Optimal Auction for Delay and Energy Constrained Task Offloading in Mobile Edge Computing

Farshad Mashhadi[a], Sergio A. Salinas Monroy[a,*], Arash Bozorgchenani[b], Daniele Tarchi[b]

[a]Department of Electrical Engineering and Computer Science, Wichita State University, Wichita, USA
[b]Department of Electrical, Electronic and Information Engineering, University of Bologna, Bologna, Italy

Abstract

Mobile edge computing has emerged as a promising paradigm to complement the computing and energy resources of mobile devices. In this computing paradigm, mobile devices offload their computing tasks to nearby edge servers, which can potentially reduce their energy consumption and task completion delay. In exchange for processing the computing tasks, edge servers expect to receive a payment that covers their operating costs and allows them to make a profit. Unfortunately, existing works either ignore the payments to the edge servers, or ignore the task processing delay and energy consumption of the mobile devices. To bridge this gap, we propose an auction to allocate edge servers to mobile devices that is executed by a pair of deep neural networks. Our proposed auction maximizes the profit of the edge servers, and satisfies the task processing delay and energy consumption constraints of the mobile devices. The proposed deep neural networks also guarantee that the mobile devices are unable to unfairly affect the results of the auctions. Our extensive simulations show that our proposed auction mechanism increases the profit of the edge servers by at least 50% compared to randomized auctions, and satisfies the task processing delay and energy consumption constraints of mobile devices.

Keywords: Mobile edge computing, Deep learning, Auction, Delay and energy sensitive tasks

1. Introduction

Mobile applications are taking advantage of the ever-increasing number of mobile devices in our society to revolutionize many fields [1]. For example, in healthcare, mobile devices can analyze medical data to provide timely diagnoses for various diseases at the point of care [2, 3]. In Industry 4.0, factory managers can visualize in real-time their factory operations with mobile devices, including augmented reality headsets [4, 5, 6]. However, the limited computation capacity and battery life-time of mobile devices prevent the full implementation of computation-intensive mobile applications.

To enable computation-intensive mobile applications to run within an acceptable amount of time without depleting the batteries of the mobile devices, companies and researchers have proposed mobile edge computing (MEC). In MEC, mobile devices offload their computing tasks to edge servers, which have large computing resources and are strategically placed near to the mobile devices, e.g., cellular base stations. Compared to running applications entirely on the mobile devices, running applications on a MEC network drastically reduces the computing task completion delay and energy consumption of the mobile devices [7]. Hence, MEC can enable sophisticated applications with large computing demands to run smoothly on mobile devices.

However, it is challenging to economically incentivize edge servers to complete the tasks of the mobile devices while ensuring that mobile devices receive the results of the computing tasks within an acceptable amount of time. On the one hand, edge servers aim to maximize their profit by servicing the tasks from mobile devices that are willing to pay the highest prices for the computing services. On the other hand, mobile devices seek to offload their task to an edge server that can reduce their task completion delay and energy consumption at the lowest price. Therefore, we need to match the mobile devices' tasks to edge computing servers in such a way that servers maximize their profit, the mobile devices have a low payment, and the task processing and mobile device energy consumption constraints of the applications are satisfied by the MEC network.

There are some works on cloud computing that could potentially be used by mobile devices to run computing-intensive applications on remote servers [8]. However, since the cloud is usually located far away from the mobile devices, offloading tasks to the cloud results in very large transmission delays that are unacceptable for run-

*Corresponding author
Email address: sergio.salinasmonroy@wichita.edu (Sergio A. Salinas Monroy)
[1]F. Mashhadi and S. Salinas are with the Department of Electrical Engineering and Computer Science, Wichita State University, USA.
[2]A. Bozorgchenani and D. Tarchi are with the Department of Electrical, Electronic and Information Engineering, University of Bologna, Italy.

ning mobile applications, which have stringent time delay constraints [9, 10, 11].

More recently, researchers have proposed various MEC task offloading approaches that consider the task processing delay and energy consumption of the mobile devices. Specifically, Liu et al. [12] propose a task offloading scheme for mobile devices that minimizes their task completion delay when offloading their tasks to the edge servers. Zhang et al. [13] propose an approach to offload the tasks of the mobile devices in such a way that their energy consumption is minimized. Dai et al. [14] propose a task offloading framework that minimizes the energy consumption of both the edge servers and the mobile devices, while satisfying tasks processing delay constraints. Yang et al. [15] designs an offloading algorithm that minimizes the energy consumption of the mobile devices in the presence of UAV edge servers. Zhang et. al [16] propose a task offloading scheme that jointly minimizes the task processing delay and energy consumption of the mobile devices. In a recent work, Bozorgi et. al [17] proposed an algorithm that jointly minimizes the processing delay and energy consumption of mobile devices, in an MEC network where tasks are divided into smaller parts and can be processed in parallel by both mobile devices and edge servers. However, these works assume that the edge servers will process the tasks from any mobile device without receiving a payment in return, which makes them unrealistic.

Some works propose to compensate the edge servers in return for processing the computing tasks of the mobile devices. Jiao et al. [18] consider a MEC network where a set of blockchain miners outsource their computations to a single edge server. To compensate the edge server, the system runs an auction that allocates the server's computing resources to the miners that are willing to pay the highest access fee. The auction mechanism collects the payments from the winning miners and pays the edge server for its services. Bahreini et al. [19] propose an envy-free auction mechanism that allocates a set of virtual machines at the edge servers to the mobile devices. Li et al. [20] propose a similar auction mechanism for a MEC network where mobile devices outsource generic computing tasks.

To improve the utility that the edge servers receive, Luong et al. [21] propose a deep learning auction mechanism. Their mechanism allocates the computing resources from the edge server to the mobile devices that are willing to pay the highest price, but it does so in a way that the utility of the edge server increases significantly without sacrificing the auction integrity properties, i.e., truthfulness, individual rationality, etc. Although these works compensate edge servers for their services, they only consider one edge server, which is unrealistic in MEC. In addition, they ignore the delay and energy constraints of the mobile devices, which can result in very long waits and quick battery drainage.

Kiani et. al [22] propose an optimization model that compensates the edge servers and satisfies the constraints of the mobile devices. Their model aims to maximize the compensation of the edge server by allocating the resources to mobile devices who are willing to pay the highest price, while minimizing the task processing delay experienced by the mobile devices. Unfortunately this work ignores the possibility that mobile devices maliciously manipulate their bids to obtain an unfair advantage. For example, by exaggerating their valuation of the edge server services, mobile devices can constantly out bid other devices and prevent them from accessing the edge servers. Ensuring that the best strategy for mobile devices is to report their true valuation in their bids is a key to incentivize both edge servers and mobile devices to participate in the auction.

To bridge this gap, we propose an auction mechanism that is executed by a pair of deep neural networks. To the best of our knowledge, this is the first truthful auction mechanism for MEC that can both maximize the profit of multiple edge servers and satisfy the task processing delay and energy consumption constraints of the mobile devices. Specifically, the proposed neural networks take the bids from the mobile devices as input, and then output task offloading decisions and payments for the mobile devices. The auction also guarantees that mobile devices are unable to unfairly affect the results of the auctions by submitting untruthful bids, and that both the edge servers and the nodes are economically incentivized to participate in the auction. We extensively evaluate our proposed auction and see that it can obtain up to a three-fold increase in profit for the edge servers compared to existing auction mechanisms. Our simulation results also show that our proposed auction mechanism can successfully satisfy the task processing delay and mobile device energy consumption constraints of the mobile devices, which existing MEC auctions are unable to do.

We summarize our contributions as follows:

- We design a deep-learning truthful auction mechanism that allocates the computing resources of multiple MEC edge servers to the mobile devices while ensuring that the mobile devices energy consumption and task completion delay constraints are satisfied. To the best of our knowledge, this is the first truthful mechanism for multiple edge servers that can satisfy the mobile device constraints.

- We design new penalty functions for the task computing delay and energy consumption constraints to guide the neural networks towards feasible solutions.

- The utility that edge servers obtain through our proposed computing resource allocation is up to three times larger to the that of existing randomized auctions. When compared to other deep learning mechanisms, our approach obtains a similar utility for the edge servers, and, remarkably, it is able to satisfy the task completion delays and energy consumption constraints of the mobile devices, which can provide a much higher quality of experience to the mobile devices.

The rest of the paper is organized as follows. Section 2 describes our considered system model. Section 3 describes our auction model. In Section 4, we introduce our deep learning auction mechanism that finds allocation decisions and payments. We show our numerical results in Section 5, and give concluding remarks in Section 6.

