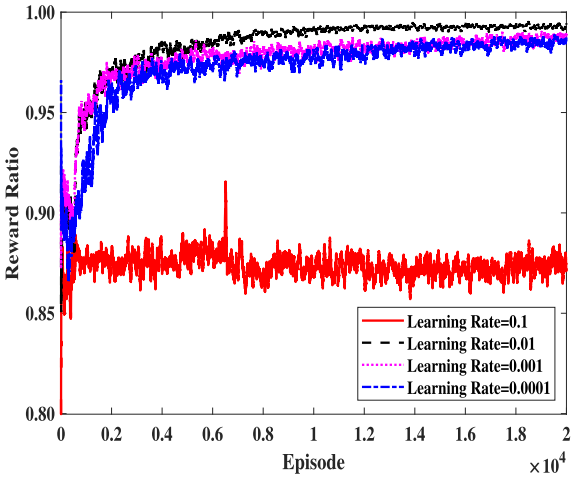


FIGURE 6. Convergence performance under different values of learning rate.

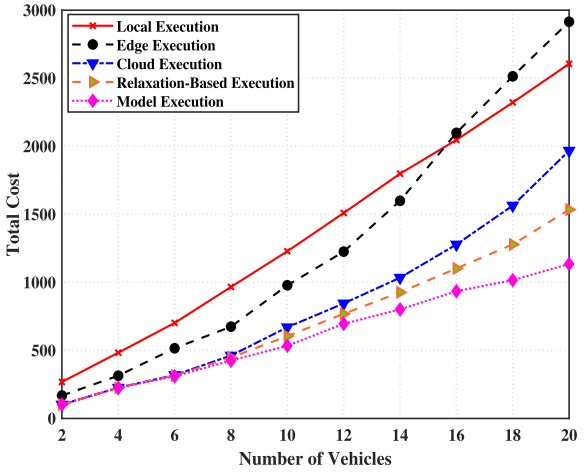


FIGURE 7. Total cost versus different number of vehicles.

simulations we set the batch size to 32 which is the most acceptable value.

In Fig. 6, we show the effects of different learning rate values on the convergence performance, where the learning rate can be used to adjust the weight update speed θ. From the figure, the convergence process for the learning rate 0.01 is shown to be faster than 0.001 and the convergence speed increases as the learning rate value increases. Nevertheless, the convergence process falls into a local optimum and cannot converge well as the learning rate (e.g., Learning Rate=0.1) becomes large. Thus, in the following simulations the learning rate value is set to 0.01 which is the most acceptable value.

2) SYSTEM PERFORMANCE

In this subsection, we evaluate the performance of our proposed model for multilevel vehicular edge-cloud computing networks with three different policies:

- **Local Execution:** There is no offloading. The computational tasks of all the vehicles will be executed locally on their resources, i.e., $x_{i,0} = 1$
- **Edge Execution:** All the vehicles offload their computational tasks to the connected RSUs for remote execution, i.e., $\sum_{j=1}^{M} x_{i,j} = 1$
- **Cloud Execution:** All the vehicles offload their computational tasks to the cloud server for remote execution, $x_{i,M+1} = 1$
- **Relaxation-Based Approach:** The computational tasks of all the vehicles will be executed either locally or remotely at one of the connected RSUs or cloud servers based on solving our optimization model using a relaxation-based method.

The total cost of executing the computational tasks versus a different number of arriving vehicles is shown in Fig. 7. The five curves in the figure represent the total cost for our proposed model with the addition of the other four policies mentioned above. It is observed from the figure that the total

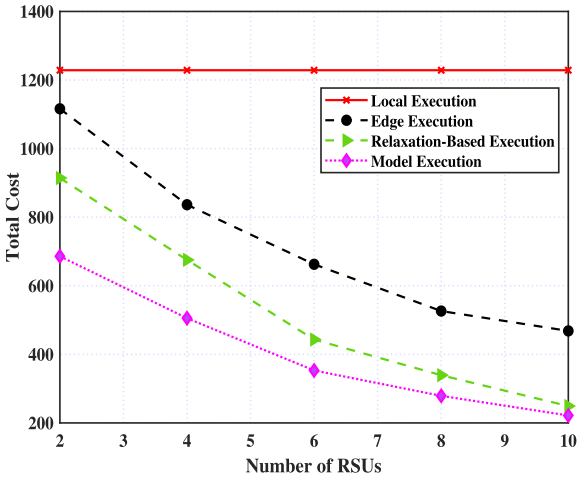


FIGURE 8. Total cost versus different number of RSUs.

