Multiple Access Binary Computation Offloading via Reinforcement Learning

Mahsa Salmani*, Foad Sohrabi†, Timothy. N. Davidson* and Wei Yu†
*McMaster University, Hamilton, Canada
†University of Toronto, Toronto, Canada

***Abstract*—Computation offloading enables energy-limited mobile devices to expand the range of applications that they can execute. When multiple devices each seek to execute a latency-constrained indivisible task, the problem of device energy minimization involves jointly making binary decisions on whether or not each user should offload its task along with the allocation resources to the offloading users. It has been shown that for a K-user system that employs a multiple access scheme that exploits the full capabilities of the channel, when the binary decisions are given, a closed-form expression for the optimal resource allocation can be obtained. In this paper, we propose a reinforcement learning-based algorithm for finding offloading decisions that takes advantage of this closed-form expression for the resource allocation. Our numerical experiments illustrate that the proposed algorithm can achieve a better trade-off between performance and computational cost as compared to the existing approaches in the literature.**

I. Introduction

The opportunity for computation offloading provided by the mobile edge computing (MEC) architecture enables mobile devices to expand the range of energy-extensive and latency-sensitive computational tasks that they can support [1]–[3]. In order to exploit this offloading opportunity, the available communication and computation resources must be efficiently shared among all the devices. This resource allocation problem usually incolves minimizing the total energy consumption of the devices in the offloading system [4]–[6].

The structure of the resource allocation problem depends on the nature of the tasks to be offloaded. Tasks that have tightly coupled components are considered to be indivisible and hence the system must make a binary decision to either completely offload the task or execute it locally. Tasks with more loosely coupled components may be able to exploit the implicit parallelism between the user and the access point by offloading a fraction of the task (partial offloading). The energy consumed by binary or partial offloading depends on the extent to which the multiple access scheme can exploit the capabilities of the channel. "Capacity-achieving" schemes that fully exploit those capabilities (FullMA) can significantly reduce the energy consumption over sub-optimal schemes such as the time division multiple access scheme; e.g., [7]–[10].

In this work, we consider the energy minimization problem for a multi-user binary computation offloading system with a FullMA scheme, in which each user seeks to complete its indivisible computational task within its own specific latency constraint. For such a system, the optimization problem involves binary offloading decisions and a resource allocation among the users that are selected to offload their tasks. There are different techniques that can be considered to address the combinatorial structure of this problem. One could relax the binary constraints, treat the problem as a partial offloading problem and then round the fractional solution to a binary one; e.g., [11]. Alternatively, one could decompose the problem into a tree search for the offloading decisions, with a resource allocation problem being solved at each node in the tree [7].

The main goal of this paper is to tackle the energy minimization problem in a FullMA binary offloading system by training a deep neural network (DNN) in such a way that it can generate a set of good binary offloading decisions based on the channel gains of the users and the requirements of the tasks that the users seek to complete. To train such a network, we take the advantage of the closed-form optimal solution to the resource allocation problem for a given set of binary offloading decisions in [7] to propose an algorithm based on the reinforcement learning (RL) framework. Our numerical results illustrate that the proposed algorithm outperforms the algorithms that are based on rounding and randomized rounding approaches, in terms of the performance and the computational cost. It is also exhibited in our numerical experiments that with a performance close to that of the greedy search approach in [7], the computational cost of the proposed RL-based algorithm is about one order of magnitude less than that of the greedy search in [7]. Although the RL framework has been explored in other offloading settings [12]–[14], those works have been limited to simplified multiple access schemes. We believe that this work is the first that employs the RL framework to address the energy minimization problem in a FullMA binary computation offloading system.

II. System Model and Problem Formulation

Consider an offloading system with K single-antenna users, each of which seeks to complete its computational task within its specific latency constraint. The computational tasks are considered to be indivisible, and hence each user either fully offloads its task or completes it locally. The offloading system employs a multiple access scheme that exploits the full capabilities of the channel, i.e., a FullMA scheme, and the users are served by a single-antenna access point equipped with a computational resource that is large enough to provide all the users with the option of offloading their tasks. To develop a communication model for this scenario we let s_k denote the transmitted signal from user k and we let h_k denote the

978-1-7281-0954-1/19/$31.00 ©2019 IEEE

corresponding channel. The received signal at the access point can then be written as $y = \sum_{k=1}^{K} h_k s_k + v$, where v is a circular zero mean white Gaussian noise with variance σ^2.