2. System Model

We consider a set of resource-limited mobile devices $\mathcal{M} = \{1, 2, ..., M\}$ who offload delay-sensitive computing tasks to one of the resource-rich edge servers in the set $\mathcal{N} = \{1, 2, ..., N\}$. The mobile devices are powered by batteries and have limited computing resources while the edge computing servers are directly connected to the power grid and have abundant computing resources. The edge servers offer their task processing services to the mobile devices in exchange for a fee. A central system operator sells and coordinates access to the edge servers, e.g., a single company may own all the edge servers, or multiple edge servers can form a coalition to offer task offloading services. Fig. 1 shows our considered network.

Suppose the mobile device $m \in \mathcal{M}$ offloads a computing task to the edge server $n \in \mathcal{N}$. Then, the total task completion time depends on the computing time at the edge server n, the task upload time and the result download time, i.e.,

$$D_{off}^{m,n} = D_{tx}^{m,n} + D_{rx}^{m,n} + D_{comp}^{m,n} \quad \forall m \in \mathcal{M}, \forall n \in \mathcal{N} \quad (1)$$

where $D_{tx}^{m,n}$, $D_{rx}^{m,n}$, $D_{comp}^{m,n}$ denote the time to upload the task, download the result, and compute the task at an edge server, respectively. The task upload time is given by

$$D_{tx}^{m,n} = \frac{t^m}{d^{m,n}},$$

where $d^{m,n}$ is the data rate of the the mobile device m's transmission to the edge server n in bits per second, and t^m is the size of the computing task in bits. The result download time is given by

$$D_{rx}^{m,n} = \frac{r^m}{d^{n,m}},$$

where r^m is the size of the result in bits and $d^{n,m}$ is the mobile device's download rate from edge server n. The task computing delay at edge server n is

$$D_{comp}^{m,n} = \frac{t^m o^m}{f^n},$$

where o^m is the number of floating point operations (flops) required to process one bit of the task, and f^n is the processing rate of edge server n in flops per second.

Since the computing tasks are delay sensitive, we require that the time it takes the mobile device to receive the results from an edge computing server should be less than the time it takes to locally complete the task, i.e.,

$$D_{off}^{m,n} \leq D_{local}^{m} \qquad \forall m \in \mathcal{M}. \qquad (2)$$

where D_{local}^{m} is the local task completion delay of mobile device m and is given by

$$D_{local}^{m} = \frac{t^m o^m}{f^m},$$

where f^m is the processing rate of the mobile device m in flops per second.

Moreover, task offloading requires mobile devices to consume energy transmitting the task, receiving the results, and maintaining an idle state while waiting to download the results. Consequently, the total energy consumption for the mobile device m to offload its task to edge server n is given by

$$E_{off}^{m,n} = E_{tx}^{m,n} + E_{id}^{m,n} + E_{rx}^{m,n} \quad \forall m \in \mathcal{M}, \forall n \in \mathcal{N} \quad (3)$$

where $E_{tx}^{m,n}$, $E_{id}^{m,n}$, and $E_{rx}^{m,n}$ represent the transmission, idling, and receiving energy of mobile device m. The task transmission energy consumption is given by

$$E_{tx}^{m,n} = t^m e_{tx}^{m,n},$$

where $e_{tx}^{m,n}$ is the energy consumed by mobile device m while transmitting one bit of the task to edge server n. The energy consumed by mobile device m while receiving a task is given by

$$E_{rx}^{m,n} = r^m e_{rx}^{m,n}$$

where $e_{rx}^{m,n}$ is the per bit receiving energy consumption. The amount of energy that mobile device m spends during the idle time is given by

$$E_{id}^{m,n} = D_{comp}^{m,n} \cdot e_{id}^{m},$$

where e_{id}^{m} is the per second idling energy consumption.

To ensure mobile device m reduces its energy consumption when offloading tasks, the amount of energy needed to offload the task to edge computing server n should be less than the amount of energy needed to locally compute the task, i.e.,

$$E_{off}^{m,n} \leq E_{local}^{m} \qquad \forall m \in \mathcal{M}, \qquad (4)$$

The energy consumption of mobile device m to locally compute its task E_{local}^{m} is given by

$$E_{local}^{m} = D_{local}^{m} e_{pr}^{m}.$$

where e_{pr}^{m} is mobile device m's per second processing energy consumption.

Moreover, we assume the computing tasks from the mobile devices have a significant sequential component, and cannot be easily computed in parallel by multiple edge servers. Thus, in addition to satisfying the task completion delay constraints and mobile device energy consumption constraints in (2) and (4), the system operator needs to ensure mobile devices offload their task to at most one edge server. We also assume that edge server n employs all of its computing resources to complete the task from mobile device m, and thus accepts at most one task at a time. The edge servers start to process the offloaded tasks immediately after they receive them.

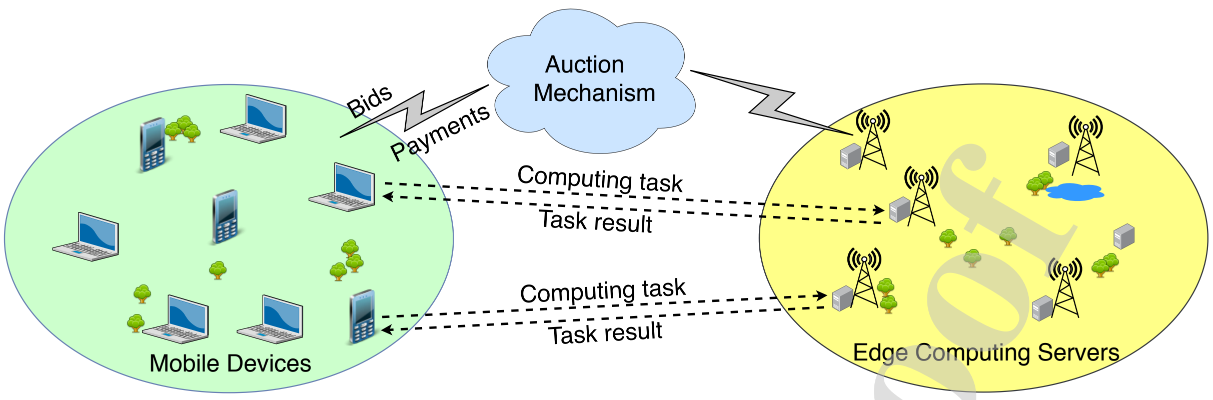


Figure 1: A mobile edge computing network architecture.

3. An Auction Model for Task Offloading

The system operator conducts an auction to assign the tasks of the mobile devices to the edge servers in such a way that the edge servers' profits are maximized and the task processing delays and energy constraints of the mobile devices are satisfied. Specifically, each mobile device $m \in \mathcal{M}$ has a private bid set $V_m = \{v_{m,1}, v_{m,2}, ..., v_{m,N}\}$ that specifies its valuation for offloading its computing task to edge server n (for all $n \in \mathcal{N}$).

Mobile device m chooses its private valuations $v_{m,n}$'s to be proportional to the number of operations o^m needed to complete its task, and inversely proportional to both the task processing delay $D^{m,n}_{off}$ and the energy needed to offload the task to the edge server n, i.e., $E^{m,n}_{off}$. We assume the number of operations needed to complete the mobile devices' tasks is a random variable with probability distribution F_m that is known to the system operator, but not to the other mobile devices or edge servers. Consequently, the private bid valuations $v_{m,n}$'s are also random variables. We denote the space of private bids V_m's by $\mathcal{V}_m$. We denote the set of bids from the mobile devices by $\mathbf{V} = \{V_1, V_2, \ldots, V_M\}$, and the space of mobile device bid sets by $\mathcal{V}$. We use F to denote the probability distribution of the set of bids $\mathbf{V} \in \mathcal{V}$. Note that the distributions F_m are independent of each other. We also define the set of bids without the bid from mobile m by $\mathbf{V}_{-m} = \{V_1, ..., V_{m-1}, V_{m+1}, ...V_M\}$.