cost for our model outperforms the other policies. In addition, we see that as the number of arriving vehicles increases(i.e., 16), the total cost for the edge execution policy exceeds the local execution policy. This is because the shared communication channels are overloaded, and the communication time increases. Moreover, the computational capability of the edge servers is not sufficient to serve too many vehicles, so selecting the offloaded tasks is an important issue under this scenario.

Finally, Fig. 8 presents the total cost for executing the computational tasks versus a different number of RSUs. From the figure, we can see that the local execution policy is not affected by the number of RSUs, whereas the sum cost for the other policies gradually decreases with increase in RSUs. In addition, the proposed model can maintain a lower cost in comparison with edge execution and relaxation-based execution. This is because the execution time decreases as the vehicles are allocated more resources, whereas the local execution does not use the RSU resources.

VI. CONCLUSION

In this paper, we jointly considered the computational offloading and resource allocation for a multilevel vehicular edge-cloud computing network in which an optimization problem is formulated whose objective is to minimize the vehicle's consumption in terms of time and energy. In addition, to practically derive the optimal solution to the formulated optimization problem, an equivalent reinforcement learning form is generated. Furthermore, we proposed a distributed deep learning algorithm to find the near-optimal computational offloading decisions in which a set of deep neural networks are used in parallel. Simulation results show that our algorithm can exhibit fast convergence and achieve better performance than the other benchmark solutions.

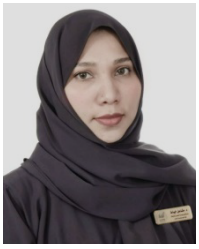
ACKNOWLEDGMENT

The authors, therefore, acknowledge the University of Jeddah for the technical support with thanks.