The problem of interest in this paper is to minimize the energy that the mobile users expend on completing their computational tasks. This involves the decisions on whether or not each user offloads its task and the allocation of communication resources to the offloading users so that the selected users can simultaneously offload their tasks with minimal energy consumption. In order to formulate that problem, let x_k denote the offloading decision for the k^{th} user as follows,

$$x_k = \begin{cases} 1, & \text{if user } k \text{ offloads its task,} \\ 0, & \text{if user } k \text{ completes its task locally.} \end{cases} \tag{1}$$

If E_{off_k} and E_{loc_k} denote the energy consumptions of the k^{th} user in the case that it offloads its task, or executes it locally, respectively, then the total energy consumption of the users is

$$E_{\text{total}} = \textstyle\sum_k x_k E_{\text{off}_k} + (1 - x_k) E_{\text{loc}_k}. \tag{2}$$

Let B_k denote the description length of the task of the k^{th} user. If R_k and P_k denote the transmission rate and the transmission power (in units per channel use) of that user, respectively, then the energy that user k would expend to offload its task is

$$E_{\text{off}_k} = \tfrac{T_s B_k}{R_k} P_k, \tag{3}$$

where T_s is the symbol interval of the channel. In this paper, we assume that the users employ dynamic voltage scaling computation architecture which has been shown that minimizes the local computation energy consumption in the users [15]. The local energy consumption in that architecture is modeled as [15],

$$E_{\text{loc}_k} = \tfrac{M_k}{L_k^2} B_k^3, \tag{4}$$

where the coefficient M_k depends on the characteristics of the chip of user k.

We consider an offloading model in which the full description of a computational task should be received before the access point starts executing the task, and the results will be sent back to the k^{th} user when its task has been fully executed. If t_{UL_k} denotes the time it takes for the k^{th} user to send its task to the access point, t_{exe_k} denotes the time it takes for the access point to execute that task, and t_{DL_k} denotes the time it takes to send the result back to the user k, the latency constraint of the k^{th} user can be written as

$$t_{\text{UL}_k} + t_{\text{exe}_k} + t_{\text{DL}_k} \le L_k, \tag{5}$$

in which L_k denotes the maximum allowable latency for user k. Under the assumption of sufficiently large computational resources in the access point, and also sufficiently large downlink communication resources for sending the results back to the users, we can assume that $T_k = t_{\text{exe}_k} + t_{\text{DL}_k}$ is a constant for each user, e.g., [4], [16], [17]. Finally, the transmission time of the k^{th} user can be written as

$$t_{\text{UL}_k} = T_s\big(\tfrac{B_k}{R_k}\big). \tag{6}$$

In order to define the achievable rate region of a FullMA scheme, let $\mathcal{S} = \{1, 2, \ldots, K\}$ denote the set of users, and let $\mathcal{N}$ denote any subset of that set, $\mathcal{N} \subseteq \mathcal{S}$. The achievable rate region of a FullMA scheme is constrained by $2^K - 1$ constraints of the form

$$\textstyle\sum_{i\in\mathcal{N}} R_i \le \log\big(1 + \sum_{i\in\mathcal{N}} \alpha_i P_i\big). \tag{7}$$

Using the notations and the definitions above, the user energy minimization problem for a system with a full multiple access scheme can be written as

$$\min_{\{x_k, R_k, P_k\}_{k=1}^K} \ \textstyle\sum_k x_k\big(\tfrac{T_s B_k}{R_k} P_k\big) + (1 - x_k)\tfrac{M_k}{L_k^2} B_k^3 \tag{8a}$$

$$\text{s.t.} \quad x_k\big(T_s\big(\tfrac{B_k}{R_k}\big) + T_k\big) \le L_k, \quad \forall k, \tag{8b}$$

$$0 \le R_k, \quad \forall k, \tag{8c}$$

$$2^{\sum_{i\in\mathcal{N}} R_i} \le 1 + \textstyle\sum_{i\in\mathcal{N}} \alpha_i P_i, \quad \forall \mathcal{N} \subseteq \mathcal{S}, \tag{8d}$$

$$x_k \in \{0, 1\}, \quad \forall k, \tag{8e}$$

in which $\alpha_i = \frac{|h_i|^2}{\sigma^2}$. Note that the constraints in (8c) and (8d) imply that the powers are all non-negative.