The auction starts with mobile devices reporting (perhaps untruthfully) their bid sets to the operator. The reported bid sets from mobile device m is denoted by V'_m, and the set of reported bids from all mobile devices is denoted by $\mathbf{V}' = \{V'_1, V'_2, ..., V'_M\}$. The operator then finds a randomized allocation rule $a : \mathcal{V} \to [0,1]^{M\times N}$. The allocation rule a maps the vector of reported mobile device bids $\mathbf{V}' \in \mathcal{V}$ to a matrix of edge server allocation probabilities $\mathbf{A}(\mathbf{V}') \in [0,1]^{M\times N}$. In matrix $\mathbf{A}(\mathbf{V}')$, there is a row for each mobile device, and a column for each edge server in the network. Thus, the element in the mth row and nth column of $\mathbf{A}(\mathbf{V}')$, denoted by $a_{m,n}(\mathbf{V}')$, is the probability that mobile device m's task is assigned to edge computing server n. Since a mobile device can offload its task to at most one edge computing server, and an edge server can only process the task from at most one mobile device at a time, we require that

$$\sum_{n=1}^{N} a_{m,n}(\mathbf{V}') \le 1, \ \forall m \in \mathcal{M},$$

and

$$\sum_{m=1}^{M} a_{m,n}(\mathbf{V}') \le 1, \ \forall n \in \mathcal{N}.$$

Note that some edge servers may remain unallocated and some mobile devices may be left without having an edge server allocated to them [3].

The auction mechanism also finds a payment rule $p : \mathcal{V} \to \mathbb{R}^{M\times 1}_{\ge 0}$ that maps the reported bids from mobile devices $\mathbf{V}' \in \mathcal{V}$ to the edge servers' expected payments $p_m(\mathbf{V}')$ (for all mobile devices $m \in \mathcal{M}$).

We define the utility of both the mobile devices and the overall utility of the edge servers as follows:

$$u_m(V_m, \mathbf{V}') = \begin{cases} \sum_{i=n}^{N} v_{m,n} a_{m,n}(\mathbf{V}') - p_m(\mathbf{V}') & \text{if (2) and (4) hold} \\ -\infty & \text{otherwise} \end{cases}$$

Note that mobile devices prefer to locally complete their tasks than to experience a longer task processing delay or a higher energy consumption during offloading.

The utility of the edge servers is given by

$$U(\mathbf{V}') = \sum_{m\in\mathcal{M}} p_m(\mathbf{V}') - \sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{N}} c^n o^m t^m a_{m,n}(\mathbf{V}') \quad (5)$$

where c^n is the cost of performing one flop at mobile edge server n, including energy, maintenance, operating costs, etc.

[3] If a mobile device's computing task is not allocated to any edge server, the mobile device has two options depending on it local energy level. First, locally complete the task if it has enough energy. Note that locally completing the task meets the energy consumption and task processing delay constraints. Second, if the mobile device has insufficient energy to complete the computation, it can submit a new bid during the next available auction.

3.1. Auction Requirements

For the mobile devices and the edge servers to participate in the auction, we need to guarantee that the auction results cannot be unfairly affected by dishonest bidding strategies, that the mobile devices will receive non-negative utilities, and that the task processing delay and mobile device energy consumption delays are satisfied. We formalize these requirements in the following definitions.

Definition 1. ***Misreported Bids.*** *Let $v_{m,n}$ be mobile device m's true valuation for offloading its computing task to edge server n, and $v'_{m,n}$ be m's reported bid for edge server n. Then, reported bid $v'_{m,n}$ is said to be misreported if it is not equal to $v_{m,n}$, i.e., $v'_{m,n} \neq v_{m,n}$.*

Definition 2. ***Misreported Bid Sets.*** *Let $V_m = \{v_{m,1}, ..., v_{m,N}\}$ be mobile device m's set of bids and $V'_m = \{v'_{m,1}, ..., v'_{m,N}\}$ be the reported set of bids from m. Then, a reported bid set is said to be misreported if any $v'_{m,n} \in V'_m$ is misreported, i.e., $V'_m \neq V_m$.*

Definition 3. ***Dominant Strategy Incentive Compatible (DSIC).*** *Denote the set of reported bids with mobile device m's bid misreported by $\mathbf{V}'_m = \{V'_1, ..., V'_m, ..., V'_M\}$, where $V'_m \neq V_m$. Then, an auction with allocation rule a and payment rule p is said to be DSIC if no mobile device can improve its own utility by bidding untruthfully, i.e.,*

$$u_m(V_m, \mathbf{V}') \geq u_m(V_m, \mathbf{V}'_m), \quad \forall \mathbf{V}' \in \mathcal{V}, \forall V'_m \in \mathcal{V}_m, \forall m \in \mathcal{M}.$$

Definition 4. ***Individual Rationality (IR).*** *An auction with allocation rule a and payment p ensures individual rationality if mobile devices receive a non-negative utility by participating in the auction, that is,*

$$u_m(V_m, \mathbf{V}') \geq 0, \qquad \forall \mathbf{V}' \in \mathcal{V}, \forall m \in \mathcal{M}.$$

Definition 5. ***Feasible Offloading Assignments.*** *An auction with allocation rule a and payment rule p is feasible if the task offloading decisions satisfy the task processing delay, and energy consumption constraints in (2), and (4), respectively, and assigns at most one computing task to each edge server, and each computing task is assigned to at most one edge server.*

4. A Deep Neural Network for Auctions in Mobile Edge Computing

Finding a task offloading allocation rule a and a payment rule p that maximize the utility of the edge servers while satisfying the auction requirements in Section 3.1, is a challenging combinatorial auction problem without a known analytical solution [23]. To efficiently solve this problem, we propose two deep neural networks that can be trained to efficiently find task offloading allocations and payments. Figure 2 shows the architecture of the proposed neural networks.

4.1. Allocation Network

To allocate tasks to edge servers, we use a deep neural network that takes the set of reported bids from the mobile devices $\mathbf{V}'$ as input and outputs the probabilities $a_{m,n}(\mathbf{V}')$'s in the allocation matrix $\mathbf{A}(\mathbf{V}')$. The network contains K_a fully-connected hidden layers with hyperbolic tangent activation functions, and two fully connected output layers with `softmax` activation functions. The left part of Fig. 2 shows the allocation network.

Specifically, let $\mathbf{h}_a^k \in \mathbb{R}^{I_a^k}$ be the output of the kth hidden layer, $\mathbf{W}_a^k \in \mathbb{R}^{I_a^k \times I_a^{k-1}}$ and $\mathbf{b}_a^k \in \mathbb{R}^{I_a^k}$ be matrices denoting the weights and biases of the kth layer respectively, and I_a^k be the length of the kth hidden layer. Then, the computations performed by the kth hidden layer are as follows:

$$\mathbf{h}_a^k = \sigma\big(\mathbf{W}_a^k \cdot \mathbf{h}_a^{k-1} + \mathbf{b}_a^k\big), \forall k = 1 \ldots, K_a. \tag{6}$$

where σ is the `tanh` activation function. The initial hidden layer is given by the mobile device bids, i.e., $\mathbf{h}^0 = \mathbf{V}'$ and has length $I_a^0 = M(N+1)$.

The output layer takes the output of the K_ath hidden layer as input, and uses a `softmax` function, i.e.,

$$a_{m,n} = \frac{e^{h^K_{a[m,n]}}}{\sum_{k=1}^{M+1} e^{h^K_{a[k,n]}}},$$

where $h^K_{a[m,n]}$ is the element of $\mathbf{h}_a^N$ that is connected to allocation probability $a_{m,n}$'s output neuron. To account for the possibility that some edge servers do not receive any tasks, we have an additional element in the Kth hidden layer for each edge server.

Using the `softmax` layer, the allocation network makes sure that the task t^m of mobile device is only assigned to only one edge server, $\sum_{n=1}^{N} a_{m,n}(\mathbf{V}') \leq 1$, $\forall m \in \mathcal{M}$. To make sure that each edge server n is also assigned with only one task, $\sum_{m=1}^{M} a_{m,n}(\mathbf{V}') \leq 1$, $\forall n \in \mathcal{N}$, the allocation network uses a second `softmax` layer that takes the output from the first `softmax` layer as input, and outputs the final allocation probabilities as shown on the left side of Fig 2. Then, the system operator allocates the computing task to the edge server that received the largest probability for each computing task.

We note that the allocation network directly enforces that only one task is assigned to each server, and that a task is assigned to only one server due to the size of its output layer.

4.2. Payment Network

The payment deep neural network takes the reported bids $\mathbf{V}'$ as input and outputs the payments $p_m(\mathbf{V}')$ for all the mobile devices as shown on the right side of Fig. 2. The neural network is formed by K_p fully connected hidden layers where the first K_{p-1} layers apply hyperbolic tangent activation functions, and the K_pth layer applies the sigmoid activation function.

