2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress

Energy-efficient Autonomic Offloading in Mobile Edge Computing

Changqing Luo
Dept. of Electrical Engineering & Computer Science
Case Western Reserve University
Cleveland, OH 44106
Email: cxl881@case.edu

Sergio Salinas
Dept. of Electrical Engineering & Computer Science
Wichita State University
Wichita, KS 67260
Email: salinas@cs.wichita.edu

Ming Li
Dept. of Computer Science & Engineering
University of Nevada, Reno
Reno, NV 89557
Email: mingli@unr.edu

Pan Li
Dept. of Electrical Engineering & Computer Science
Case Western Reserve University
Cleveland, OH 44106
Email: lipan@case.edu

***Abstract*—The booming growth and popularity of mobile devices have led to the surge of various mobile applications. Many mobile applications, such as online video, gaming, are essentially computation-intensive, and hence can quickly deplete mobile devices' battery energy. To address this issue, academia and industry have proposed mobile edge computing (MEC) that can enable mobile devices to automatically offload computations to the edge servers located within the radio access networks of cellular operators. However, energy-hungry wireless communications incur extra energy consumption that may offset the energy saving due to computation offloading. To this end, we design an energy-efficient autonomic offloading scheme by jointly considering the physical layer design and application running latency. Specifically, we first mathematically model the energy consumption of a mobile application in MEC environment by taking into account the energy consumption incurred by the interactions among the tasks for the same application, which is largely ignored by previous studies. Then, we identify task execution flows based on a task interaction matrix, and formulate the maximum of the task flow's latencies as the application's latency. Finally, we formulate an energy-efficient offloading problem, which is generally NP-hard, and develop an efficient heuristic method to solve the problem. We present extensive simulation results to show that our proposed scheme can achieve significant reduction (up to 20% around) in energy consumption compared with previous schemes.**

1. Introduction

The booming growth and popularity of mobile devices like smartphones and tablets have resulted in the surge of various mobile applications. Many mobile applications like online video, virtual reality, and interactive gaming are typically computation-intensive, thus resulting in very high energy consumption [1]. Mobile users tend to enjoy such mobile applications much longer and more often than before, which can deplete their mobile devices' battery energy very quickly.

To overcome this problem, industry and academia have recently proposed mobile edge computing (MEC) which enables mobile devices to automatically offload computations to the edge servers located within the radio access networks, instead of the core network, of cellular operators [2]. In MEC, mobile devices may reduce their energy consumption and in the meanwhile experience lower latency compared with that in cloud computing.

However, we notice that wireless communications are energy-hungry. Thus, outsourcing computations to edge servers can incur extra energy consumption due to data transmission over wireless links [3], which may even offset the energy saving due to computation offloading. In particular, a mobile application is generally composed of a number of tasks. Some of them are computation-intensive but data-light, while others may be computation-light but data-intensive [4]. A mobile device consumes a huge amount of energy when it locally processes computation-intensive but data-light tasks, or offloads computation-light but data-intensive tasks to edge servers. Therefore, when designing our energy-efficient autonomic offloading scheme, we need to carefully determine which tasks should be offloaded to edge servers.

Some previous works have attempted to reduce the energy consumption of offloading computations in MEC. Chen *et al.* [1] study a multi-user computation offloading problem in a multi-channel wireless interference environment. The distributed computation offloading decision-making problem is formulated as a multi-user computation offloading game. Zhang *et al.* [5] develop an energy-efficient computation offloading mechanism for MEC in 5G heterogeneous networks. Sardellitti *et al.* [6] propose to reduce the energy consumption by jointly considering the radio resources and computational resources. Mao *et al.* [3] exploit renewable energy to help reduce the energy consumption of mobile devices. However, such works have not considered the influence of physical-layer design like the adaptive modulation and coding schemes.

In this paper, we design an energy-efficient autonomic offloading scheme that can automatically offload computational tasks to edge servers. Specifically, we aim to minimize the total amount of energy consumption of a certain application running on a mobile device. In particular, we notice that an application is usually composed of a number of computation tasks, some of which are dependent on data input from others. Energy consumption due to task interactions has been largely overlooked in previous studies. Thus, in this study, we consider that the energy consumption of an application on a mobile device is incurred by local task computation, wireless communications in task offloading, and wireless communications in task interactions. Moreover, considering that the physical layer design like the adaptive modulation and coding (AMC) scheme and radio resources allocation has great impact on energy consumption of offloading transmission [7], we also investigate the physical layer design in our proposed scheme. In addition, due to mobile users' quality of experience (QoE) demand, we consider a constraint on the latency of a mobile application. Consequently, we formulate the energy-efficient autonomic offloading problem as a joint physical layer design and computation task assignment problem with constrained latency and limited radio resources. We then solve this problem by developing a heuristic algorithm. Finally, we show by extensive simulation results that the proposed scheme can achieve significant reduction in energy consumption.