This problem has a mixture of binary and continuous variables. One conventional way to address the resulting combinatorial structure is to partition the problem into an energy minimization problem over the continuous variables, $\{R_k\}_{k=1}^K$ and $\{P_k\}_{k=1}^K$, for a given set of offloading decisions, $\{x_k\}_{k=1}^K$, and a searching strategy over the binary offloading decisions to find a good set of offloading decisions.

In [8], a closed-form solution for the optimal resource allocation for a given set of offloading decisions was obtained. In theory, an exhaustive search over all possible binary offloading decisions can then be applied to the problem in (8) to achieve the optimal set of offloading decisions. However, the search space of 2^K possibilities results in a large computational cost. In this paper, we take the advantage of the closed-form optimal solution of the resource allocation problem in [8] to propose a reinforcement learning-based algorithm to obtain a close-to-optimal offloading decisions. In Section III, we will provide a brief explanation of the closed-form solution to the resource allocation problem for a given set of offloading decisions. In Section IV, we will exploit that closed-form optimal solution to propose a RL-based approach that tackles the problem of finding the solutions for offloading decisions.

III. Optimal Resource Allocation for Given Offloading Decisions

For a given set of offloading decisions, $\{x_k\}_{k=1}^K$, if $\mathcal{S}' \triangleq \{k | x_k = 1\}$, with $|\mathcal{S}'| = K'$, denotes the set of users that are selected to offload their tasks, then the remaining resource allocation problem can be written as

$$\min_{\{R_k, P_k\}_{k=1}^{K'}} \ \textstyle\sum_{k\in\mathcal{S}'} \tfrac{T_s B_k}{R_k} P_k \tag{9a}$$

$$\text{s.t.} \quad T_s\big(\tfrac{B_k}{R_k}\big) + T_k \le L_k, \quad \forall k \in \mathcal{S}', \tag{9b}$$

$$0 \le R_k, \quad \forall k \in \mathcal{S}', \tag{9c}$$

$$2^{\sum_{i\in\mathcal{N}'} R_i} \le 1 + \textstyle\sum_{i\in\mathcal{N}'} \alpha_i P_i, \quad \forall \mathcal{N}' \subseteq \mathcal{S}'. \tag{9d}$$

978-1-7281-0954-1/19/$31.00 ©2019 IEEE

To outline the closed-form solution to this problem that was obtained in [8], we first decompose the problem into an inner problem over the powers and an outer problem over the rates:

$$\min_{\{R_k\}_{k=1}^{K'}} \quad \text{s.t.} \quad (9b), (9c), \qquad \min_{\{P_k\}_{k=1}^{K'}} \quad \sum_{k\in\mathcal{S}'} \frac{T_s B_k}{R_k} P_k \quad \text{s.t.} \quad (9d). \tag{10}$$

For a fixed set of transmission rates, the inner problem in (10) is a linear programme in $\{P_k\}_{k=1}^{K'}$ with the feasibility region being a polyhedron. As such, it is sufficient to consider the vertices of the polyhedron, each of which is described by the simultaneous satisfaction of K' of the linear inequality constraints in (9d) with equality [8]. If $\rho_k \triangleq \frac{B_k}{\alpha_k R_k}$, and if π denotes the permutation such that

$$\rho_{\pi(K)} \leq \rho_{\pi(K-1)} \leq \cdots \leq \rho_{\pi(1)}, \tag{11}$$

exploiting the polymatroid structure of the achievable rate region of a FullMA scheme [18] enables us to obtain the closed-form optimal solution for the transmission powers as

$$P_{\pi(k)} = (\tfrac{2^{R_{\pi(k)}}-1}{\alpha_{\pi(k)}}) 2^{\sum_{j=1}^{k-1} R_{\pi(j)}}, \quad \forall k \in \mathcal{S}'. \tag{12}$$

With that solution in place, the outer problem in (10) can be written as

$$\min_{\{R_k\}_{k=1}^{K'}} \quad \sum_{k=1}^{K'} \frac{T_s B_{\pi(k)}}{R_{\pi(k)}} (\tfrac{2^{R_{\pi(k)}}-1}{\alpha_{\pi(k)}}) 2^{\sum_{j=1}^{k-1} R_{\pi(j)}} \tag{13a}$$

$$\text{s.t.} \quad \left(\tfrac{T_s B_k}{L_k - T_k}\right) \leq R_k, \quad \forall k \in \mathcal{S}'. \tag{13b}$$

Since the objective in (13a) is increasing in each transmission rate and the constraints are separable, the optimal rate for each user is the minimum feasible rate according to its latency constraint, namely, $R_k = (\frac{T_s B_k}{L_k - T_k})$. Having obtained the optimal transmission rates for all the users in the set $\mathcal{S}'$, the values of ρ_k can be sorted and the optimal transmission power of each user can be obtained using (12).