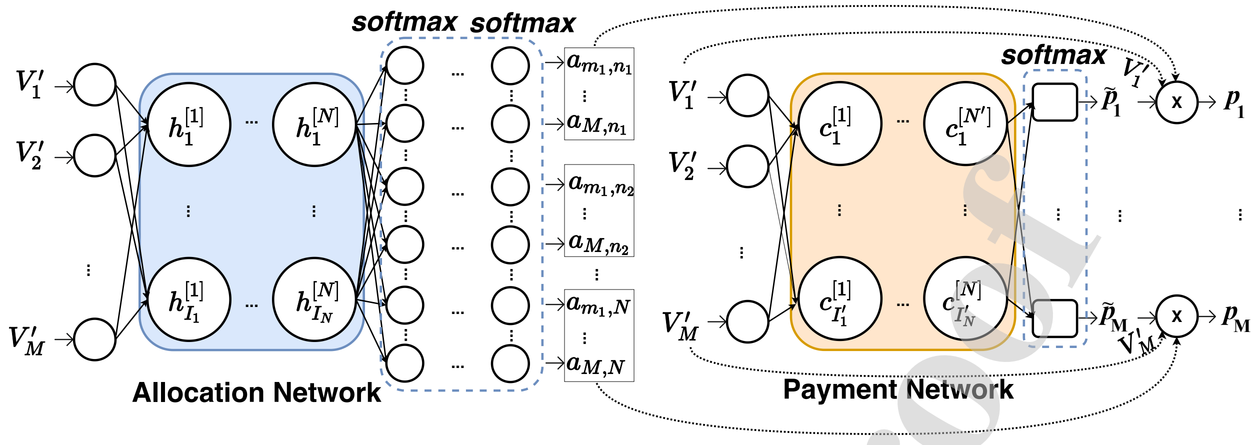


Figure 2: Diagram of the proposed deep neural networks.

The hidden layers in the payment network have the same structure as those in the allocation network. In particular, let $\mathbf{h}_p^k \in \mathbb{R}^{I_p^k}$ be the output of the kth hidden layer, $\mathbf{W}_p^k \in \mathbb{R}^{I_p^k \times I_p^{k-1}}$ and $\mathbf{b}_p^k \in \mathbb{R}^{I_p^k}$ be matrices denoting the weights and biases of the kth layer respectively, and I_p^k be the length of the kth hidden layer. The kth layer of the payment neural network performs computations similar to those in the hidden layers of the allocation network in (6).

The last layer of the payment network uses a `softmax` activation function to output a non-negative fractional value $\tilde{p}_m \in [0, 1]$, that we then transform into the expected payment for mobile device m as follows:

$$p_m = \tilde{p}_m \sum_{i=1}^{N} a_{m,i} v_{m,i}, \qquad \forall m \in M. \tag{7}$$

Note that by defining the payments as in (7) we enforce the individual rationality constraint in Definition 4 because mobile device m's payment is greater than zero only if it receives allocation probabilities that are larger than zero.

4.3. Training the Deep Neural Networks

We formulate an optimization problem to find the deep neural network parameters that can find an allocation rule a and a payment rule p that maximize the operator's utility while satisfying the DSIC, IR, and feasibility constraints in Section 3.1.

We start our formulation by defining a set of penalty functions that measure how much the allocation rule a and payment rule p deviate from the auction requirements. We later use the penalty functions to define the constraints of the optimization problem.

First, we measure how much an auction deviates from the DSIC requirement in Definition 3 by calculating the maximum gain in utility that a mobile device m can obtain by misreporting its bid under allocation rule a and payment rule p, i.e.,

$$P_m^R(a,p) = E_{\mathbf{V}' \sim F}\left[\max_{V'_m \in \mathcal{V}_m} \left(u_m(V_m, \mathbf{V}'_m) - u_m(V_m, \mathbf{V}')\right)\right], \tag{8}$$

where $\mathbf{V}'_m$ is the set of bids with mobile device m's bid misreported. Equation (8) is called the regret of mobile device m for untruthfully reporting its bid. Note that a zero regret for the mobile devices indicates that the auction satisfies the DSIC requirement.

We measure the tardiness of the task results at mobile device m as the amount of time that it takes to receive the result beyond the local computing time as defined in constraint (2), i.e.,

$$P_m^D(a,p) = E_{\mathbf{V}' \sim F}[\max(0, \sum_{n \in \mathcal{N}} a_{m,n}(\mathbf{V}') D_{off}^{m,n} - D_{local}^m)]. \tag{9}$$

The amount of energy that it takes mobile device m to offload its task beyond the amount of energy that it needs to take to locally compute its task as described in (4) is given by

$$P_m^E(a,p) = E_{\mathbf{V}' \sim F}[\max(0, \sum_{n \in \mathcal{N}} a_{m,n}(\mathbf{V}') E_{off}^{m,n} - E_{local}^m)], \tag{10}$$

Next, our objective is to minimize the negated utility of the edge servers obtained under auction:

$$\mathcal{L}(a,p) = -E_{\mathbf{V}' \sim F}[U(\mathbf{V}')] \tag{11}$$

We are now ready to formulate the optimization problem to train our allocation and payment neural networks. Let α denote the weights $\mathbf{W}_a^k$ and biases $\mathbf{b}_a^k$ of the allocation neural network (for all $k \in [1, K_a]$) , and β denote the weights $\mathbf{W}_p^k$ and biases $\mathbf{b}_p^k$ of the payment neural network (for all $k \in [1, K_p]$). Let a^α and p^β be the allocation and payment rules given by the allocation and payment neural networks under parameters α, and β, respectively. Then, the optimization problem to train the deep neural networks is given by

$$\begin{aligned} \min_{\alpha,\beta} \quad & \mathcal{L}(a^\alpha, p^\beta) \\ \text{s.t.} \quad & P_m^i(a^\alpha, p^\beta) = 0, \ \forall i \in \{R, D, E\}, \ \forall m \in \mathcal{M}. \end{aligned} \tag{12}$$

Note that we can accommodate mobile devices leaving and joining the network without retraining or redesigning the neural networks by setting the bids of missing mobile devices to zero.

4.4. A Computing Task Offloading Rule

Although the above deep neural networks can find allocation decision probabilities that satisfy the auction properties in expectation, the system operator needs a concise rule to decide which mobile device should offload its computing task to each edge server. To this end, we propose to allocate edge server n to the mobile device m with the highest probability $a_{m,n}$ for edge server n. The system operator can apply this rule by computing the allocation decisions as follows:

$$\hat{a}_{m,n}(\mathbf{V}'; a^\alpha) = \begin{cases} 1 & \text{if } \ m = \arg\ \max_i \{a_{i,n}(\mathbf{V}'; a^\alpha) | i \in \mathcal{M}\}, \\ 0 & otherwise \end{cases} \tag{13}$$

for all $m \in \mathcal{M}$ and $n \in \mathcal{N}$. If two or more mobile devices have the same allocation probability $a_{m,n}$ for some n, the system operator chooses one of them uniformly at random.

After finding the allocation decisions $\hat{a}_{m.n}$, the system operator calculates the mobile devices' payment as follows:

$$\hat{p}_m(\mathbf{V}'; p^\beta) = \tilde{p}_m(\mathbf{V}'; p^\beta) \sum_{i=1}^{N} \hat{a}_{m,n}(\mathbf{V}'; a^\alpha) v'_{m,i}. \tag{14}$$

Consequently, the utility of the mobile devices is

$$\hat{u}_m(V_m, \mathbf{V}') = \sum_{i=1}^{N} \hat{a}_{m,n}(\mathbf{V}'; a^\alpha) v'_{m,i} - \hat{p}_m(\mathbf{V}'; p^\beta)$$

for all $m \in \mathcal{M}$, and the utility of the edge server becomes

$$\hat{U}(\mathbf{V}') = \sum_{m \in \mathcal{M}} \hat{p}_m(\mathbf{V}'; p^\beta) - c^n o^m t^m$$

4.5. Sample Mean Optimization Problem

Directly solving (12) is challenging because of the expected values in the objective and constraints. Instead, we reformulate (12) in terms of sample bid profiles. Specifically, we sample L bid profiles i.i.d. from the distribution of profiles F and group them in set $\mathcal{S} = \{\mathbf{V}^{(1)}, \mathbf{V}^{(2)}, \ldots, \mathbf{V}^{(L)}\}$. To calculate the mobile devices' regret in (8), we sample an additional set of misreported bid profiles $\mathcal{S}'_l = \{\bar{\mathbf{V}}^{(1)}, \ldots, \bar{\mathbf{V}}^{(Q)}\}$ for each $\mathbf{V}^{(l)}$ in $\mathcal{S}$, where $\bar{\mathbf{V}}^k \in \mathcal{S}'_l$ (for all $k \in [1, Q]$) are sampled from the space $\mathcal{V}$ according to a distribution that is not necessarily equal to F.