The key contributions of this paper are summarized as follows:

1) We design an energy-efficient autonomic offloading scheme for MEC by jointly considering physical layer design and application latency, which can achieve up to 20% around reduction in energy consumption compared with previous schemes.
2) We develop three energy consumption models for task computation, offloading transmission, and task interaction, respectively, the last of which is largely ignored by previous studies.
3) We construct a directed graph based on a task interaction matrix to identify task execution flows, and in turn formulate the application latency.
4) We develop an efficient heuristic algorithm to solve the formulated optimization problem.

The rest of this paper is organized as follows. In Section 2, we describe a mobile application model and mobile edge computing framework. Section 3 presents the mathematical energy consumption models for task computation, task offloading, and task interaction, respectively. Section 4 details the formulation of the energy-efficient autonomic offloading problem and the solution to the formulated problem. We present the simulation results in Section 5, and finally conclude this paper in Section 6.

2. System Model

In this paper, we consider a typical MEC framework as shown in Fig. 1, where a mobile device offloads parts of an application to an edge server via wireless communications.

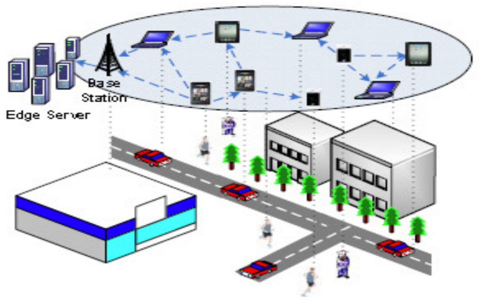


Figure 1. The MEC framework.

2.1. Mobile Application Model

In practice, a mobile device runs applications in the granularity of computing units, i.e., computation tasks. Suppose that a certain application running on a mobile device is composed of N computation tasks represented by $\mathcal{N} = \{1, \cdots, i, \cdots, N\}$, where i denotes task i that is usually defined by a 2-tuple $< L_i, D_i >$. Here L_i, i.e., the workload, is the number of CPU cycles and D_i, i.e., the data size, is the number of bits. Both L_i and D_i are typically used to represent the computation required to complete task i. The relationship between them is that for a specific data size, its corresponding workload is a random variable with an empirical distribution [8].

In particular, to run an application, some computation tasks are often dependent on data input from others, which is called as task interactions [9]. Let $\boldsymbol{B}_{int} = [b_{i',i}]_{N\times N}$ denote a matrix indicating whether task interaction occurs between two tasks, where $b_{i',i}$, a binary variable, indicates whether task interaction occurs between tasks $i^{'}$ and i. $b_{i',i} = 1$ implies that task i needs data from task $i^{'}$. It is noteworthy that we set $b_{i',i} = 0$ when $i = i^{'}$. The computation due to task interaction is also represented by a 2-tuple $< L_{i',i}, D_{i',i} >$, where $L_{i',i}$ denotes the additional workload that needs to be processed by task i and $D_{i',i}$ is the additional data bits that needs to be sent from tasks $i^{'}$ to i.

2.2. Mobile Edge Computing Framework

In MEC, mobile devices and edge servers execute tasks locally and remotely, respectively. In this paper, we consider virtual machines as computational resources that are used to execute tasks [10] and particularly one virtual machine is only able to execute one task at a time.

For a specific application running on a mobile device, the mobile device needs to conduct the autonomic scheduling for task offloading and transmit the computation tasks to be executed remotely to the edge servers located within radio access networks. Let a_i be a binary variable to indicate where task i is executed. $a_i = 0$ means that task i is executed remotely and $a_i = 1$ implies that task i is processed locally. We represent the number of tasks to be executed locally by N_{md} and remotely by N_{cc}, respectively, and we have

$$N_{md} = \sum_{i=1}^{N} a_i, \text{ and } N_{cc} = \sum_{i=1}^{N} \bar{a}_i, \tag{1}$$

where $\bar{a}_i = 1 - a_i$.

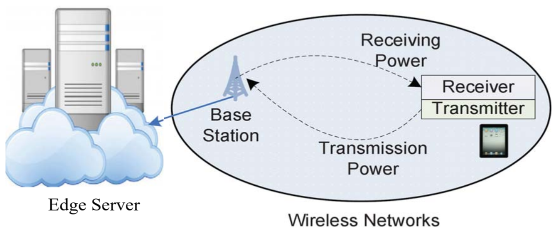


Figure 2. A simplified network scenario for offloading transmission.

To offload computation tasks to the edge servers, wireless communications need to be used in MEC, as shown in Fig. 2. We notice that wireless communications are also energy-hungry. The energy consumption of wireless communications usually depends on physical layer design, i.e., the AMC rate and the radio resources (i.e., transmission power and bandwidth) at the physical layer. However, in wireless communications, the AMC rate and the radio resources are usually limited. In the following, we show the constraints on AMC rate, transmission power, and transmission bandwidth, respectively.