IV. Offloading Decision Making via Reinforcement Learning

Since we have a closed-form expression for the optimal resource allocation for a given set of offloading decisions, we now aim to develop an efficient algorithm to find good binary offloading decisions. In particular, we seek to find the mapping of the system parameters, namely, the channel gains, $\{h_k\}_{k=1}^K$, the latency constraints, $\{L_k\}_{k=1}^K$, and the problem description lengths, $\{B_k\}_{k=1}^K$, to the optimal set of offloading decisions, $\{x_k\}_{k=1}^K$, as

$$f : \mathbf{v} \to \mathbf{x}, \tag{14}$$

where $\mathbf{v} = [h_1, \ldots, h_K, L_1, \ldots, L_K, B_1, \ldots, B_K]$, and $\mathbf{x} = [x_1, \ldots, x_K]$. In this work, we develop such a mapping by employing an N-layer deep neural network, which is parametrized by

$$\boldsymbol{\theta} = \{\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2, \ldots, \mathbf{W}_N, \mathbf{b}_N\}, \tag{15}$$

where $\mathbf{W}_n$ and $\mathbf{b}_n$ denote the weights and the biases of the neurones in the n^{th} layer. If σ_n denotes the activation function in the n^{th} layer of the network, the mapping function of the algorithm can be written as

$$f_{\boldsymbol{\theta}}(\mathbf{v}) = \sigma_N(\mathbf{W}_N \sigma_{N-1}(\cdots \mathbf{W}_2 \sigma_1(\mathbf{W}_1 \mathbf{v} + \mathbf{b}_1) + \cdots) + \mathbf{b}_N).$$

While the activation functions at the internal layers, $\sigma_1, \sigma_2, \ldots, \sigma_{N-1}$, are design parameters of the network, in order to generate binary offloading decisions as the outputs, the activation function in the last layer of the DNN ought to be a binary thresholding function. However, the fact that the gradient of the binary thresholding function is zero almost everywhere, it precludes training the network. To address this issue, we use the sigmoid function as the activation function in the last layer of the neural network, i.e., $\sigma_N(x) = 1/(1+e^{-x})$. This results in an output vector that provides a soft offloading decision (the probability of offloading) for each user.

Conventional DNNs employ a supervised learning approach, in which a large number of previously labeled input-output data pairs is required. However, in the case of combinatorial problems, due to the complexity of finding the optimal combination for each input data sample, providing a large number of labeled data for training the DNN is computationally costly. In order to address this impediment, we propose an algorithm based on the RL framework, which avoids the requirement of labeled data to train the DNN.

The structure of the proposed algorithm is illustrated in Fig. 1. As shown in that figure, our proposed iterative algorithm contains two main phases. In the first phase, the network takes a random set of system parameters, $\mathbf{v}$, as an input and generates soft offloading decisions (between zero and one) as an output based on the characteristics of the DNN, i.e., the current value of $\boldsymbol{\theta}$. The generated soft offloading decisions (the offloading probabilities) are then used to generate different candidates for the offloading decisions. We obtain the energy consumption of the users for each candidate, and then we choose the candidate with the minimum energy consumption as the desired offloading decisions, $\mathbf{x}^\star$. The obtained pair of the system parameters-offloading decisions, $(\mathbf{v}, \mathbf{x}^\star)$, is then added to a memory of size D. After performing the first phase for D different data samples, in the second phase of the algorithm, the pairs of system parameters-offloading decisions in the memory are used to update the offloading policy by updating the parameters of the network, $\boldsymbol{\theta}$. By iterating between those two phases, the neural network continuously observes more samples, generates better offloading decisions for each sample, and finally improves its offloading policy. In the rest of this section, we will describe each of those phases of the proposed algorithm in more details.