REFERENCES

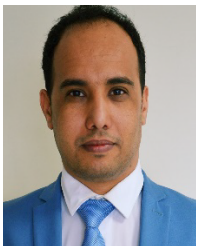
[1] J. Wu, C. Yuen, M. Wang, and J. Chen, "Content-aware concurrent multipath transfer for high-definition video streaming over heterogeneous wireless networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 3, pp. 710–723, Mar. 2016.
[2] M. Khayyat, A. Alshahrani, S. Alharbi, I. Elgendy, A. Paramonov, and A. Koucheryavy, "Multilevel service-provisioning-based autonomous vehicle applications," *Sustainability*, vol. 12, no. 6, p. 2497, Mar. 2020.
[3] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.
[4] D. Xu, Y. Li, X. Chen, J. Li, P. Hui, S. Chen, and J. Crowcroft, "A survey of opportunistic offloading," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2198–2236, 3rd Quart., 2018.
[5] H. Zhou, H. Wang, X. Chen, X. Li, and S. Xu, "Data offloading techniques through vehicular ad hoc networks: A survey," *IEEE Access*, vol. 6, pp. 65250–65259, 2018.
[6] I. A. Elgendy, M. Elkawkagy, and A. Keshk, "An efficient framework to improve the performance of mobile applications," *Int. J. Digit. Content Technol. Appl.*, vol. 9, no. 5, pp. 43–54, 2015.
[7] D. Elminaam, F. Elanezi, and K. Hosny, "An efficient framework for mobile cloud computing," in *Proc. 32th Int. Bus. Inf. Manage. Assoc. (IBIMA)*, 2018, pp. 5783–5796.
[8] I. A. Elgendy, M. El-kawkagy, and A. Keshk, "Improving the performance of mobile applications using cloud computing," in *Proc. 9th Int. Conf. Informat. Syst.*, Dec. 2014, p. PDC-109.
[9] A. Boukerche and R. E. De Grande, "Vehicular cloud computing: Architectures, applications, and mobility," *Comput. Netw.*, vol. 135, pp. 171–189, Apr. 2018.
[10] I. Elgendy, W. Zhang, C. Liu, and C. Hsu, "An efficient and secured framework for mobile cloud computing," *IEEE Trans. Cloud Comput.*, early access, Jun. 18, 2018, doi: 10.1109/TCC.2018.2847347.
[11] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive offloading," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 36–44, Jun. 2017.
[12] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2092–2104, Feb. 2020.
[13] R. A. Dziyauddin, D. Niyato, N. C. Luong, M. A. M. Izhar, M. Hadhari, and S. Daud, "Computation offloading and content caching delivery in vehicular edge computing: A survey," 2019, *arXiv:1912.07803*. [Online]. Available: http://arxiv.org/abs/1912.07803
[14] Y. Cao, T. Jiang, O. Kaiwartya, H. Sun, H. Zhou, and R. Wang, "Toward pre-empted EV charging recommendation through V2V-based reservation system," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Jun. 11, 2019, doi: 10.1109/TSMC.2019.2917149.
[15] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4377–4387, Jun. 2019.
[16] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," 2019, *arXiv:1908.06849*. [Online]. Available: http://arxiv.org/abs/1908.06849
[17] M. Liwang, S. Dai, Z. Gao, Y. Tang, and H. Dai, "A truthful reverse-auction mechanism for computation offloading in cloud-enabled vehicular network," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4214–4227, Jun. 2019.
[18] S. Raza, S. Wang, M. Ahmed, and M. R. Anwar, "A survey on vehicular edge computing: Architecture, applications, technical issues, and future directions," *Wireless Commun. Mobile Comput.*, vol. 2019, pp. 1–19, Feb. 2019.
[19] J. Zhang, H. Guo, and J. Liu, "A reinforcement learning based task offloading scheme for vehicular edge computing network," in *Proc. Int. Conf. Artif. Intell. Commun. Netw.* Cham, Switzerland: Springer, 2019, pp. 438–449.
[20] G. Qiao, S. Leng, S. Maharjan, Y. Zhang, and N. Ansari, "Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 247–257, Jan. 2020.
[21] Y. Wang, P. Lang, D. Tian, J. Zhou, X. Duan, Y. Cao, and D. Zhao, "A game-based computation offloading method in vehicular multiaccess edge computing networks," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4987–4996, Jun. 2020.
[22] J. Wang, D. Feng, S. Zhang, J. Tang, and T. Q. S. Quek, "Computation offloading for mobile edge computing enabled vehicular networks," *IEEE Access*, vol. 7, pp. 62624–62632, 2019.