Based on the sample set $\mathcal{S}$, and the utility of the edge servers given by the computing task allocation rule described in Section 4.4, we reformulate the objective in (12) as follows:

$$\hat{\mathcal{L}}(a^\alpha, p^\beta) = -\frac{1}{L} \sum_{l=1}^{L} \hat{U}. \tag{15}$$

The regret of the buyers can be rewritten in terms of the set of sample bid profiles $\mathcal{S}$ and the set of sample misreported bids $\mathcal{S}'_l$ as follows:

$$\hat{P}^R_m(a^\alpha, p^\beta) = \frac{1}{L} \sum_{l=1}^{L} \max_{\bar{\mathbf{V}} \in \mathcal{S}'_l} \left(\hat{u}_m(V^{(l)}_m, \bar{\mathbf{V}}^{(l)}_m) - \hat{u}_m(V^{(l)}_m, \mathbf{V}^{(l)}) \right),$$

where $\bar{\mathbf{V}}^{(l)}_m = \{V^{(l)}_1, V^{(l)}_2, ..., \bar{V}_m, ..., V^{(l)}_M\}$, and $\bar{V}_m \in \bar{\mathbf{V}}$. The sample delay constraint $\hat{P}^D_m(a^\alpha, p^\beta)$, and sample energy constraint $\hat{P}^E_m(a^\alpha, p^\beta)$ of the mobile devices as can be similarly defined as follows:

$$\hat{P}^D_m(a^\alpha, p^\beta) = \frac{1}{L} \sum_{l=1}^{L} \max\{0, \sum_{n \in \mathcal{N}} \hat{a}_{m,n}(\mathbf{V}^{(l)}; a^\alpha) D^{m,n}_{total} - \delta_m\}, \quad \forall m \in \mathcal{M} \tag{16}$$

$$\hat{P}^E_m(a^\alpha, p^\beta) = \frac{1}{L} \sum_{l=1}^{L} \max\{0, \sum_{n \in \mathcal{N}} \hat{a}_{m,n}(\mathbf{V}^{(l)}; a^\alpha) E^{m,n}_{total} - \beta_m\}, \quad \forall m \in \mathcal{M} \tag{17}$$

Based on the sample mean objective and constraints, we reformulate the optimization problem in (12) as follows:

$$\begin{aligned} \min_{\alpha, \beta} \quad & \hat{\mathcal{L}}(a^\alpha, p^\beta) \\ \text{s.t.} \quad & \hat{P}^i_m(a^\alpha, p^\beta) = 0, \forall i \in \{R, D, E\}, \ \forall m \in \mathcal{M}. \end{aligned} \tag{18}$$

To solve (18), we use a combination of the Lagrangian method of multipliers and the ADAM solver [24]. The main idea is to form an augmented Lagrangian function and iteratively update the parameters α and β using the ADAM solver. At every iteration, the ADAM solver computes the allocation a and payment p using the current state of the neural networks, calculates the value of the augmented Lagrangian function, and updates the neural network parameters α and β. The Lagrangian multipliers are held constant for a fixed number of iterations and then updated based on the current state of α and β.

Specifically, the augmented Lagrangian function of (18) is given by

$$\begin{aligned} \mathcal{F}_\rho(a^\alpha, p^\beta; \boldsymbol{\lambda}^R, \boldsymbol{\lambda}^D, \boldsymbol{\lambda}^E) &= \hat{\mathcal{L}}(a^\alpha, p^\beta) \\ &+ \sum_{m \in \mathcal{M}} \lambda^R_m \hat{P}^R_m(a^\alpha, p^\beta) + \frac{\rho}{2} \sum_{m \in \mathcal{M}} (\hat{P}^R_m)^2(a^\alpha, p^\beta)^2 \\ &+ \sum_{m \in \mathcal{M}} \lambda^D_m \hat{P}^D_m(a^\alpha, p^\beta) + \frac{\rho}{2} \sum_{m \in \mathcal{M}} (\hat{P}^D_m)^2(a^\alpha, p^\beta) \\ &+ \sum_{m \in \mathcal{M}} \lambda^E_m \hat{P}^E_m(a^\alpha, p^\beta) + \frac{\rho}{2} \sum_{m \in \mathcal{M}} (\hat{P}^E_m)^2(a^\alpha, p^\beta) \end{aligned} \tag{19}$$

where λ_m^R, λ_m^D, and λ_m^E are the Lagrangian multipliers corresponding to the DSIC, task processing delay, and energy consumption penalty constraints of mobile device m in (18) (for all $m \in \mathcal{M}$), respectively, and $\rho > 0$ is a constant. We group the Lagrangian multipliers into the vectors $\boldsymbol{\lambda}_t^R, \boldsymbol{\lambda}_t^D, \boldsymbol{\lambda}_t^E$.

Based on the augmented Lagrangian function, we iteratively update the neural network parameters as follows:

$$(a^\alpha, p^\beta) = \arg \min_{(a^\alpha, p^\beta)} \mathcal{F}_\rho(a_t^\alpha, p_t^\beta; \boldsymbol{\lambda}_t^R, \boldsymbol{\lambda}_t^D, \boldsymbol{\lambda}_t^E)$$

where t denotes the iteration number of the Lagrangian multiplier update. The $(t+1)$th update of the Lagrangian multipliers is given by the partial derivative of the Lagrangian function with respect to the penalty constraints, i.e.,

$$\begin{aligned}
\lambda_{m(t+1)}^R &= \lambda_{m(t)}^R + \rho\, \hat{P}_m^R(a_{t+1}^\alpha, p_{t+1}^\beta), \forall m \in M,\\
\lambda_{m(t+1)}^D &= \lambda_{m(t)}^D + \rho\, \hat{P}_m^D(a_{t+1}^\alpha, p_{t+1}^\beta), \forall m \in M,\\
\lambda_{m(t+1)}^E &= \lambda_{m(t)}^E + \rho\, \hat{P}_m^E(a_{t+1}^\alpha, p_{t+1}^\beta), \forall m \in M.
\end{aligned}$$

Algorithm 1 describes the training process of the proposed auction mechanism.

Algorithm 1 Training of Auction Mechanism

```
Input: Mini-batches of $S_1, ..., S_T$ of size B.
  Initialize: $\mathbf{W}_a^k \in \mathbb{R}^{I_a^k \times I_a^{k-1}}$, $\mathbf{W}_p^k \in \mathbb{R}^{I_p^k \times I_p^{k-1}}$, $\boldsymbol{\lambda}_t^R, \boldsymbol{\lambda}_t^D, \boldsymbol{\lambda}_t^E$
  for $t = 1$ to $T$ do
    Receive mini-batch $S_t$.
    Initialize misreported bid $V_m^{(l)} \in V_m$, $\forall l \in [L]$, $\forall m \in M$.
    for $r = 1$ to $Q$ do
      Perform gradient update to optimize $V_m^{(l)}$, $\forall l \in [L]$, $\forall m \in M$.
    end for
    Compute constraints gradients:
      $\hat{P}_m^j$, $\forall j \in \{R, D, E\}$, $\forall l \in [L]$, $\forall m \in M$.
    Compute Lagrangian gradient in (19) and update $\alpha$ and $\beta$.
    Update Lagrangian multipliers every $Z$ iterations:
    if $t \% Z = 0$ then
      $\lambda_{m(t+1)}^j \leftarrow \lambda_{m(t)}^j + \rho\, \hat{P}_m^j(a_{t+1}^\alpha, p_{t+1}^\beta), \forall j \in \{R, D, E\}, \forall m \in M$.
    else
      $\lambda_{m(t+1)}^j \leftarrow \lambda_{m(t)}^j, \forall j \in \{R, D, E\}, \forall m \in M$
    end if
Output: $a^\alpha, p^\beta$
  end for
```

4.6. Overview of the Auction Procedure

In this subsection, we summarize the overall proposed auction procedure. First, the system operator trains the allocation and neural networks as described in Section 4.5. Once the neural networks are trained, the system operator collects the bid vectors from the mobile devices V'_m (for all $m \in \mathcal{M}$), and uses them as inputs to both the allocation and payment networks. Second, based on the task allocation rule in Section 4.4, the system operator decides to which edge node each mobile device will offload its computing task, and calculates the payment for each mobile device. Third, the mobile devices submit their payments to the system operator, and upload their computing tasks to their assign edge node. The edge nodes compute their assigned tasks and return the results to the mobile devices.