A. Offloading Decision Generation

In the first phase of each iteration, the network takes D data samples and it aims to generate appropriate offloading decisions for each data sample. To do so, in the i^{th} iteration of the algorithm with the offloading policy of the network being $\boldsymbol{\theta}_i$, the system parameters of sample d, i.e., $\mathbf{v}_{i,d}$, are first mapped to the set of soft offloading decisions $\hat{\mathbf{x}}_{i,d}$ as

$$\hat{\mathbf{x}}_{i,d} = f_{\boldsymbol{\theta}_i}(\mathbf{v}_{i,d}), \quad d = 1, 2, \ldots, D. \tag{16}$$

978-1-7281-0954-1/19/$31.00 ©2019 IEEE

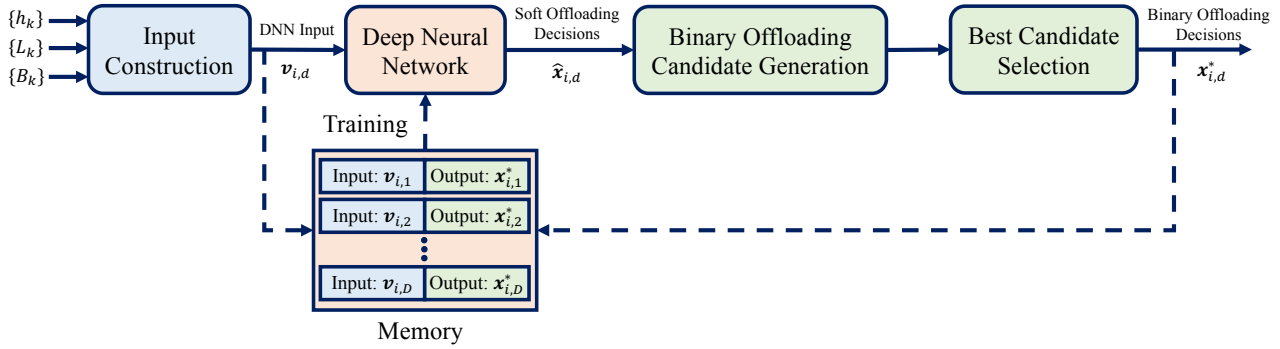

Fig. 1: The structure of the proposed RL-based algorithm to obtain offloading decisions in a binary offloading system.

Now, by utilizing the obtained soft offloading decisions, we can find the binary offloading decision vector. The choice of the approach to generating the binary offloading decisions involves a trade-off between the efficiency and the complexity of the algorithm. A straightforward approach is to round the soft offloading decisions in $\hat{\mathbf{x}}_{i,d}$ to binary decisions [11]. However, in that case the binary decisions for the users with soft decisions close to the rounding threshold, 0.5, may not be sufficiently accurate. In order to address this issue, rather than generating a single binary offloading decision vector, we consider a list $\mathcal{C}_{i,d}$ of size $M \in \mathbb{Z}^+$ of binary offloading decision candidates that are generated based on the soft offloading decisions in $\hat{\mathbf{x}}_{i,d}$.

There exist various techniques in the literature of different context to generate a list of candidates, with binary entries, based on a given probability vector for each element. As we will explain below, we will consider two different approaches, namely, the order-preserving quantization approach [13], and a bit-flipping approach, e.g., [19], [20]. Note that for ease of explanation, we will abuse the notation by dropping indices d and i for the generated candidates in the following subsections.

1) Order-Preserving Quantization: This approach has been introduced in [13] to generate a list of candidate for binary offloading decisions in a wireless-powered TDMA-based mobile edge network. The basic idea in this approach is to preserve the order of the probabilities (soft decisions) during quantization and only to change the quantization threshold based on which the binary decisions are made.