Due to the capability of mobile devices, the AMC rate is constrained. Let ρ_{max} denote the maximal AMC rate that a mobile device can provide. Thus, to offload computation task i and the additional computation due to task interaction between tasks $i^{'}$ and i, the used AMC rates need to satisfy:

$$0 \le \rho_i, \rho_{i^{'},i} \le \rho_{max}, \tag{2}$$

where ρ_i is the AMC rate for offloading computation task i and $\rho_{i^{'},i}$ is the AMC rate for offloading the additional computation incurred by task interaction between tasks $i^{'}$ and i. On the other hand, due to the essentially limited radio resources in wireless networks, the transmission power and bandwidth are constrained as well. Moreover, due to the features of radio links, the computation tasks to be offloaded or those involved in task interaction share the transmission bandwidth offered by a wireless link and the transmission power provided by a base station or a mobile device. We consider that all task transmissions are allocated the identical radio resources. In addition, to avoid transmission collision, the number of virtual machines at a mobile device, denoted by N_w, is equal to the number of task execution flows. Let P_{max} denote the maximum transmission power, and B_{max} the maximum transmission bandwidth. Therefore, for each wireless channel, the maximum transmission power is $P_{max}^{vm} = \frac{P_{max}}{N_w}$, and the maximum transmission bandwidth $B_{max}^{vm} = \frac{B_{max}}{N_w}$. The transmission power for offloading task i and the computation incurred by task interaction between tasks $i^{'}$ and i are denoted by P_i and $P_{i^{'},i}$, respectively, and we have

$$0 \le P_i, P_{i^{'},i} \le P_{max}^{vm}. \tag{3}$$

Similarly, the transmission bandwidth for offloading task i and the additional computation for task interaction between tasks $i^{'}$ and i are denoted by B_i and $B_{i^{'},i}$, respectively, and we have

$$0 \le B_i, B_{i^{'},i} \le B_{max}^{vm}. \tag{4}$$

3. Modeling and Analysis of Mobile Device's Energy Consumption

In this section, we present the energy consumption models for task computation, offloading transmission over wireless connections, and task interaction, respectively.

3.1. Energy Consumption for Task Computation

The amount of energy consumption incurred by executing computation tasks locally depends on the workload and the used clock frequency of a CPU (central processing unit). Particularly, the energy consumption for each operation at a virtual machine is proportional to f^2 [11], where f is the clock frequency of the virtual machine. Thus, we can calculate the energy consumption as follows:

$$f(L_i, f) = k_{dev} \cdot f^3 \cdot t(L_i, f), \tag{5}$$

where k_{dev} is the energy coefficient depending on the chip architecture [12], and $t(L_i, f)$ is the execution time that can be given by

$$t(L_i, f) = \frac{L_i}{f}. \tag{6}$$

3.2. Energy Consumption for Offloading Transmission

When task i is offloaded to the edge servers over a wireless link, wireless communications consume the energy. To find the energy consumption incurred by offloading transmission, we first obtain the bit error rate suffered by the wireless link as follows [13]:

$$BER_i = k_1 \cdot \exp(-\frac{k_2 \gamma_i P_i}{2^{\rho_i} - 1}), \tag{7}$$

where BER_i is the bit error rate, γ_i is the signal-to-noise ratio (SNR) of the wireless link used for transmitting task i, ρ_i is the used AMC rate determined by the modulation rate ρ_i^{mod} and the channel coding rate ρ_i^{cod}, i.e., $\rho_i = \rho_i^{mod} \cdot \rho_i^{cod}$ [13], and k_1 and k_2 are constants related to the specific constellations and codes, respectively.

Subsequently, we can obtain the corresponding frame error probability, denoted by FE_i, as follows:

$$FE_i = 1 - (1 - BER_i)^{L_{fr}}, \tag{8}$$

where L_{fr} is the length of a frame.

Due to the frame error, a frame possibly needs to be retransmitted. In a basic ARQ scheme, a source successfully transmits a frame only if the number of retransmission trials reaches a pre-defined threshold N_{re}. Thus, the expected number of transmissions for one frame, denoted by $N_{ex}(\rho_i, P_i)$, can be derived as follows:

$$\begin{aligned} N_{ex}(\rho_i, P_i) &= \{1 + 2FE_i + \cdots + N_{re} \cdot FE_i^{N_{re}-1}\}(1 - FE_i) \\ &+ (N_{re} + 1) \cdot FE_i^{N_{re}} = (1 - FE_i^{N_{re}+1})/(1 - FE_i). \end{aligned} \tag{9}$$

Therefore, given ρ_i, P_i, and B_i for transmitting task i, the expected transmission time, denoted by $t_{tr}(D_i, \rho_i, P_i, B_i)$, can be approximated by

$$t_{tr}(D_i, \rho_i, P_i, B_i) \approx \frac{D_i}{(L_{fr} - L_{frh})} \cdot \frac{L_{fr} \cdot N_{ex}(\rho_i, P_i)}{R(\rho_i, B_i)}, \tag{10}$$

where L_{frh} is the length of a frame header and $R(\rho_i, B_i)$ is the transmission rate that is determined by the AMC scheme and the transmission bandwidth, i.e., we have $R(\rho_i, B_i) = \rho_i \cdot B_i$. Finally, we can find the amount of energy consumption, denoted by $f_{tr}(D_i, \rho_i, P_i, B_i)$, as follows:

$$f_{tr}(D_i, \rho_i, P_i, B_i) = P_i \cdot t_{tr}(D_i, \rho_i, P_i, B_i). \tag{11}$$

3.3. Energy Consumption for Task Interaction

Some of computation tasks forming an application often need to data input from others, i.e. task interaction, which incurs the additional energy consumption at the mobile device. The amount of energy consumption incurred by task interaction between two tasks usually depends on the locations where two tasks are. Specifically, if both tasks are executed locally, or if a task at a mobile device requests data input from a task in the edge servers, the energy consumption is incurred by conducting task computation locally. Moreover, if a task in the edge servers requests data input from a task at the mobile device, the energy consumption is incurred by wireless communications.