Algorithm 2 Auction Procedure

```
Input: Trained allocation and payment deep neural networks
1: Receive mobile devices $m$ bid sets $V'_m(\forall m \in \mathcal{M})$, and form the
   vector of all bids $\mathbf{V}'$ .
2: Feed $\mathbf{V}'$ to the allocation and payment networks.
3: Find the allocation decisions $\hat{a}_{m,n}(\mathbf{V}'; a^\alpha)$ using (13), and share
   it with mobile device $m$ and edge node $n$ ($\forall m \in \mathcal{M}$, and $\forall n \in \mathcal{N}$).

4: for $m \in \mathcal{M}$ do
5:   Calculate mobile device $m$ payment $\hat{p}_m(\mathbf{V}'; p^\beta)$ using (14),
     and collect it.
6:   If $\hat{a}_{m,n}(\mathbf{V}'; a^\alpha) = 1$, then offload computing task from mobile
     device $m$ to the edge server $n$.
7:   If $\hat{a}_{m,n}(\mathbf{V}'; a^\alpha) = 1$, send task results from edge server $n$ to
     mobile device $m$.
8: end for
```

5. Simulation Results

We run extensive simulations of our proposed test our deep neural network auction mechanism, and compare its performance to existing auction mechanisms in terms of the utility obtained by the edge servers and the its ability to satisfy the task processing and energy consumption constraints of the mobile devices.

5.1. Simulation Setup

We consider a mobile edge computing network with the following settings. The mobile devices' transmission, receiving, and idling power are 1.3 W, 1.6 W, and 0.7 W, respectively. The size of the task result is set to 20% of the original task size. Nodes are placed at random locations on a 200m × 200m area. The data rates $d^{m,n}$'s between mobile devices and servers are calculated using the Log-normal shadowing propagation model with a 10 Mhz bandwidth. The bids of the mobile devices are calculated as follows:

$$v_{m,n} = \frac{t_m}{D_{off}^{m,n} + E_{off}^{m,n}}.$$

We summarize the parameters of our mobile edge computing network simulation in Table 1.

To investigate the utility obtained by our proposed approach and how well it can satisfy the constraints, we consider three scenarios each one with 20 mobile devices and 10 edge servers.

- **Scenario I** considers mobile devices with processing rates randomly chosen from the set $f^m \in \{10, 20, 30\}$ G flops, and processing power consumption 1.1 W.
- **Scenario II** consider a more strict task processing delay constraint by increasing the local processing rate of the mobile devices to $f^m \in \{60, 70, 80\}$ G flops and keeping their power consumption at 1.1W.

Table 1: Simulation Parameters

Parameter	Value
Area	200m x 200m
Channel model	Log-normal shadowing
Size of computing task	$t^m \in [1-5]$ MB
Processing rate of mobile devices for Scenarios I and III	$f^m \in \{10, 20, 30\}$ G flops/s
Processing rate of mobile devices for Scenario II	$f^m \in \{60, 70, 80\}$ G flops/s
Processing rate of edge servers	$f^n \in \{100, 125, 150\}$ G flops/s
Task operations	$o^m = 10 \times 10^9$ flops/MB of data
Local processing energy consumption for Scenarios I and II	$E^m_{pr} = 1.1$W
Local processing energy consumption for Scenario III	$E^m_{pr} = 0.5$W
Transmission energy consumption	$E^m_{tx} = 1.3W$
Receiving energy consumption	$E^m_{rx} = 1.6W$
Idle energy consumption	$E^m_{id} = 0.7$ W
Task result size	$r^m = 0.2t^m$ MB

- **Scenario III** considers a more strict mobile device energy consumption constraint by decreasing the local task processing power consumption of the mobile devices to 0.5 W and keeping the processing rates as in Scenario I.

To train the neural networks, we generate a sample set $\mathcal{S}$ with $L = 640,000$ of sample bid profiles, as well as one set of misreported bids $\mathcal{S}'_l$'s for each bid profile in $\mathcal{S}$. We set aside $10,000$ bid profiles for testing. We train the deep neural networks over 80 epochs with $5,000$ mini-batches of size $B = 128$.

In addition, we tested our deep neural networks under different hyper-parameters to find the best performing ones. Specifically, we trained the networks with varying number of layers and layer sizes, and found that the highest utility for the edge servers is achieved with 3 hidden layers of 100 neurons each. We initialized the value of ρ in the Lagrangian function to 0.01, and set that to increment every 2 epochs. The update on $\lambda^j_m, \forall j \in R, D, E$ parameter is also set to perform once in every 100 mini-batches, (i.e., Z = 100).

We use TensorFlow to implement the deep neural networks on a general purpose PC with an Nvidia GTX 1070 Ti GPU, 16GB RAM, and 1TB SSD external memory to implement all the simulations.

5.2. Results

We first evaluate the utility of the edge servers obtained by our proposed deep neural network under Scenarios I, II, and III in Fig. 3. We compare our results to the utility obtained under a multi-unit auction [25], which does not consider the tasks processing delay and mobile device energy consumption constraints. In Scenario I, our approach obtains a utility that is three times higher than that obtained by the multi-unit auction. In Scenarios II and III, which have more strict constraints, we observe a utility reduction due to more tasks from the mobile devices being unassigned to an edge server. The reason is that it becomes harder for the system to assign tasks to edge servers without violating the constraints and, thus, it leaves some tasks unassigned. However, we see that our approach still finds a higher utility than the multi-unit auction scenarios while satisfying the constraints in all cases.

Next, we investigate how well our proposed approach can meet the regret, task processing delay, and mobile device energy consumption constraints. In Fig. 4, we see that our approach reaches an allocation and payment rule

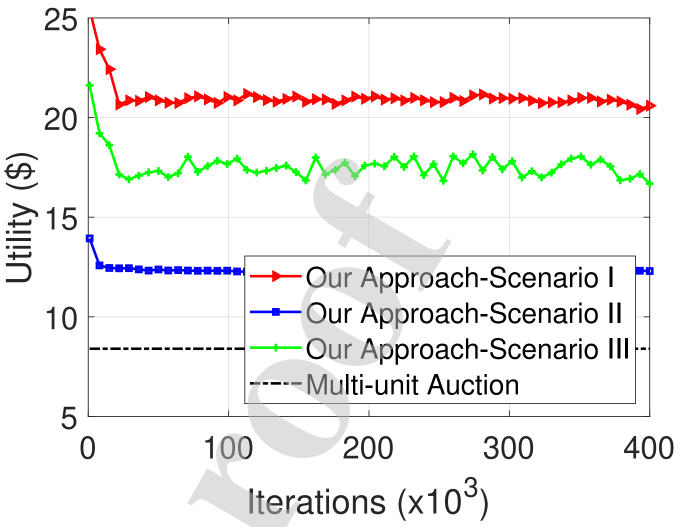


Figure 3: Utility of the edge servers.

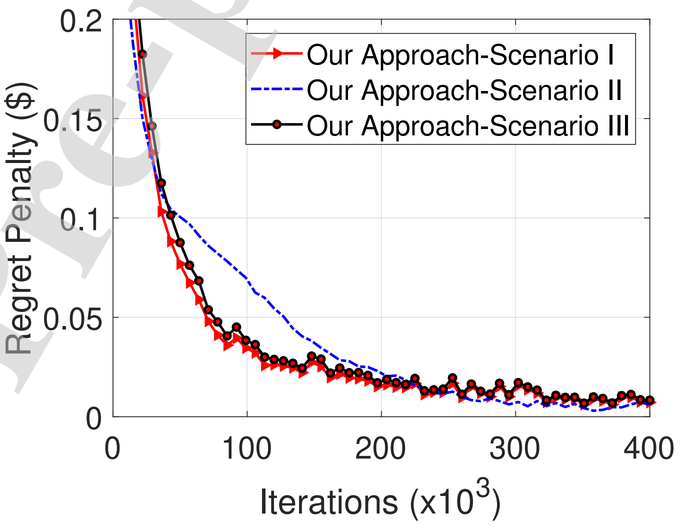


Figure 4: Mobile device regret.