In this approach, each candidate, say candidate m, is generated based on a threshold, denoted by $x_{\text{th}}^{(m)}$. The first candidate is generated based on the natural choice of $x_{\text{th}}^{(0)} = 0.5$; the j^{th} entry of the first candidate, $\mathbf{x}_{\text{op}}^{(0)}$, is generated as follows

$$x_{\text{op}_j}^{(0)} = \begin{cases} 1, & \text{if } \hat{x}_j \geq 0.5, \\ 0, & \text{if } \hat{x}_j < 0.5, \end{cases} \tag{17}$$

for $j = 1, 2, \ldots, K$. In order to define the thresholds for generating the other candidates in the list, we let the permutation β determine the order of the entries of the vector $\hat{\mathbf{x}}$ with respect to the distance to 0.5, namely,

$$|\hat{x}_{\beta(1)} - 0.5| \leq |\hat{x}_{\beta(2)} - 0.5| \leq \cdots \leq |\hat{x}_{\beta(K)} - 0.5|, \tag{18}$$

and, we set the threshold for generating candidate m to be

$$x_{\text{th}}^{(m)} = \begin{cases} \hat{x}_{\beta(m)} + \epsilon, & \text{if } \hat{x}_{\beta(m)} > 0.5, \\ \hat{x}_{\beta(m)} - \epsilon, & \text{if } \hat{x}_{\beta(m)} \leq 0.5, \end{cases} \tag{19}$$

where ϵ is a small number that ensures that for each user there exists a candidate with a binary offloading decision different from the one obtained in generating the first candidate; see (17). Accordingly, up to $K + 1$ candidates can be generated by using this approach with candidate m being written as

$$x_{\text{op}_j}^{(m)} = \begin{cases} 1, & \text{if } \hat{x}_j \geq x_{\text{th}}^{(m)}, \\ 0, & \text{if } \hat{x}_j < x_{\text{th}}^{(m)}. \end{cases} \tag{20}$$

2) Bit Flipping: The bit flipping approach to generating binary candidates has been used in a variety of contexts, e.g., [19], [20]. The basis of this approach is the binary decision vector that is obtained by rounding the given probabilities (soft decisions). The list is then generated by flipping each of the entries of that vector. The rationale behind this approach is to explore broader range of possibilities for the offloading decisions, and, more importantly, to avoid one-bit error that may occur in generating the first candidate from the given probabilities.

If we let $\mathbf{x}_{\text{bf}}^{(0)} = \mathbf{x}_{\text{op}}^{(0)}$ denote the offloading decision vector obtained by rounding the soft decisions, then the K candidates are generated by flipping one of the entries of $\mathbf{x}_{\text{bf}}^{(0)}$ as

$$x_{\text{bf}_j}^{(m)} = \begin{cases} 1 - x_{\text{bf}_j}^{(0)}, & \text{if } j = m, \\ x_{\text{bf}_j}^{(0)}, & \text{otherwise.} \end{cases} \tag{21}$$

The list that we consider in this work contains all the candidates generated by each of the two approaches explained above. That is, we consider a list of $M = 2K + 1$ binary decision candidates, with $K + 1$ candidates generated by the order-preserving approach in Section IV-A1, and K candidates generated by the bit-flipping approach in Section IV-A2.

Having generated the list, in order to find solutions for the binary offloading decisions, we need to first find the minimum user energy consumption for each of the candidates in the list, $E_{\text{total}}(\mathbf{v}, \mathbf{x}^{(m)})$. That can be done by using the closed-form optimal solution explained in Section III. The candidate with

978-1-7281-0954-1/19/$31.00 ©2019 IEEE

the minimum user energy consumption would be then selected as solution for the binary offloading decisions, namely,

$$\mathbf{x}_{i,d}^{\star} = \arg \min_{\mathbf{x}_{i,d} \in \mathcal{C}_{i,d}} E_{\text{total}}(\mathbf{v}_{i,d}, \mathbf{x}_{i,d}). \tag{22}$$

We remark that by using the closed-form optimal solution for the total energy consumption in a system with given offloading decisions, explained in Section III, and by computing those energies in parallel for different candidates in the list, the step of selecting the best candidate in (22) can be performed with a reasonable computational cost.

B. Updating Policy

In the second phase of each iteration of the proposed RL-based algorithm, we use the D samples of system parameters-binary decisions pairs, $(\mathbf{v}_{i,d}, \mathbf{x}_{i,d}^{\star})$ $d = 1, 2, \ldots, D$, that have been written in the memory to train the DNN by updating the network parameter, $\boldsymbol{\theta}_i$, i.e., updating the offloading policy of the network.

The considered DNN can be trained by applying stochastic gradient descent (SGD) [21] on the set of all data samples in the memory. The loss function that we consider is the average cross entropy between the desired offloading decisions, $\mathbf{x}_{i,d}^{\star}$, obtained in the first phase of the current iteration, and the output of the DNN.