[23] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.
[24] X. Xu, Y. Xue, L. Qi, Y. Yuan, X. Zhang, T. Umer, and S. Wan, "An edge computing-enabled computation offloading method with privacy preservation for Internet of connected vehicles," *Future Gener. Comput. Syst.*, vol. 96, pp. 89–100, Jul. 2019.
[25] H. Zhou, X. Chen, S. He, J. Chen, and J. Wu, "DRAIM: A novel delay-constraint and reverse auction-based incentive mechanism for WiFi offloading," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 4, pp. 711–722, Apr. 2020.
[26] J. Sun, Q. Gu, T. Zheng, P. Dong, A. Valera, and Y. Qin, "Joint optimization of computation offloading and task scheduling in vehicular edge computing networks," *IEEE Access*, vol. 8, pp. 10466–10477, 2020.
[27] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, Aug. 2019.
[28] H. Ye, G. Y. Li, and B.-H.-F. Juang, "Deep reinforcement learning based resource allocation for V2V communications," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3163–3173, Apr. 2019.
[29] H. Peng and X. S. Shen, "Deep reinforcement learning based resource management for multi-access edge computing in vehicular networks," *IEEE Trans. Netw. Sci. Eng.*, early access, Mar. 6, 2020, doi: 10.1109/TNSE.2020.2978856.
[30] W. Zhan, C. Luo, J. Wang, C. Wang, G. Min, H. Duan, and Q. Zhu, "Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5449–5465, Jun. 2020.
[31] Q. Luo, C. Li, T. H. Luan, and W. Shi, "Collaborative data scheduling for vehicular edge computing via deep reinforcement learning," *IEEE Internet Things J.*, early access, Mar. 26, 2020, doi: 10.1109/JIOT.2020.2983660.
[32] Z. Zhao, L. Guardalben, M. Karimzadeh, J. Silva, T. Braun, and S. Sargento, "Mobility prediction-assisted Over-the-Top edge prefetching for hierarchical VANETs," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 8, pp. 1786–1801, Aug. 2018.
[33] L. Yao, A. Chen, J. Deng, J. Wang, and G. Wu, "A cooperative caching scheme based on mobility prediction in vehicular content centric networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 5435–5444, Jun. 2018.
[34] X. Lyu and H. Tian, "Adaptive receding horizon offloading strategy under dynamic environment," *IEEE Commun. Lett.*, vol. 20, no. 5, pp. 878–881, May 2016.
[35] F. Liu, Z. Huang, and L. Wang, "Energy-efficient collaborative task computation offloading in cloud-assisted edge computing for IoT sensors," *Sensors*, vol. 19, no. 5, p. 1105, Mar. 2019.
[36] S. Deb and P. Monogioudis, "Learning-based uplink interference management in 4G LTE cellular systems," *IEEE/ACM Trans. Netw.*, vol. 23, no. 2, pp. 398–411, Apr. 2015.
[37] I. A. Elgendy, W. Zhang, Y.-C. Tian, and K. Li, "Resource allocation and computation offloading with data security for mobile edge computing," *Future Gener. Comput. Syst.*, vol. 100, pp. 531–541, Nov. 2019.
[38] D. Fooladivanda and C. Rosenberg, "Joint resource allocation and user association for heterogeneous wireless cellular networks," *IEEE Trans. Wireless Commun.*, vol. 12, no. 1, pp. 248–257, Jan. 2013.
[39] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu, "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing," *Digit. Commun. Netw.*, vol. 5, no. 1, pp. 10–17, Feb. 2019.
[40] L. Huang, X. Feng, L. Zhang, L. Qian, and Y. Wu, "Multi-server multi-user multi-task computation offloading for mobile edge computing networks," *Sensors*, vol. 19, no. 6, p. 1446, Mar. 2019.
[41] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, MA, USA: MIT Press, 2018.
[42] H. Yi Ong, K. Chavez, and A. Hong, "Distributed deep Q-Learning," 2015, *arXiv:1508.04186*. [Online]. Available: http://arxiv.org/abs/1508.04186
[43] L. Huang, X. Feng, A. Feng, Y. Huang, and L. P. Qian, "Distributed deep learning-based offloading for mobile edge computing networks," *Mobile Netw. Appl.*, vol. 23, pp. 1–8, Nov. 2018.
[44] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," 2016, *arXiv:1603.04467*. [Online]. Available: http://arxiv.org/abs/1603.04467
[45] L. Beal, D. Hill, R. Martin, and J. Hedengren, "GEKKO optimization suite," *Processes*, vol. 6, no. 8, p. 106, Jul. 2018.



MASHAEL KHAYYAT received the bachelor's degree (Hons.) in computer science degree from King Abdul-Aziz University, in 2004, the master's degree in applied information systems (AIS) from the Arab Academy for Science and Technology and Maritime Transport, Alexandria, Egypt, the second master's degree in technology management (MTM) from the University of New South Wales (UNSW), Sydney, Australia, and the Ph.D. degree in computer science and statistics from Trinity College Dublin (TCD), Dublin, Ireland, in 2017. She has been a Supervisor at the Department of Computer and Network Engineering and an Assistant Professor at the Information Systems and Technology Department, College of Computer Science and Engineering, University of Jeddah, since 2017. Prior to that, she worked at the Information Systems Department, King Abdul-Aziz University, as an Assistant Professor. She received international and distinguished research grants from the University of Jeddah.

IBRAHIM A. ELGENDY received the M.Sc. degree from the Computer Science Department, Faculty of Computers and Information, Menoufia University, Egypt, in 2016. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China. He has been an Assistant Lecturer at the Faculty of Computers and Information, Menoufia University, Egypt, since December 2011. His research interests include cloud computing, mobile edge computing, and distributed computing.



AMMAR MUTHANNA (Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees from the St. Petersburg State University of Telecommunications, Russia, in 2009, 2011, and 2016, respectively. From 2012 to 2013, he took part in the Erasmus Student Program at the Faculty of Electrical Engineering, University of Ljubljana. He is currently an Associate Professor with the Department of Telecommunication Networks, and also the Head of the SDN Laboratory, St. Petersburg State University of Telecommunications. He has published more than 60 scientific articles. He acted as a reviewer for many international and high ranked journals. He is an editor in the editorial boards of several international scientific journals. His main areas of research include the IoT, SDN, and MEC.

ABDULLAH S. ALSHAHRANI graduated in computer science from King Khalid University, in 2008. He received the M.Sc. degree in computer Science from La Trobe University, Melbourne, Australia, in 2010, and the Ph.D. degree from The Catholic University of America, USA, in 2018. He is currently an Assistant Professor at the Department of Computer Science and Artificial Intelligence, College of Computer Science and Engineering, University of Jeddah, Saudi Arabia. His research interests include wireless sensor networks, network security, parallel computing, smart homes systems, the IoTs, and data science.

SOLTAN ALHARBI received the B.S. and M.S. degrees in computer engineering from the Florida Institute of Technology, Melbourne, USA, and the Ph.D. degree in electrical and computer engineering from University of Victoria, Victoria, Canada.

He is currently an Assistant Professor and the Chairman of the Department of Computer and Network Engineering, College of Computer Sciences and Engineering, University of Jeddah, Jeddah, Saudi Arabia. His research interests include digital forensics investigation (both reactive and proactive), computer vision, network security, and information security. He is currently working on implementing a proactive system that would complement the current practice of reactive investigation. He is a member of the IEEE Computer Society and High Technology Crime Investigation Association (HTCIA).

ANDREY KOUCHERYAVY graduated from the Leningrad University of Telecommunications, in 1974.

He joined the Telecommunication Research Institute LONIIS, where he worked till October 2003 (from 1986 to 2003 as the First Deputy Director). Since 1998, he has been a Professor at the Bonch-Bruevich St. Petersburg State University of Telecommunications (SUT), where he became a Chair Professor of the ''Telecommunication Networks and Data Transmission'' department, in 2011. He was an Advisor of the Central Science Research Telecommunication Institute (ZNIIS), from 2003 to 2010. He was a Co-Founder of the International Teletraffic Seminar in 1993, 1995, 1998, and 2002, a Founder of the model network for digital networks at LONIIS, in 1997, a Co-Founder of the model network for packet networks at ZNIIS, in 2004, a Co-Founder of the Internet of Things Laboratory, in 2012, and a Quality of Experience and IPTV Laboratory, SUT, in 2014. He is a Chair of the Scientific School on Teletraffic Theory in LONIIS, from 1990 to 2003, has been the Founder and Scientific School Chair of the ''Internet of Things and self-organizing networks'' in SUT, since 2010.

Dr. Koucheryavy was a Host and Technical Program Committees Member of the ''Kaleidoscope 2014'' at SUT. He was also an Honorary Member of Popov's Society, in 2002. He was a Steering Committee Member of the IEEE technically co-sponsored series of conferences ICACT and NEW2AN. He was the Vice-Chairman of SG11 ITU-T, from 2005 to 2008 and 2009 to 2012. He was also the Chairman of WP3/WP4 SG11, from 2006 to 2012, the Vice-Chairman of WP4 SG11, from 2015 to 2016, the Chairman of SG11 in Study period 2017–2020. He was a Co-founder of the International Testing Center for new telecommunications technologies at ZNIIS, under ITU-D competence.