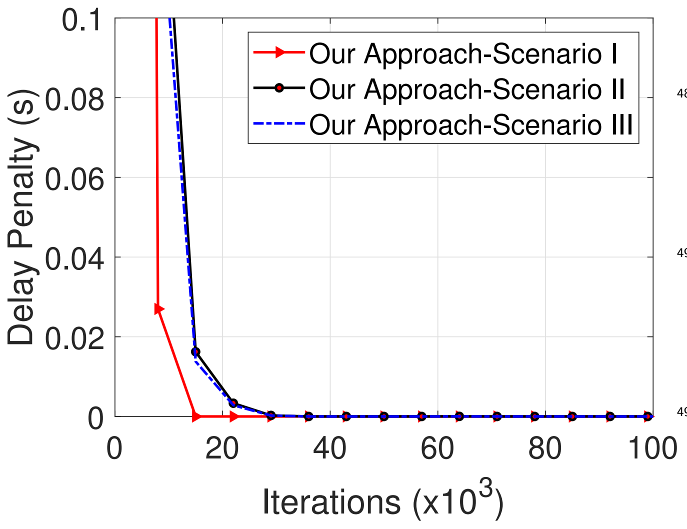


Figure 5: Task completion tardiness.

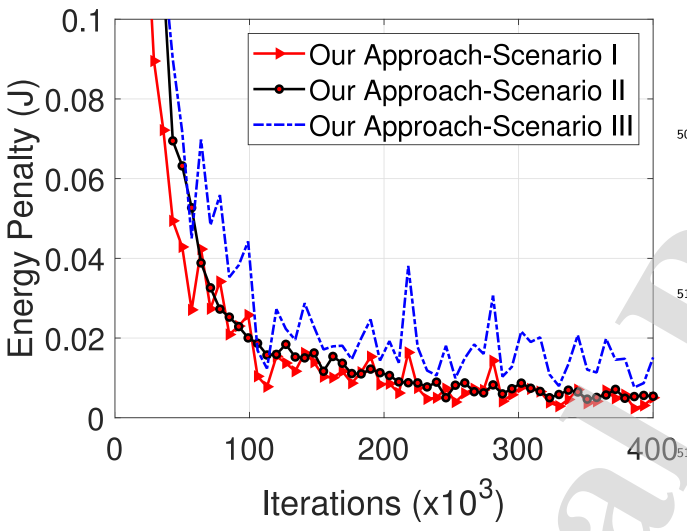


Figure 6: Additional energy consumption ($M = 20, N = 10$).

with a negligible regret, which indicates that the our auction meets the DSIC requirement for all three scenarios.

In Fig. 5, we show that the tardiness of the computing tasks approaches zero within the first few training iterations, which indicates that tasks are completed with delays that are at most the delays when they are locally processed in all scenarios.

Fig. 6 shows the difference between the energy that mobile devices consume when offloading their tasks, and when locally computing their tasks. We observe that although Scenarios II and III take longer to reach a low value due to their stricter constraints, all scenarios are able to reach similar energy penalty values.

Next, we compare our proposed deep learning auction to the auction mechanism proposed by [21], which is the work closest to ours. The authors in [21] consider a scenario where there is only one edge server, and multiple users submit their bids aiming to obtain access to the server's computing resources. They use a deep learning based auction to find the winner mobile device that can access to computing resources provided by the single edge server. Thus, to make a fair comparison, we measure the utility obtained by the edge server, the task processing delay and the energy consumption of the mobile devices obtained under both approaches for varying numbers of mobile devices in the network.

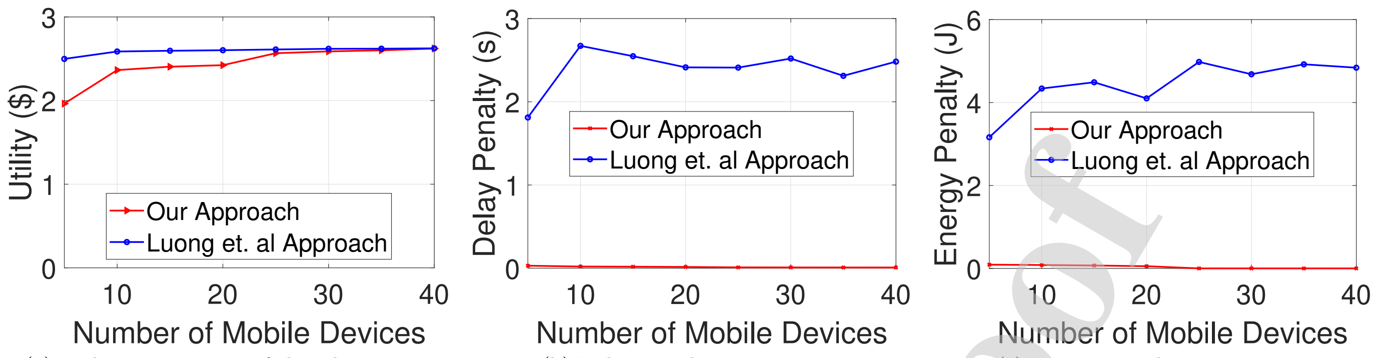
In Fig. 7a, we compare the utility of our proposed neural networks with those proposed by [21]. We observe that when the number of mobile devices is small, our approach achieves a lower utility than [21]. However, as the number of mobile devices increases, the utility achieved by our auction mechanism converges with the utility obtained by [21] approach. The reason is that when the number of users is small there are only a few solutions that satisfy the constraints, and thus it can only choose from a few. When there are more users, our approach can choose from more solutions that satisfy the constraints and can achieve higher utility.

In Figures. 7b and 7c, we show the task processing delay and energy consumption penalty violations defined in (9) and (10) of the two approaches. We see that for all the scenarios our neural networks easily satisfy the task processing and energy consumption constraints of the mobile devices while the approach by Luong et. al [21] results in significant constraint violations.

Finally, we investigate the computational complexity of our proposed auction, and show the results in Table 2. We find that our deep learning neural networks can find the allocation and payments in a very short time, e.g., 2.14s for the scenarios with 20 mobile devices and 10 edge servers. We also observe that the training time for 20 mobile devices and 10 edge servers on a relatively modest hardware is 10.2 hours. For the scenarios with only one edge server, we observe even lower training and computing time, due to the smaller deep neural networks that we need to train.

Table 2 also shows the time it takes the truthful multi-unit auction in [25] to find the allocations and payments for the mobile devices. We observe that the computing time of our proposed deep neural network auction mechanism and the truthful multi-unit auction are comparable. Although the multi-unit auction can run faster than our proposed neural networks, we note that it is unable to meet the task completion and energy consumption constraints of the mobile devices. For example, in a network with 20 mobile devices and 10 edge servers, the multi-unit auction finds a utility of \$8.21 and violates the task completion delay and energy consumption constraints by 2.8 seconds and 4.3 J, respectively. In the same network, our proposed neural networks found a utility of \$ 20.7, which is twice more than the multi-unit auction, and meets the constraints.

We emphasize that it is challenging to create a fair comparison between our approach and the other existing task offloading works for edge computing, e.g., [12, 13, 16, 18, 19, 21, 22, 20] because most of them ignore the



(a) Utility comparison of the edge server. (b) Delay penalty comparison. (c) Energy penalty comparison.

Figure 7: Performance evaluation of the proposed neural networks.

Table 2: Training and computing time of the deep neural networks for MECs of different sizes.

(Edge Servers, Mobile Devices)	(1,5)	(1, 10)	(1, 15)	(1, 20)	(1, 25)	(20, 10)
Training Time (Proposed auction)	3.4(h)	4.37(h)	4.4(h)	4.9(h)	5.28(h)	10.2(h)
Computing Time (Proposed auction)	0.38(s)	0.41(s)	0.58(s)	0.67(s)	0.76(s)	2.14(s)
Computing Time (Multi-unit auction)	0.18(s)	0.21(s)	0.30(s)	0.36(s)	0.39(s)	1.04(s)

constraints on task completion delays and mobile device energy consumption.

6. Conclusions

We have investigated the problem of finding a task offloading strategy in mobile edge computing that ensures that edge servers make a profit while the mobile devices receive the results of their tasks within a deadline using a minimum amount of energy. To this end, we propose an auction that allocates edge servers to the tasks from mobile devices and sets the payments for the computing services. The auction is carried out by a pair of deep neural networks that prevent mobile devices from unfairly affecting the results of the allocation using untruthful bidding strategies. The allocations satisfy the task completion delay and energy consumption of the mobile devices. We extensively evaluate our proposed auction mechanism and observe that it can obtain a significant increase in utility for the edge servers compared to existing auctions. We also see that the edge computing server allocation meets the task completion delays and energy consumption constraints of the mobile devices.