Training methods that are based on the SGD require the computation of partial derivative of the loss function with respect to all the (trainable) parameters of the network in order to update those parameters—a task that can be performed efficiently using back-propagation. The number of back-propagation steps that is performed on each batch of size D is a parameter that impacts the trade-off between the performance and the computational cost of training a DNN. Performing a large number of steps over a given batch of data from the first phase can result in over-fitting, whereas performing a small number of steps may not effectively extract the information available in the given batch of data.

C. Implementation Details

We implement the RL framework in Fig. 1 on TensorFlow which is an open source Python-based machine learning framework [22]. In our implementation, the number of hidden layers in the DNN is set to $N = 8$, while the number of hidden neurones in each layer is set to be $3K$. Further, we employ rectified linear units (ReLUs) as the activation function of the hidden layers. The size of the memory is considered to be $D = 2048$. The number of back-propagation steps for each batch is considered to be 50 for the first 1,000 iterations and then it is reduced to 10 for the rest of the iterations. In the second phase of the proposed algorithm, in which the DNN parameters need to be updated, we use a variant of the stochastic gradient descent method, called *Adam optimizer* [23] with a learning rate of 10^{-4}.

After the DNN is trained, the network is capable of generating soft offloading decisions that are ideally close to the optimal binary offloading decisions. Therefore, in the test phase, it is sufficient to generate a few binary offloading decision candidates from the output of the network (the soft offloading decisions), and to choose the candidate with the minimum energy consumption as the binary decisions. In our implementation, the number of candidates in the test phase is set to 2, and those candidates are generated based on the order-preserving approach explained in Section IV-A1.

V. Numerical Results

We will now illustrate the performance of the proposed RL-based algorithm for the binary computation offloading system. We will compare performance and computational cost of the proposed algorithm with those of three different algorithms developed in [7]. Before presenting those results, we first provide a brief explanation of each of the algorithms in [7].

Relaxation-Rounding: One approach to determining binary offloading decisions is to relax the binary constraints on the decisions, solve the resulting partial offloading problem, and then round the offloading fractions back to binary decisions.

Relaxation Randomized-Rounding: The relaxation-rounding technique can naturally be extended to randomized rounding, in which multiple candidates for the binary offloading decisions are generated based on independent Bernoulli distributions with the obtained offloading fraction in the case of partial offloading, $x_k \in [0, 1]$, being considered as the probability of offloading for user k. The candidate that leads to the minimum energy consumption is selected as the set of binary offloading decisions.

Greedy Search: An alternative approach is to view the offloading decisions as a decision tree and to employ the tailored greedy search algorithm introduced in [7]. That algorithm is initialized by the case in which all the users are scheduled to complete their tasks locally. In each step, the user for which offloading results in the maximum reduction of the total energy consumption of the users, is added to the set of offloading users. In addition, at each step the users for which offloading would impose higher energy consumption to the system than the local execution are removed from the searching space.

In our numerical experiments, we consider a FullMA-based offloading system with $K = 20$ users, each of which seeks to complete an indivisible computational task with the description length, B_k, randomly chosen from the set $[2, 3, 4] \times 10^6$(bits), within a specific latency constraint, L_k, randomly chosen from the set $[1.2, 2.2, 3.2]$(sec). We assume that the access point is equipped with sufficiently large computation resources so that t_{DL} and t_{exe} can be considered constant, and equal, for all the users. Accordingly, we set $T_k = t_{\text{exe}} + t_{\text{DL}} = 0.2$(sec) for all the users. We will also assume that the symbol rate of the channel is $1/T_s = 10^6$. The channel model in our numerical experiments is a slow-fading channel with the path-loss exponent equal to 3.7, and Rayleigh distributed small-scale fading. The noise variance is set to $\sigma^2 = 10^{-13}$. We consider a cell of radius $R = 500$m over which the users are uniformly distributed.

978-1-7281-0954-1/19/$31.00 ©2019 IEEE

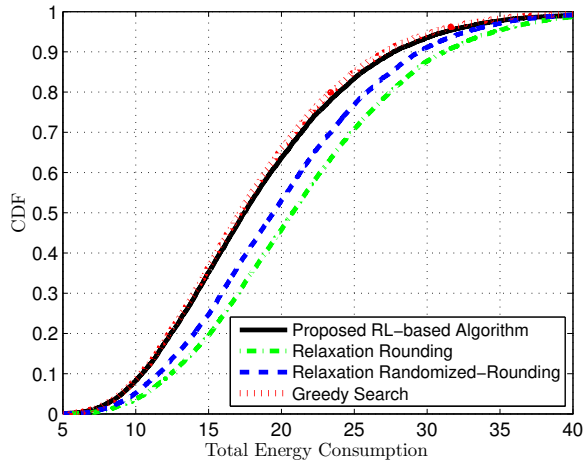

Fig. 2: The CDF of the energy consumption for different methods.

TABLE I: CPU times (in seconds) required for each algorithm in Fig. 2.

Algorithm	**CPU time**	**Norm. CPU time**
Proposed RL-based Algorithm	5.72	1.00
Relaxation Rounding	512.93	89.59
Relaxation Randomized-Rounding	556.42	97.19
Greedy Search	187.67	32.78

The empirical cumulative distribution function (CDF) of the energy consumption, obtained from 10^4 different system parameters, for the proposed RL-based algorithm and for the three existing algorithms explained above are illustrated in Fig. 2. Furthermore, the corresponding CPU time required by each of those algorithms to obtain the solution of the energy minimization problem is provided in Table I. The CDF of the energy consumptions in Fig. 2 and the required CPU times in Table I indicate that the proposed RL-based algorithm can achieve a better performance as compared to relaxation rounding and relaxation randomized-rounding approaches while it requires much lower CPU time. Moreover, it can be seen that the proposed RL-based algorithm can achieve a performance close to that of the greedy search algorithm while the required CPU time for that algorithm is about one order of magnitude less that that of the greedy search approach. Considering these numerical results, it can be concluded that the proposed RL-based algorithm can achieve a better trade-off between performance and computational cost as compared to the existing approaches in the literature.

REFERENCES

[1] 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, Aug. 2017.

[2] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, Mar. 2017.

[3] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Netw. Appl.*, vol. 18, no. 1, pp. 129–140, Feb. 2013.

[4] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 1, no. 2, pp. 89–103, June 2015.

[5] O. Muñoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.

[6] X. Cao, F. Wang, J. Xu, R. Zhang, and S. Cui, "Joint computation and communication cooperation for mobile edge computing," Oct. 2017. [Online]. Available: https://arxiv.org/abs/1704.06777v2

[7] M. Salmani and T. N. Davidson, "Uplink resource allocation for multiple access computational offloading (extended version)," Apr. 2019. [Online]. Available: https://arxiv.org/abs/1809.07453v2

[8] ——, "Multiple access binary computational offloading in the K-user case," in *Proc. IEEE Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, US, Oct. 2018, pp. 1599–1603.

[9] ——, "Energy-optimal computational offloading for simplified multiple access schemes," in *Proc. IEEE Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, US, Oct. 2017, pp. 1847–1851.

[10] ——, "Multiple access computational offloading: Communication resource allocation in the two-user case (extended version)," Oct. 2018. [Online]. Available: https://arxiv.org/abs/1805.04981v2

[11] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, June 2018.

[12] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for MEC," in *IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2018, pp. 1–6.

[13] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online offloading in wireless powered mobile-edge computing networks," Apr. 2018. [Online]. Available: https://arxiv.org/abs/1808.01977

[14] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1930–1941, Feb. 2019.

[15] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, 2016.

[16] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1784–1797, 2018.

[17] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, 2017.

[18] D. N. C. Tse and S. V. Hanly, "Multiaccess fading channels—Part I: Polymatroid structure, optimal resource allocation and throughput capacities," *IEEE Trans. Inf. Theory*, vol. 44, no. 7, pp. 2796–2815, 1998.

[19] M. Salmani, M. Nekuii, and T. N. Davidson, "Semidefinite relaxation approaches to soft MIMO demodulation for higher order QAM signaling," *IEEE Trans. Signal Process.*, vol. 65, no. 4, pp. 960–972, 2017.

[20] Y. Liu, X. Niu, and M. Zhang, "Multi-threshold bit flipping algorithm for decoding structured LDPC codes," *IEEE Commun. Lett.*, vol. 19, no. 2, pp. 127–130, 2015.

[21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[22] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," Mar. 2016. [Online]. Available: https://arxiv.org/abs/1603.04467

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Dec. 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

978-1-7281-0954-1/19/$31.00 ©2019 IEEE