Based on the above analysis, we calculate the energy consumption for task interaction in the following. When task i is executed at the mobile device and needs data input from task $i^{'}$, the amount of energy consumption, denoted by $E^{md}_{i^{'},i}$, can be calculated as $E^{md}_{i^{'},i} = f(L_{i^{'},i}, f_{md})$, where f_{md} is the clock frequency of a virtual machine at the mobile device. When task i is executed in the edge servers and requests data from task $i^{'}$ executed at the mobile device, the amount of energy consumption, denoted by $E^{tr}_{i^{'},i}$, can be calculated as $E^{tr}_{i^{'},i} = f_{tr}(D_{i^{'},i}, \rho_{i^{'},i}, P_{i^{'},i}, B_{i^{'},i})$. When two tasks are both located in the edge servers, the mobile device has no energy consumption. Thus, the amount of energy consumption incurred by task i interacting with all other tasks, denoted by E^{tt}_i, can be calculated as follows:

$$E^{tt}_i = \sum_{i^{'}=1}^{N} [E^{md}_{i^{'},i} \cdot a_i + E^{tr}_{i^{'},i} \cdot (a_{i^{'}} \cdot \bar{a_i})]. \tag{12}$$

4. Energy-efficient Task-level Offloading Scheme

In this section, we formulate the energy-efficient autonomic offloading problem as a mobile device's energy consumption minimization problem. Particularly, this optimization problem is subject to AMC scheme, radio resources, and user-defined application latency. To solve the formulated problem, we propose an efficient heuristic method. In what follows, we describe the problem formulation and the solution to the formulated problem, respectively.

4.1. Problem Formulation

Our objective is to minimize a mobile device's total energy consumption when it runs a certain application in MEC. In particular, we take into account the energy consumption for task computation, wireless communications in task offloading, and wireless communications in task interaction. Moreover, we explore physical layer design and computation task assignment when we optimize the mobile device's energy consumption. In addition, we consider the application running latency constraint as the user-defined QoE demand.

If task i is executed locally, the amount of energy consumption, denoted by E^{md}_i, is $E^{md}_i = f(L_i, f_{md})$. Moreover, if task i is offloaded over a wireless connection to the edge servers, the amount of energy consumption, denoted by E^{tr}_i, is $E^{tr}_i = f_{tr}(D_i, \rho_i, P_i, B_i)$. Thus, we can obtain the total amount of energy consumption incurred by task i, denoted by E_i, as follows:

$$E_i = E^{md}_i \cdot a_i + E^{tr}_i \cdot \bar{a_i} + E^{tt}_i. \tag{13}$$

Finally, we can find the total amount of energy consumption at the mobile device for running an application, denoted by E, as follows:

$$E = \sum_{i=1}^{N} E_i. \tag{14}$$

On the other hand, to satisfy a mobile user's QoE demand, a certain application needs to be executed completely before a user-defined latency T. To characterize the application running latency, we first construct a directed graph, as shown in Fig. 3, indicating the task execution sequence, where nodes present the tasks and edges indicate the task execution order. Note that this graph can be constructed by matrix power operation B^n_{int} ($n \in \{1, 2, \cdots, N\}$), where n is the number of power, indicating the number of hops from one node arriving at another one. Thus, the application running latency can be obtained by finding each task execution flow's latency. Specifically, we denote the set of task execution flows by $\mathcal{F}$, and have $N_w = |\mathcal{F}|$. Let A_{i,F_j} be a binary variable, and $A_{i,F_j} = 1$ indicates that task i is on task execution flow F_j ($F_j \in \mathcal{F}$).

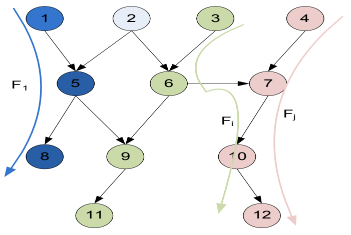


Figure 3. An example of a directed graph constructed from a task interaction matrix.

To find each task execution flow's running latency, we first derive the time spent for executing a task. For task i on task execution flow F_j, its processing time, denoted by t_i, depends on the task's and other interacting tasks' locations. Note that, if both tasks are executed at a mobile device or in the edge servers, their communication delay is negligible.

Specifically, we consider t_i in following four folds. First, if task i is running by a virtual machine at a mobile device, t_i^{md} can be calculated as $t_i^{md} = t(L_i, f_{md})$. Second, if task i is running by a virtual machine in the edge servers, the processing time includes the transmission delay over a wireless channel and the remote execution time. Thus, we can calculate t_i^{cc} as $t_i^{cc} = t_{tr}(D_i, \rho_i, P_i, B_i) + t(L_i, f_{cc})$, where f_{cc} is the clock frequency of a virtual machine in the edge servers. Third, if task i at a mobile device interacts with task $i^{'}$ in the edge servers, the processing time consists of the transmission delay over the wireless channel and the local execution time. Hence, we can derive $t_{i^{'},i}^{mc}$ as $t_{i^{'},i}^{mc} = t_{tr}(D_{i^{'},i}, \rho_{i^{'},i}, P_{i^{'},i}, B_{i^{'},i}) + t(L_{i^{'},i}, f_{md})$. Finally, if task i in an edge server interacts with task $i^{'}$ at a mobile device, the processing time is composed of the transmission delay over the wireless channel and the remote execution time. Therefore, we can calculate $t_{i^{'},i}^{cm}$ as $t_{i^{'},i}^{cm} = t_{tr}(D_{i^{'},i}, \rho_{i^{'},i}, P_{i^{'},i}, B_{i^{'},i}) + t(L_{i^{'},i}, f_{cc})$.

Based on the above discussion, we are able to calculate task i's total processing time, denoted by t_i, as follows:

$$t_i = \sum_{i^{'}=1}^{N} [t_{i^{'},i}^{mc} \cdot (a_i \cdot \bar{a_{i^{'}}}) + t_{i^{'},i}^{cm} \cdot (\bar{a}_i \cdot a_{i^{'}})] \cdot b_{i^{'},i} + t_i^{md} \cdot a_i + t_i^{cc} \cdot \bar{a}_i. \tag{15}$$

Subsequently, we can obtain the task execution flow latency along task execution flow F_j, denoted by T_{F_j}, as follows:

$$T_{F_j} = \sum_{i=1}^{N} t_i \cdot A_{i,F_j}. \tag{16}$$

Here, T_{F_j} is bounded by T, i.e.,

$$0 < T_{F_j} < T. \tag{17}$$

Therefore, we can formulate the energy-efficient autonomic offloading problem as follows:

$$\begin{aligned} \textbf{MCO:} \quad & \textbf{min } E, \\ & \textbf{s.t. } \text{Eqs. } (2)-(4), (17), \\ & \quad a_i, a_{i'} \in \{0,1\}, \\ & \quad i, i^{'} \in \mathcal{N}, \\ & \quad F_j \in \mathcal{F}. \end{aligned} \tag{18}$$

4.2. The Solution to The Energy-Efficient Offloading Problem

We can see that **MCO** is a quadratically constrained mixed-integer quadratic programming problem, which is generally NP-hard [14]. To efficiently find a solution to **MCO**, we propose an efficient heuristic algorithm. Specifically, we first randomly fix the binary variables a_i's to reformulate **MCO** as **MCO-PHY** that is solved by employing a KKT Lagrangian multiplier method. Then, we fix the physical layer parameters by using the derived optimal physical layer parameters to reformulate **MCO** as **MCO-BV** that can be solved by the off-the-shelf DP solver. In the following, we describe our proposed algorithm.

To find the solution to **MCO**, we first randomly fix the binary variables a_i's to reformulate **MCO** as the transmission energy consumption optimization problem, i.e., **MCO-PHY**, as follows:

$$\begin{aligned} \textbf{MCO-PHY:} \quad & \textbf{min } E_{fb}, \\ & \textbf{s.t. } \text{Eqs. } (2)-(4), (17), \\ & \quad i, i^{'} \in \mathcal{N}, \\ & \quad F_j \in \mathcal{F}, \end{aligned} \tag{19}$$

where E_{fb} represents the total amount of energy consumption in **MCO-PHY**.

To solve **MCO-PHY**, we employ KKT Lagrangian multiplier method that extends the unconstrained first-order condition to the case of inequality constraints. To setup the KKT, we build up the Lagrangian by adding to E_{fb} inequality constraints each with a Lagrange-like multiplier [15]. The KKT Lagrangian is expressed as Eq. (20). In Eq. (20), λ_{F_j}, α_i, $\alpha_{i^{'},i}$, β_i, $\beta_{i^{'},i}$, ϵ_i, and $\epsilon_{i^{'},i}$ are the Lagrangian multipliers. By solving this equation, we can find the optimal parameters, i.e., ρ_i^*, $\rho_{i^{'},i}^*$, P_i^*, $P_{i^{'},i}^*$, B_i^*, and $B_{i^{'},i}^*$.

Then, we fix the physical layer parameters that were derived by solving **MCO-PHY** to reformulate **MCO** as **MCO-BV**:

$$\begin{aligned} \textbf{MCO-BV:} \quad & \textbf{min } E_{fp}, \\ & \textbf{s.t. } \text{Eq. } (17), \\ & \quad a_i, a_{i'} \in \{0,1\}, \\ & \quad i, i^{'} \in \mathcal{N}, \\ & \quad F_j \in \mathcal{F}, \end{aligned} \tag{21}$$

where E_{fp} represents the total amount of energy consumption in **MCO-BV**.

$$L(\rho_i, \rho_{i',i}, P_i, P_{i',i}, B_i, B_{i',i}, \lambda_{F_j}, \alpha_i, \alpha_{i',i}, \beta_i, \beta_{i',i}, \epsilon_i, \epsilon_{i',i}) = E_{fb} + \sum_{F_j=1}^{|\mathcal{F}|} \lambda_{F_j} (\sum_{i=1}^{N} t_i A_{i,F_j} - T) + \sum_{i'=1}^{N} \sum_{i=1}^{N} \alpha_{i',i}(\rho_{i',i} - \rho_{max})$$
$$+ \sum_{i=1}^{N} \alpha_i(\rho_i - \rho_{max}) + \sum_{i=1}^{N} \beta_i(P_i - P_{max}^{vm}) + \sum_{i'=1}^{N} \sum_{i=1}^{N} \beta_{i',i}(P_{i',i} - P_{max}^{vm}) + \sum_{i=1}^{N} \epsilon_i(B_i - B_{max}^{vm}) + \sum_{i'=1}^{N} \sum_{i=1}^{N} \epsilon_{i',i}(B_{i',i} - B_{max}^{vm}). \quad (20)$$

We can see that **MCO-BV** is a quadratically constrained binary quadratic programming problem, which is generally NP-hard [16]. Generally, such a problem can be solved by using the off-the-shelf DP solver [17]. Thus, we find the solution, i.e., a_i's, to **MCO-BV** by DP solver in this paper.

To sum up, we solve **MCO** by separately solving **MCO-PHY** and **MCO-BV** iteratively until the optimal physical layer design and task assignment policy are obtained.

5. Performance Evaluation

In this section, we present the simulation settings and the simulation results, respectively.

5.1. Simulation Settings

To validate the efficacy of the proposed scheme, we carry out extensive simulations. Simulations are conducted by using CPLEX 12.4 and Java on a Laptop with a dual-core 2.7 GHz CPU and 8GB RAM memory. We aim at demonstrating the performance improvement over other two previous schemes (i.e., one without considering task interaction and the other without considering the physical layer design), and showing the impact of physical layer design.

In the simulations, we consider a square network of area $500m \times 500m$, with a base station located at the center. We assume that the base station can provide mobile devices with the transmission bandwidth of 60 MHz and the transmission power of the base station is 1 W. Since an edge server located in this access network is very close to the base station, we set the transmission delay is zero in the simulations.

Moreover, we set the parameters of the wireless link as follows. First, we set the transmission power of a mobile device as 400 mW. Then, to evaluate the impact of the physical layer design on energy consumption, we use quadrature PSK (QPSK), eight PSK (8PSK), and 16 QAM as the modulation scheme and $3/4$ Turbo code as the coding scheme. The maximal AMC rate is 3 and the SNR is varied within the range $[0dB, 30dB]$. At last, we set the parameters of the data frame as follows: the maximum length of a frame is 1120 $bits$ and the maximal number of retransmissions for each frame is 5.

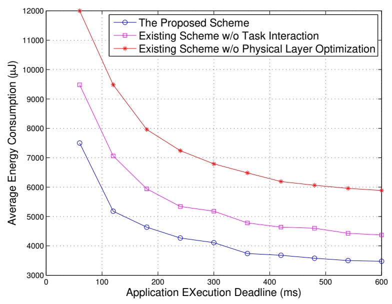
Moreover, we consider an application that is composed of 400 tasks. For simplicity, each task has the same profile, i.e., $D = 600$ $bytes$ and L obeying the Gaussian distribution with $(6000, 100)$. Besides, the sign indicating the interaction between two tasks, $b_{i',i}$ (for each $i, i' \in \mathcal{N}$, except $i = i'$), is randomly set as 1 or 0. When $b_{i',i} = 1$, the corresponding $D_{i',i} = 60$ $bytes$ and $L_{i',i}$ is considered as a Gaussian distribution with $(600, 10)$. Besides, we assume that this application has to be completed within 200 ms.

The mobile device has to offload its computations to an edge server via wireless communications. The clock frequencies of the mobile device and the edge serve are set as $f_{md} = 10$ MHz and $f_{cc} = 50$ MHz, respectively. The energy coefficient, k_{dev}, is $0.344 \cdot 10^{-9}$ [18].

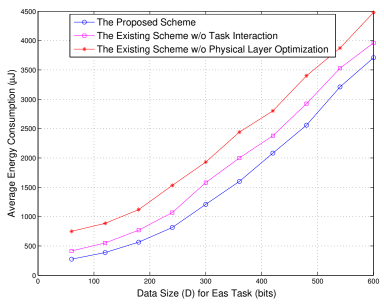
5.2. Performance Comparison

Fig. 4 shows the comparison of the energy consumption at the mobile device among our proposed scheme, the one without considering task interaction, and the one without considering physical layer design. Besides, this figure also demonstrates the impact of the application running latency, the task data size, and the data size of the task interaction on the energy consumption.

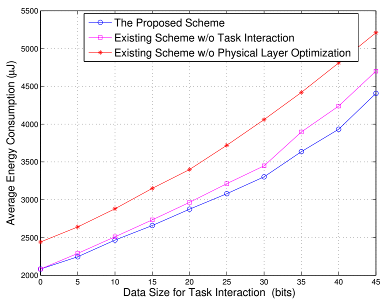
Specifically, Fig. 4(a) illustrates the mobile device's energy consumption as the predefined application running delay changes. From this figure, we can see that the mobile device's energy consumption decreases with the increase in the application execution deadline. The reason is that some tasks may cost high energy consumption and low execution delay at the mobile device but low energy consumption and high execution delay for remote execution. Thus, extending the application running delay makes these tasks can be offloaded into the edge servers in order for the further energy consumption reduction. Moreover, the proposed scheme can achieve the lowest energy consumption. For example, when the application running latency is 300 ms, the amounts of energy consumption for the proposed scheme, the one without considering task interaction, and the one without considering physical layer design are 4100 J, 5200 J, and 6900 J around, respectively. We can see the the reduction in energy consumption is up to 20% around. This means that task interaction and physical layer design do need to be carefully considered in the energy consumption minimization. Furthermore, the scheme considering physical layer design but without task interaction can lead to higher energy consumption than the proposed scheme. This is because task interaction can affect the task offloading policy. As a result, the mobile device's energy consumption is increased. In addition, the scheme considering task interaction but without physical layer design has the highest energy consumption.



(a) The execution deadline.



(b) The data size of each task.



(c) The exchanged data size for task interaction.

Figure 4. Performance improvement by the proposed scheme.

This curve indicates that the physical layer design has significant impact on the mobile device's energy consumption.

Fig. 4(b) shows mobile device's energy consumption as the data size of each task changes. We find that the mobile device's energy consumption is increasing with the increase in data size, and especially the growth rate is monotonously increasing. The reasons are as follows. Firstly, the increase in data size of each task definitely results in rising energy consumption; Secondly, the physical layer settings for wireless communications have to be adjusted to execute the tasks within their predefined execution deadline, and thus the energy consumption may be raised. Third, since the energy consumption for offloading transmission is increasing, the task assignment policy would be changed to minimize the total energy consumption. Moreover, from Fig. 4(b), we can also see that three curves corresponding to the three schemes has the same relationship as shown in Fig. 4(a). Thus, we have the insight that the energy consumption for offloading transmission can greatly affect the task assignment policy.

Fig. 4(c) shows mobile device's energy consumption as the data size of task interaction rises. From Fig. 4(c), we can notice the similar curves to that in 4(b). The reason is also the same as that of increasing the data size. From this figure, we can see that the task interaction has a significant impact on the task assignment policy.

5.3. Effects of Physical Layer Design Parameters on Energy Consumption

Fig. 5 show the impact of the AMC scheme, transmission power, and transmission bandwidth on the energy consumption due to wireless communications.

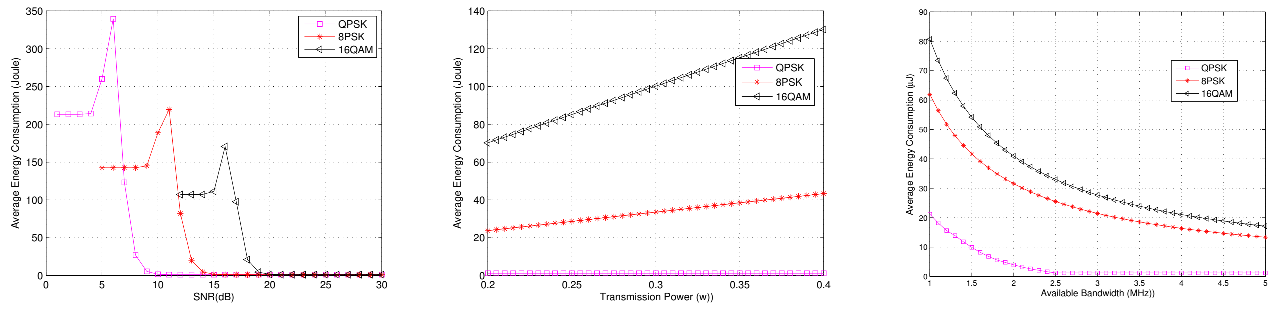
Specifically, Fig. 5(a) demonstrates the energy consumption under different AMC schemes as the channel gain increases. From this figure, we can see that the three curves under the different AMC schemes have the similar trends. Specifically, the energy consumption approaches a constant when the channel condition is bad. This is because no information can be transmitted successfully to the receiver and the energy consumption is caused due to the transmission attempts. Then, a lower-order AMC scheme can first achieve a sharp increase in energy consumption as the channel gain increases, then the sharp decreasing, and finally the stable curve. The reason is that some portion of information can be sent to the receiver but the average number of retransmissions are very high. As a consequence, much more energy is consumed. Another observation is that the energy consumption decreases as the channel gain increases. This is because the lower BER leads to the lower average number of retransmissions, and thus the energy consumption is decreasing. In particular, if the channel condition is very good, the information can nearly be sent to the receiver at a time. Thus, all AMC schemes can achieve the same energy consumption. More importantly, this figure shows us that AMC schemes at the physical layer have to be well considered to minimize energy consumption.

Fig. 5(b) illustrates the energy consumption under different AMC schemes as transmission power increases. From this figure, we can find that the energy consumption is increasing as the transmission power increases. This is because the increasing transmission power would potentially lead to the growing energy consumption. Moreover, the lower-order AMC scheme has the lower energy consumption. The reason is that the lower-order AMC scheme can achieve the lower bit error rate and then the lower energy consumption when the channel is at a certain state. This figure shows us that whether the increase in the transmission power may lead to the increasing energy consumption depends on the channel condition.

Fig. 5(c) depicts the energy consumption under different AMC schemes as the available bandwidth increases. We can notice that the energy consumption is degrading with the increase in the transmission bandwidth. The more transmission bandwidth, the lower transmission delay. Thus, the energy consumption may be reduced. Moreover, similar to the curves in Fig. 5(b), the lower-order AMC scheme can achieve lower energy consumption. This figure gives us an insight that the transmission bandwidth has a significant influence on the energy consumption.

6. Conclusions

In this paper, we have studied an energy consumption minimization problem for autonomic offloading in MEC. To address this problem, we have designed an energy-efficient



(a) Energy consumption vs. modulation and coding schemes. (b) Energy consumption vs. transmission power. (c) Energy consumption vs. available bandwidth.

Figure 5. Effects of physical layer settings on the energy consumption.

offloading scheme by jointly considering physical layer design and application running latency. Specifically, we first have mathematically modeled the energy consumption of a mobile application by considering the energy consumption incurred by the interactions among the tasks for the same application. We then have constructed a directed graph based on a task interaction matrix by calculating k-hop connectivity to identify task execution flows, and formulated each task execution flow's latency. Finally, we have formulated the energy consumption minimization problem as a quadratically constrained integer-mixed quadratic programming problem, which is generally NP-hard. To solve the formulated problem, we have proposed an efficient heuristic method. To validate the efficacy of the proposed scheme, we have offered extensive simulation results to show the significant reduction in energy consumption.

References

[1] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Networking*, vol. 24, pp. 2795–2808, Oct. 2016.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, pp. 637–646, Oct. 2016.

[3] Y. Mao, J. Zhang, and K. Lataief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Select. Areas Commun.*, vol. PP, pp. 1–16, Sep. 2016.

[4] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, "To offload or not to offload? The bandwidth and energy costs of mobile cloud computing," in *Proc. IEEE INFOCOM'13*, (Turin, Italy), pp. 1285–1293, Apr. 14–19 2013.

[5] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, Aug. 2016.

[6] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimisation of radio and computational resources for multiple mobile-edge computing," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 1, pp. 89–103, Jun. 2015.

[7] C. Luo, L. T. Yang, P. Li, X. Xie, and H. Chao, "A holistic energy optimization framework for cloud-assisted mobile computing," *IEEE Wireless Commun. Mag.*, vol. 22, pp. 118–123, Jun. 2015.

[8] J. R. Lorch and A. J. Smith, "Improving dynamic voltage scaling algorithms with pace," in *Proc. ACM SIGMETRICS'01*, (Cambridge, MA, USA), pp. 50–62, Jun. 10–20 2001.

[9] J. A. Stankovic and K. Ramamritham, "The spring kernel: A new paradigm for real-time systems," *IEEE Software*, vol. 8, pp. 62–72, May 1991.

[10] M. Shiraz, S. Abolfazli, Z. Sanaei, and A. Gani, "A study on virtual machine deployment for application outsourcing in mobile cloud computing," *The Journal of Supercomputing*, vol. 63, pp. 946–964, Mar. 2013.

[11] J. M. Rabaey, "Digital integrated circuits," *Prentice Hall*, 1996.

[12] T. Burd and R. Broderson, "Processor design for portable systems," *Journal of VLSI Singapore Process*, vol. 13, pp. 203–222, Aug. 1996.

[13] X. Wang, G. Giannakis, and A. Marques, "A unified apporach to QoS-guaranteed scheduling for channel-adaptive wireless networks," *Proc. of the IEEE*, vol. 95, pp. 2410–2431, Dec. 2007.

[14] A. M. Geoffrion, "Generalized bender's decomposition," *Journal of Optimal Theory and Application*, vol. 10, pp. 237–260, 1972.

[15] H. W. Kuhn and A. W. Tucher, "Nonlinear programming," in *Proc. the Second Berkeley Symposium on Math. Stat. and Prob., J. Neyman, ed.*,, (University of California), 1951.

[16] C. Audet, P. Hansen, B. Jaumard, and G. Savard, "A branch and cut algorithm for nonconvex quadratically constrained quadratic programming," *Math. Program., Springer*, vol. 87, no. 1, pp. 131–152, 2000.

[17] "IBM ILOG CPLEX Optimizer," *http://www-01.ibm.com/software/integration/optimization/cplex-optimizer*.

[18] W. Zhang, Y. Wen, K. Guan, D. Lilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, pp. 4569–4581, Sep. 2013.