7. Funding Sources

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

[1] J. Huang, Q. Duan, C.-C. Xing, H. Wang, Topology control for building a large-scale and energy-efficient internet of things, IEEE Wireless Communications 24 (1) (2017) 67–73.

[2] Y. Cao, S. Chen, P. Hou, D. Brown, FAST: A fog computing assisted distributed analytics system to monitor fall for stroke mitigation, in: 2015 IEEE International Conference on Networking, Architecture and Storage (NAS), Boston, MA, USA, 2015, pp. 2–11.

[3] V. Stantchev, A. Barnawi, S. Ghulam, J. Schubert, G. Tamm, Smart items, fog and cloud computing as enablers of servitization in healthcare, Sensors & Transducers 185 (2) (2015) 121.

[4] M. Schneider, J. Rambach, D. Stricker, Augmented reality based on edge computing using the example of remote live support, in: 2017 IEEE International Conference on Industrial Technology (ICIT), IEEE, 2017, pp. 1277–1282.

[5] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin, K.-W. Wen, K. Kim, R. Arora, A. Odgers, L. M. Contreras, S. Scarpina, MEC in 5G networks, White Paper 28, ETSI (Jun. 2018).

[6] A. Al-Shuwaili, O. Simeone, Energy-efficient resource allocation for mobile edge computing-based augmented reality applications, IEEE Wireless Communications Letters 6 (3) (2017) 398–401.

[7] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, V. Young, Mobile edge computing—a key technology towards 5g, ETSI white paper 11 (11) (2015) 1–16.

[8] Z. Jiang, S. Mao, Energy delay tradeoff in cloud offloading for multi-core mobile devices, IEEE Access 3 (2015) 2306–2316.

[9] J. Zhang, X. Hu, Z. Ning, E. C.-H. Ngai, L. Zhou, J. Wei, J. Cheng, B. Hu, Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks, IEEE Internet of Things Journal 5 (4) (2017) 2633–2645.

[10] Y. Zhang, C. Lee, D. Niyato, P. Wang, Auction approaches for resource allocation in wireless systems: A survey, IEEE Communications surveys & tutorials 15 (3) (2013) 1020–1041.

[11] L. Zheng, C. Joe-Wong, C. W. Tan, M. Chiang, X. Wang, How to bid the cloud, ACM SIGCOMM Computer Communication Review 45 (4) (2015) 71–84.

[12] J. Liu, Y. Mao, J. Zhang, K. B. Letaief, Delay-optimal computation task scheduling for mobile-edge computing systems, in: 2016 IEEE International Symposium on Information Theory (ISIT), Barcelona, Spain, 2016, pp. 1451–1455.

[13] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, Y. Zhang, Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks, IEEE Access 4 (2016) 5896–5907.

[14] Y. Dai, D. Xu, S. Maharjan, Y. Zhang, Joint computation offloading and user association in multi-task mobile edge computing, IEEE Transactions on Vehicular Technology 67 (12) (2018) 12313–12325.

[15] Z. Yang, C. Pan, K. Wang, M. Shikh-Bahaei, Energy efficient resource allocation in uav-enabled mobile edge computing networks, IEEE Transactions on Wireless Communications 18 (9) (2019) 4576–4589.

[16] J. Zhang, X. Hu, Z. Ning, E. C. . Ngai, L. Zhou, J. Wei, J. Cheng, B. Hu, Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks, IEEE Internet of Things Journal 5 (4) (2018) 2633–2645.

[17] A. Bozorgchenani, F. Mashhadi, D. Tarchi, S. A. Salinas Monroy, Multi-objective computation sharing in energy and delay constrained mobile edge computing environments, IEEE Transactions on Mobile Computing (2020) 1–1.

[18] Y. Jiao, P. Wang, D. Niyato, Z. Xiong, Social welfare maximization auction in edge computing resource allocation for mobile blockchain, in: 2018 IEEE international conference on communications (ICC), IEEE, 2018, pp. 1–6.

[19] T. Bahreini, H. Badri, D. Grosu, An envy-free auction mechanism for resource allocation in edge computing systems, in: 2018 IEEE/ACM Symposium on Edge Computing (SEC), IEEE, 2018, pp. 313–322.

[20] Z. Li, Z. Yang, S. Xie, Computing resource trading for edge-cloud-assisted internet of things, IEEE Transactions on Industrial Informatics.

[21] N. C. Luong, Z. Xiong, P. Wang, D. Niyato, Optimal auction for edge computing resource management in mobile blockchain networks: A deep learning approach, in: 2018 IEEE International Conference on Communications (ICC), IEEE, 2018, pp. 1–6.

[22] A. Kiani, N. Ansari, Toward hierarchical mobile edge computing: An auction-based profit maximization approach, IEEE Internet of Things Journal 4 (6) (2017) 2082–2091.

[23] P. Dütting, Z. Feng, H. Narasimhan, D. C. Parkes, Optimal auctions through deep learning, arXiv preprint arXiv:1706.03459.

[24] Z. Feng, H. Narasimhan, D. C. Parkes, Deep learning for revenue-optimal auctions with budgets, in: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 354–362.

[25] C. Borgs, J. Chayes, N. Immorlica, M. Mahdian, A. Saberi, Multi-unit auctions with budget-constrained bidders, in: Proceedings of the 6th ACM conference on Electronic commerce, ACM, 2005, pp. 44–51.

Author Biographies

Farshad Mashhadi received his BSc degree in Electrical engineering and Power Systems from the Azad University of Arak, Iran, in 2013, and the MSc degree in Computer Networks from the Azad University of Qazvin, Iran, in 2016. He is currently working toward his Ph.D. degree at cyber-physical system security lab (CPSSL) at the Wichita State University. His research interests include cyber-physical systems, cloud computing, and optimization algorithms.

Sergio Salinas received the BS degree in Telecommunications Engineering from Jackson State University, Jackson, in 2010, and the PhD degree in Electrical and Computer Engineering from Mississippi State University, Starkville, MS, in 2015. He is currently an Assistant Professor in the Department of Electrical Engineering and Computer Science, Wichita State University, Wichita, KS. His research interests include security and privacy in cyberphysical systems, cloud computing, and big data. He is a member of the IEEE.

Arash Bozorgchenani was born in Rasht, Iran on 19th December 1989. He has received a B.Sc. degree in Information Technology in 2013, and a M.Sc degree in Information Technology with a major in Computer Networks in 2016 in Iran. Since 2016, He has been a Ph.D. student in Electronics, Telecommunications and Information Technology at University of Bologna, Italy. He is University of Bologna's full scholarship recipient. He has also been involved in the national research project Gaucho (PRIN 2015) in Italy. Arash has done a 6-month research at university of Manitoba, Canada as a visiting PhD student. His research interest is in the area of wireless communication and resource allocation in wireless networks. He is working more specifically on fog/edge computing toward IoT and 5G applications. He is an IEEE student member since 2017.

Daniele Tarchi [S'99, M'05, SM'12] was born in Florence, Italy, on 4th October 1975. He received his M.Sc. degree in Telecommunications Engineering and Ph.D. degree in Informatics and Telecommunications Engineering from the University of Florence, Florence, Italy, in 2000 and 2004, respectively.

From 2004 to 2010, he was a Research Associate with University of Florence, Italy. From 2010 to 2019 he was an Assistant professor at the University of Bologna, Bologna, Italy. He is now an Associate Professor at the University of Bologna, Italy. He is author of more than 100 papers, among which more than 30 on international journals. His research interests are mainly on resource allocation techniques, heterogeneous networks, Cognitive Radios and Networks, and Wireless Multimedia Networks, applied to both terrestrial and satellite wireless communications. Recently he has mainly focused on Smart City scenarios, multimedia systems, Fog Networks, and Smart Grid. He has been involved in several national and international research projects, and collaborates with several foreign research institutes.

Prof. Daniele Tarchi is an IEEE Senior Member since 2012. He is Editorial Board member for IEEE Wireless Communications Letters, IEEE Transactions on Vehicular Technology and IET Communications. He has been symposium co-chair for IEEE WCNC 2011, IEEE Globecom 2014, IEEE Globecom 2018 and IEEE ICC 2020, and workshop co-chair at IEEE ICC 2015.

Authors' Pictures

Farshad Mashhadi

Sergio Salinas Monroy

Arash Bozorgchenani

Daniele Tarchi

Author statement

Farshad Mashhadi: Methodology, Software, Writing - Original Draft
Sergio A. Salinas Monroy: Conceptualization, Writing - Original Draft
Arash Bozorgchenani: Writing - Review & Editing
Daniele Tarchi: Conceptualization, Writing - Review & Editing

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: