Digital Communications and Networks 8 (2022) 1048–1058

Contents lists available at ScienceDirect

Digital Communications and Networks

journal homepage: www.keaipublishing.com/dcan

Energy-efficient task offloading strategy in mobile edge computing for resource-intensive mobile applications

Michael Pendo John Mahenge [a,b,*], Chunlin Li [b], Camilius A. Sanga [a]

[a] Department of Informatics and Information Technology, Sokoine University of Agriculture, Morogoro, 3038, Tanzania
[b] School of Computer Science and Technology, Wuhan University of Technology, Wuhan, Hubei, 430063, China

ARTICLE INFO

Keywords:
Mobile edge computing
Quality of experience
Task offloading
Communication networks
Particle swarm optimization

ABSTRACT

Mobile Edge Computing (MEC) has been considered a promising solution that can address capacity and performance challenges in legacy systems such as Mobile Cloud Computing (MCC). In particular, such challenges include intolerable delay, congestion in the core network, insufficient Quality of Experience (QoE), high cost of resource utility, such as energy and bandwidth. The aforementioned challenges originate from limited resources in mobile devices, the multi-hop connection between end-users and the cloud, high pressure from computation-intensive and delay-critical applications. Considering the limited resource setting at the MEC, improving the efficiency of task offloading in terms of both energy and delay in MEC applications is an important and urgent problem to be solved. In this paper, the key objective is to propose a task offloading scheme that minimizes the overall energy consumption along with satisfying capacity and delay requirements. Thus, we propose a MEC-assisted energy-efficient task offloading scheme that leverages the cooperative MEC framework. To achieve energy efficiency, we propose a novel hybrid approach established based on Particle Swarm Optimization (PSO) and Grey Wolf Optimizer (GWO) to solve the optimization problem. The proposed approach considers efficient resource allocation such as sub-carriers, power, and bandwidth for offloading to guarantee minimum energy consumption. The simulation results demonstrate that the proposed strategy is computational-efficient compared to benchmark methods. Moreover, it improves energy utilization, energy gain, response delay, and offloading utility.

1. Introduction

The rapid penetration of smart Mobile Devices (MDs) and Internet of Things (IoT) technology has brought opportunities for the development of innovative mobile applications such as virtual reality, face recognition, social networks and games [1,2]. The IoT technology is considered to be a significant innovation that implements the concept of a connected world to extend the idea of anywhere and anytime [3]. Recently, IoT has been applied in various domain such as healthcare, smart cities, autonomous driving, smart home, and smart agriculture [4]. On the other hand, mobile applications require High-Performance Computing (HPC) environments with low latency, energy consumption and cost. Over the years, cloud computing has been considered significant technological advancement for the provision of HPC and cost-efficiency services to end-users.

Accordingly, Mobile Cloud Computing (MCC) has been considered to be the key technology to facilitate offloading of computation tasks from MDs into the cloud to improve Quality of Experience (QoE) and resource utilization [5,6]. The MCC architecture shown in Fig. 1(a) is a 2-tier hierarchy comprised of two major components, including the mobile devices in the frontend and the cloud in the backend. To facilitate communication between the frontend and the backend, MCC utilizes technological infrastructures such as wireless communication, location-based tools, and mobility services to provide plentiful resources in the cloud [7]. The main benefits of MCC include augmenting mobile devices by delivering productive processing capacity and storage in the cloud. It improves performance in resource-intensive mobile applications by outsourcing execution into the cloud, and it supports multi-platforms to execute the workload of various applications in the cloud.

However, the number of connected MDs is growing rapidly and predicted to grow more in future, producing more data in the fifth-generation (5G) network than in the fourth-generation (4G) network [8]. Similarly, resource-intensive and delay-critical applications rapidly emanate great challenges in the MCC system in terms of both capacity

* Corresponding author. Centre for Information and Communication Technology, Sokoine University of Agriculture, Morogoro, 3218, Tanzania.
E-mail address: mahenge@sua.ac.tz (M.P.J. Mahenge).

https://doi.org/10.1016/j.dcan.2022.04.001
Received 11 March 2020; Received in revised form 29 March 2022; Accepted 3 April 2022
Available online 7 April 2022

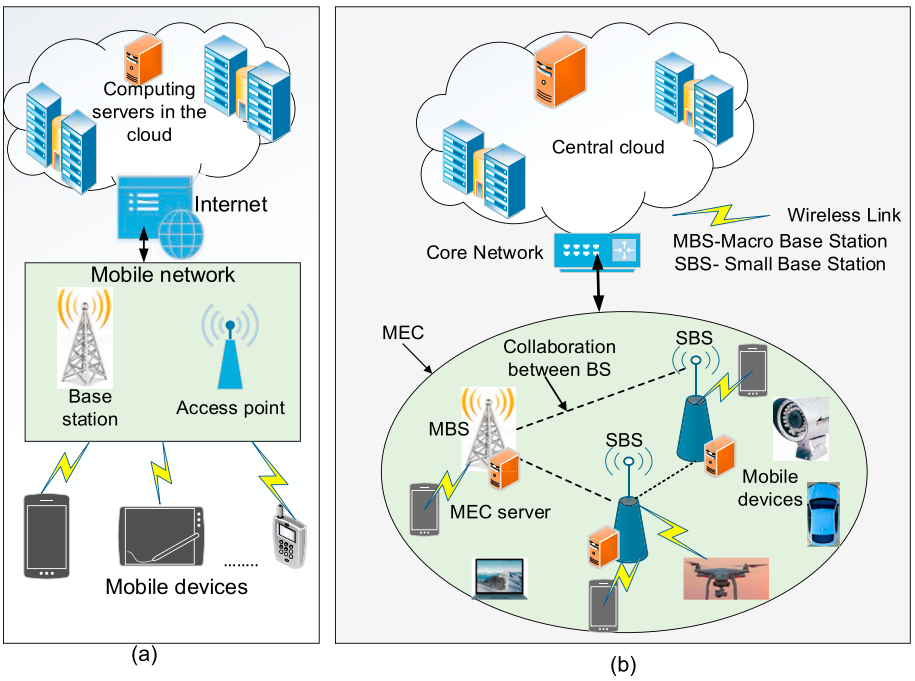

Fig. 1. The overview of mobile cloud computing architecture (a) and mobile edge computing architecture (b).

and performance [9]. In particular, such challenges include intolerable delay, congestion in the core network, insufficient QoE for end-users, high cost of resource utility such as energy and bandwidth due to multi-hop connection between end-users and the cloud [10,11]. Moreover, MDs face performance challenges emanating from limited power sources, processing, and storage capacities that hinder achieving the required QoE, especially for resource-intensive mobile applications.

To address the above-mentioned limitations, previous studies proposed architectures that bring computing resources at the proximity of end-users. In Bonomi et al. [12], the authors proposed a fog architecture that facilitates execution of delay-sensitive tasks at the edge and routes resource-intensive tasks to the cloud through a core network to achieve low latency in IoT and big data applications. In the same vein, Fan and Ansari [13] proposed workload-balancing strategy to enhance workload assignment among Base Stations (BSs) and task allocation among fog nodes. Also, in Patel et al. [14], the authors proposed Mobile Edge Computing (MEC) that brings processing capability closer to the end-users at the Access Points (AP) or BS of cellular networks. Essential features of MEC include energy-efficiency, proximity, low response delay, mobility, and location tracking support [14]. These properties are in line with the required IoT application demands of high performance, low delay, cost-effectiveness, and efficient resource utilization.

MEC architecture is a 3-tier architecture comprising of mobile devices at the frontend, MEC servers at the intermediate, and the central cloud, as shown in Fig. 1(b). MEC servers with necessary processing and storage resources are deployed at BS or AP closer to end-users. Each mobile device is associated with one closest BS or AP, which is considered as home BS depending on signal strength. MEC plays a great role, such as task offloading, caching, processing, and service delivery, in alleviating congestion in the core network, which is potential for minimizing bandwidth cost, energy usage, and latency [15,16].

Compared to MCC, MEC is a promising solution to guarantee improved performance, QoE for users, cost-efficient service delivery, and efficient resource utilization especially for resource-intensive and delay-critical applications because of its capability to deliver services at a shorter distance than cloud computing. Previous studies investigated offloading problem to reduce delay [17] or energy utilization [18,19] with an assumption that resources at the MEC are plentiful, while in reality MEC resources are inadequate, especially in terms of power, bandwidth and processing capability compared to cloud computing. Therefore, to optimize the QoE especially for computation-intensive applications, energy-efficient task offloading strategy that considers computing capacity constraint and latency requirements is worthy to be studied for the following reasons. First, the burden to process workloads generated by IoT devices is large. The pressure from the big data generated by IoT devices emanates challenges in traditional computing architectures due to limited resources such as bandwidth, power and processing resources. Second, high congestion in the core network, and third, insufficient service quality emanating from low bandwidth and limited processing capacity. Therefore, collaborative strategy between MEC servers or between MEC and the cloud is a promising solution towards increasing task processing capacity that can guarantee cost-efficient service delivery along with ensuring QoE for end users.

In this paper, the task offloading problem is studied in collaborative MEC with the objective of minimizing overall energy consumption along with satisfying capacity constraint and latency requirements. This, in returns, improves QoE for end-users. Two real-world examples of application scenarios that can benefit from the proposed approach by offloading resource-intensive tasks to more resourceful MEC are given. First is medical imaging [20] where the execution task can be offloaded to the closest MEC server for processing instead of the central cloud which is located at a longer distance from end users. The MEC servers can respond to the end users after completion, or forward request to the central cloud. The second is face recognition application [21] whereby image acquisition can be processed at the mobile device and the other tasks, such as face detection, pre-processing, feature extraction, and classification, can be offloaded to more resourceful MEC servers or central cloud for processing. The two examples demonstrate the advantages of cooperative MEC which are closer to end users rather than the central cloud. Thus, the proposed approach is potential for increasing processing capacity, minimizing response delay, and resource consumption such as energy and bandwidth which in returns improves QoE for end users. Therefore, by considering limited resource settings and latency requirement, an optimization problem for task offloading is formulated. Furthermore,

Energy-Efficient Task Offloading Strategy (ETOS) based on swarm intelligent optimization techniques is proposed. The key contributions of this paper are summarized as follows:

- We propose a MEC-assisted model to design an energy-efficient task offloading scheme that considers capacity constraints (both computing and transmission), proximity constraints, and latency requirements. Furthermore, to ensure minimum latency and increased computing power, this study leverages MEC collaborative capability that utilizes collaborative benefits between MEC servers or MEC servers and the central cloud.
- In order to achieve energy efficiency, we propose a novel hybrid approach established on the basis of Particle Swarm Optimization (PSO) and Grey Wolf Optimizer (GWO) to solve the optimization problem. The proposed approach considers efficient resource allocation such as sub-carriers, power, and bandwidth for offloading to guarantee minimum energy consumption along with satisfying delay requirements
- We perform extensive simulations to assess the performance of the proposed strategy using various metrics. The simulation results from various parameters show that the proposed ETOS outperforms the baseline approaches in terms of energy utilization, energy gain, response delay, and offloading utility under a limited resource setting.

The remainder of the paper is organized as follows: the latest progress of existing works related to this study is presented in Section 2, the proposed approach mathematical models are presented in Section 3, and the proposed algorithm is presented in Section 4. Section 5 describes the simulation results, and lastly, Section 6 gives conclusion and proposes future works.

2. Related works

Numerous studies have been done mostly in developed countries and least in developing countries to examine methodologies for cost-efficient computing and service delivery in MCC architectures. In Satyanarayanan et al. [21], the cloudlet, a small-scale data center with computing and storage capability placed near mobile users and accessed through WiFi connection is proposed. However, limited WiFi coverage is one of the major limitations reported in the literature. In Chiang [22], a fog architecture is presented where edges are part of the core networks and data centers. Nevertheless, the known challenge in the literature is difficult to guarantee QoE for mobile users as computations are not incorporated in the mobile network architecture. In Khan et al. [23], the authors investigated the MCC architecture, incorporating cloud computing into the mobile environment to augment the performance. Challenges facing MCC architecture reported in the literature include response delay, high bandwidth utilization, and high cost of data transportation to the cloud for processing. Furthermore, a traditional centralized cloud computing faces challenges of high network congestion, network overload and depressed robustness [24]. On the other hand, the distributed cloud faces challenges of complexity and resource management complications [25]. In order to address such challenges, the edge cloud architecture [26] and the MEC architecture [14,27] are proposed to meet the requirements of IoTs applications, such as low delay, cost-effectiveness, and efficient resource consumption.

The energy and latency metrics have been considered important in evaluating the Quality of Experience (QoE) in MEC systems. A dynamic offloading and resource assignment model based on a stochastic optimization strategy to reduce energy usage is proposed in Kwak et al. [28]. In Mao et al. [29], the authors proposed a dynamic strategy for computation offloading in a single-user setup with energy harvest equipment to solve the challenge of energy in mobile devices. The proposed approach is based on the Lyapunov optimization policy and focuses on offloading choice to effectively utilize the available resources. In Sardellitti et al. [30], the authors studied the offloading problem in multi-user MEC networks and proposed a resource allocation scheme based on a joint optimization strategy to achieve low energy consumption and latency. In Ref. [31], the authors studied the power-latency problem in multi-user MEC network that aimed to reduce power consumption. They proposed an online offloading decision algorithm governed by the Lyapunov optimization strategy. In Ren et al. [32], the authors investigated the latency optimization problem in multi-user MEC systems. They proposed a combined approach for communication and resource provisioning strategy to minimize the total energy usage and latency in resource-constrained mobile devices. In Mahenge et al. [33], the authors studied task unloading problems in collaborative mobile edge and cloud computing focusing on improving QoE in resource-poverty devices. In Cheng et al. [34], the authors proposed the combined approach for offloading and resource provisioning to reduce overall energy usage in delay-critical applications based on multi-MEC networks. In Wang et al. [35], the authors studied task offloading in MEC, focusing on the joint optimization problem of delay and energy. They modeled the problem as MINLP, which was solved by the relaxation method. In Cao et al. [36], the authors proposed a joint optimization of computation and communication resource provisioning in MEC to reduce energy consumption while satisfying latency constraints. They found that the cooperative approach considerably enhances the performance and is energy efficiency compared to other approaches without cooperative design. The energy-efficient approaches in MEC have been proposed by a number of prior works. In Liu et al. [37], the authors proposed an optimal task scheduling scheme in single-user MEC to minimize latency under power constraints. In You et al. [38], the authors studied an energy minimization problem constrained by delay in multi-user MEC networks. They proposed an efficient resource provisioning scheme that significantly reduced the overall energy utilization. Also, a novel approach that incorporates the Wireless Power Transfer (WPT) into MEC to address the challenges of limited energy in mobile computing is proposed in [39,40].

In conclusion, this work differs from existing studies: first, most existing research addressed the task offloading problem in MEC without taking into account characteristics of MEC such as limitation of MEC resources. The proposed strategy considers capacity constraints (both computing and transmission), and latency requirements. Then, efficiently allocates MEC resources that meet minimum energy consumption and deadline constraint. Second, most existing works focused on optimising task offloading decisions autonomously among single user/multi-user MEC and the central cloud or among several MEC servers. Instead, this study considers a cooperative approach among multiple MEC resources or between MEC servers and the central cloud to enhance capacity. Third, the proposed strategy based on intelligent swarm optimization techniques only requires minimum information from MEC system, which suits significantly in a decentralized approach.

3. System models

In the network model, an application scenario comprising of MDs with N tasks is considered.

Also, it is considered that MEC servers with necessary processing and storage resources are deployed in BS or AP. Correspondingly, it is assumed that each MD is associated with one closest BS or AP, which is considered as home BS depending on signal strength. Furthermore, the home BS play two potential roles. First, it is responsible for task execution if it has the resources required to satisfy the task requirements. Second, it acts as a relay node to forward the task to the neighbouring BS or remote servers. The end users connect to MEC servers through a wireless network while the MEC server connects to neighbouring servers and remote server via a backhaul link. Let $\varphi = \{1, 2, \ldots . Q\}$ be the set of MDs, and $\mathscr{J} = \{1, 2, \ldots N\}$ denotes the set of tasks, respectively. Each task can be characterized by three parameters denoted as $J(s_k, c_k, D_k)$ where $s_k(inbits)$ denotes the input-data size of the task, $c_k(\text{MIPS})$ denotes the computational intensity (workload), and $D_k(inseconds)$ represents the

maximum endurable delay. The estimation of these parameters can be achieved by task profilers based on the nature of applications [41]. Also, let $\Phi = \{1, 2, \ldots, s, \ldots S\}$ be the set of MEC servers in the collaborative domain with the total computation capacity $\mathscr{K}$ and bandwidth capacity $\mathscr{B}$ respectively. Moreover, let α_s denote the portion of resource allocated to an individual task k at each MEC server s ∈ S. Table 1 presents the list of notations and Fig. 2 shows the overview of task offloading system model.

3.1. Communication model

In this case, it is considered that users send requests to the BS/AP currently associated (home BS), afterwards, the BS check if the requested service can be delivered by the home BS. If the requested service cannot be satisfied by the home BS, then the BS forwards the request to the nearby BS/AP. Referring to this scenario, let P_η denotes the power incurred by users' equipment during communication with the BS, and ψ_η denotes the channel gain related to user η and the BS. Then, when a request is sent to the BS/AP, data rate of users' device η can be calculated by

$$R_\eta = Blog_2\left(1 + \frac{P_\eta \psi_\eta}{\sum_{k \in \mathscr{J}} x_k P_k \psi_k + \sigma^2}\right) \quad (1)$$

where $x_k \in \{0, 1\}$ is the control variable which is 1 if the task k is executed in the mobile device, and 0 if it is offloaded to MEC layer, the term $\sum_{k \in \mathscr{J}} x_k P_k \psi_k$ computes the interference at home BS which is caused by other MDs uploading tasks to the MEC servers, σ^2 is the noise variance, and B is the network bandwidth.

For each request arriving at the BS/AP, MEC servers can cooperate with one another to deliver the requested service. For example, if a user requests a video, MEC servers can transcode the video to the suitable bitrate form and then deliver it to the requesting user. Transcoding a video requires intensive-resources for processing, thus requiring efficient resource assignment strategy to minimize cost that would be incurred for task transmission and processing. The control server at the BS is responsible for scheduling and assigning tasks to the MEC servers related to minimum request execution cost. Let ψ_s be the channel gain of server s at BS i when communicating with the server h at BS j, and P_s be the transmission power of server s. Therefore, the transmission rate between server s and server h can be found by

$$R_s = Blog_2\left(1 + \frac{P_s \psi_s}{\sum_{k \in \mathscr{J}} y_k P_k \psi_k + \sigma^2}\right) \quad (2)$$

where $y_k \in \{0, 1\}$ is the control variable which is 0 if the task k can be satisfied by the home BS and 1 if the task is forwarded to the neighbouring BS in the collaborative range, $\sum_{k \in \mathscr{J}} y_k P_k \psi_k$ is the interference caused by inter-cell communication when MEC servers in different locations communicate.

3.2. Service utility cost models

In this study, we consider binary offloading, referred to as the process whereby the entire task can be computed either locally or unloaded to more resourceful MEC servers without being partitioned. For local computation, let f_{dev} be the processing capacity (CPU frequency) of the MD. Then, without offloading policy, the delay cost can be obtained by $D_L = \frac{C_k}{f_{dev}}[seconds]$, and the energy utilization cost can be obtained by $E_L = \mathscr{E}_d (f_{dev})^2 C_k$ [42], where $\mathscr{E}_d$ is the energy consumed per CPU cycle.

When the task is offloaded to the MEC layer, the home BS can process the task only if it has the required resources to complete processing. Otherwise, the task is forwarded to other nearest MEC servers in the collaborative domain or to the remote servers based on capacity and proximity constraint. Therefore, the offloaded task incurs communication as well as computation costs. The offloading parameters are defined as $x_k \in \{0, 1\}$, where $x_k = 1$ is an indicator that task k is processed locally in MD; otherwise, $x_k = 0$ denotes that task k is offloaded to the MEC layer. Also, let $y_k \in \{0, 1\}$, with $y_k = 0$ indicating that the offloaded task k is processed by home BS; otherwise, $y_k = 1$ denotes that task k is forwarded to the nearest BS in the collaborative domain or remote servers based on resources and time constraint. Therefore, the delay and cost models for an offloaded tasks are formulated as follows.

Delay-cost model: In this case, the end-to-end delay is considered. For example, consider an application scenario where a medical image is captured by IoTs devices that need to be offloaded to more resourceful MEC servers for processing. In this case, the system incurs the image uploading delay, overhead delay for queueing and task scheduling, the execution delay, and delay incurred for a server to respond to the MD. Without the loss of generality, the response delay is ignored in this study. Therefore, let $D_{d,s}$ denote the delay incurred during task transmission from MD to MEC, which can be obtained as $D_{d,s} = \frac{s_k}{R_\eta}$. Also, let D_{qo} represent the overhead delay which can be obtained by $D_{qo} = s_k/\omega$, where ω is the average rate for sub-carriers assignment. Also, denote $D_{s,j}$ as the communication delay incurred for the task forwarded from home BS s to the nearby BS j in the collaborative domain through X_2 link [43] or to the remote server through a backaul link. Therefore, $D_{s,j}$ can be obtained by $D_{s,j} = \sum_{k=1}^{N} \frac{y_k s_k}{R_s}$ where R_s is the link capacity connecting adjacent MEC servers. Moreover, if f_{ser} denotes the CPU execution rate of task k allocated by MEC server, the execution delay denoted as D_p can be computed as $D_p = \sum_{k=1}^{N} \frac{x_k c_k}{f_{ser}}$. Therefore, the overall delay of the offloaded tasks can be obtained by

$$D_{off} = D_{d,s} + D_{qo} + D_{s,j} + D_p \quad (3)$$

Energy-cost model: Let $\mathscr{F} = \{1, 2, \ldots, \omega, \ldots, F\}$ represent the set of manageable sub-carriers with sub-carrier average bandwidth B_ω for transferring task k to the MEC. The parameter ω_{ks}^{ω}, where$\omega \in \mathscr{F}, k \in \mathscr{J}$, $s \in \Phi$, denotes the offloading indicator such that $\omega_{ks}^{\omega} = 1$ means task k is offloaded to MEC server s by sub-carrier ω, otherwise, $\omega_{ks}^{\omega} = 0$. Also, the

Table 1
List of important notations.

List of notations			
P_η, P_s	Power consumption of device η , server s when communicating with BS or other servers	D_L	Delay for local execution
ψ_η, ψ_s	Channel gain of device η , server s	E_L	Energy consumption for local execution
R_η, R_s	Data transmission rate of device η , server s	s_k	Input-data size of the task
B	Network bandwidth	D_{off}	Total delay for remote task execution
σ^2	Noise variance	E_{off}	Total energy consumption for remote task execution
x_k, y_k	Offloading control variables	$P_{d,s}$	Uplink transmission power
C_k	Computational intensity (workload)	$pBest_z^l$	Best location vector of each device
$\mathscr{E}_d$, $\mathscr{E}_{ser}$	The energy coefficient of device, server	$gBest$	Global best position vector
$B_{k\omega}$	Sub-carrier bandwidth allocated to task k	f_{dev}	CPU frequency of the mobile device
D_k	Task execution deadline	f_{ser}	CPU frequency of MEC server

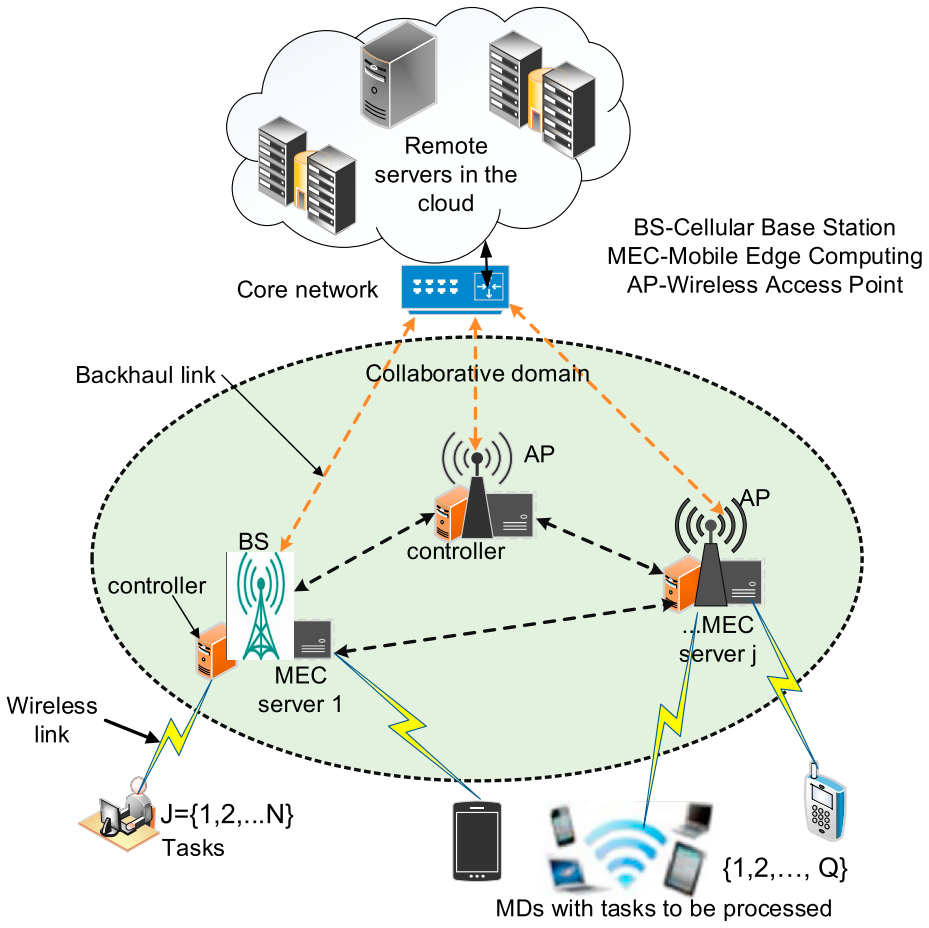

Fig. 2. Overview of task offloading system model.

power allocation matrix can be represented by $\mathscr{R} = \{\mathscr{P}_{\mathrm{ks}}^{\omega} | \mathscr{P}_{\mathrm{ks}}^{\omega} \in [0, \mathscr{P}_{\mathrm{budget}}], \omega \in \mathscr{T}, \mathrm{k} \in \mathscr{J}, \mathrm{s} \in \Phi\}$, where $\mathscr{P}_{\mathrm{ks}}^{\omega}$ indicates the power allocated to the task k uploaded to MEC server. Therefore, the task uploading strategy must satisfy the constraint expressed as

$$\sum_{s \in \Phi} \sum_{\omega \in \mathscr{T}} \Omega_{\mathrm{ks}}^{\omega} \leq 1, \mathrm{k} \in \mathscr{J} \tag{4}$$

Also, let $P_{d,s}$ [Watts] be the power for transferring a job k from MD to MEC, and $P_{s,j}$ [Watts] be power required for communication overhead between server s and server j. The energy consumption for uplink transmission ($E_{\mathrm{d,s}}$) can be found by $E_{\mathrm{d,s}} = P_{d,s}\mathrm{D}_{\mathrm{d,s}} = P_{d,s}s_k/R_{\eta}$. Likewise, the energy consumption incurred for communication overhead between MEC servers can be obtained by $E_{s,j} = P_{s,j}\frac{s_k}{R_s}$. If the task is not forwarded to other nearby server, then $P_{s,j} = 0$. Moreover, the execution energy cost of the tasks offloaded to a particular server is denoted as E_e which can be obtained by $E_e = \mathscr{E}_{ser}(f_{ser})^2 c_k$, where $\mathscr{E}_{ser}$ is the energy cost per one CPU cycle of the MEC server. Therefore, the overall energy cost (E_{off}) incurred for the offloaded task can be obtained by

$$E_{\mathrm{off}} = E_{\mathrm{d,s}} + E_{s,j} + E_e \tag{5}$$

3.3. Problem formulation

In this study, the main objective is to find out the energy-efficient design that minimizes the system's overall energy cost and satisfies the capacity and delay constraints. Therefore, the task offloading problem is transformed into an optimization model with an objective function expressed as

$$\underset{x_k, y_k}{minimize} \sum_{k=1}^{N} \left(\sum_{i=1}^{Q} x_k E_L + \sum_{s=1}^{S} y_k E_{off} \right) \tag{6}$$

Subject to

$$\sum_{k=1}^{N} x_k = 1, \forall_{k \in \mathscr{J}} \tag{7}$$

$$\sum_{k=1}^{N} y_k = 1, \forall_{k \in \mathscr{J}} \tag{8}$$

$$\sum_{k=1}^{N} \sum_{s=1}^{S} x_k \alpha_s \leq \mathscr{K} \tag{9}$$

$$\sum_{\omega=1}^{F} B_{k\omega} \leq \mathscr{B}, \forall_{k \in \mathscr{J}} \tag{10}$$

$$D_{off} \leq D_k \tag{11}$$

The expression (6) above presents the optimization objective that minimizes the total energy usage cost. Constraint (7) guarantees the feasibility of the task offloading strategy by restricting each MD to offload task k to at most one MEC server, and (8) guarantees that the home BS can choose only one nearby MEC server in the collaborative domain to process task k. Constraints (9-10) ensures that the allocated computing resources and bandwidth cannot exceed the available capacities, and (11) ensures that task processing delay meets its maximum tolerable delay.

Moreover, to demonstrate that problem (6) with constraints (7)–(11) is NP-complete, we consider that the Set Cover Problem (*SCP*) may be defined as: given a universal set $\mathbb{P} = \{p_1, p_2, \ldots p_N\}$ and a collective subset of $\mathbb{P}$ represented by $\mathbb{H} = \{h_1, h_2, \ldots h_L\}$ where every subset h_k incurs energy cost ξ_k for processing task k. The aim is to find out a collection of members of $\mathbb{H}$ whose union equals to $\mathbb{P}$, i.e. $, (\cup h_i = \mathbb{P},\ i = 1, 2, \ldots)$ and the overall energy consumption is minimized.

To prove NP-hardness, the *SCP* is reduced to an instance of the problem formulated in this study as follows: First, each member p_i in the *SCP* is mapped to a MD requesting services from the BS. Second, the energy consumption for processing user requests at the i^{th} BS is set to ξ_k. Third, the energy cost $E_{off} = 0$ if $p_i \in h_k$; otherwise, it is set to $2\xi_k$, and fourth, the energy cost $E_{s,j} = 0$ if the task is processed at the home BS. All members of *SCP* are covered if and only if the overall energy cost (E_{off}) of the solution is 0. Moreover, if $\exists_x$ such that x is the solution in *SCP* with energy cost $\mathscr{E}$, then the sets form the MEC servers and the overall cost is $\mathscr{E}$. Moreover, it is assumed that there is a solution to the optimization problem (6) with energy cost $\mathscr{E} = \mathscr{E}^{local} + \mathscr{E}^{remote}$, where $\mathscr{E}^{local}$ is the entire energy cost for executing tasks locally in MDs and $\mathscr{E}^{remote}$ is the overall energy cost for offloaded tasks.

Then, the first case occurs when the total energy cost (E_{off}) is 0. Subsequently, the chosen MEC servers produce the collective sets of the *SCP* with energy cost $\mathscr{E}$. Conversely, the second case occurs when $E_{\text{off}} > 0$, which implies that there exist some members in the *SCP* that are not covered. Then, for each BS i whose $E_{\text{off}} \neq 0$, the new MEC server (i.e., the new subset h_k) is selected to process task k where $E_{s,j} = 0$. Subsequently, $E_{s,j} + (E_{\text{d,s}} + E_e) = 0 + \xi_k \leq 2\xi_k$, thus the new overall energy cost $\mathscr{E}^* \leq \mathscr{E}$. Therefore, the *SCP*, which is NP-hard, is a polynomial-time that can be reduced to the formulated optimization problem.

4. The proposed energy-efficient task offloading strategy

The optimization problem (6) is a Non-Linear Problem (NLP) that is NP-hard; in practice, finding the best solution to the problem (6) can be challenging and time-consuming. Therefore, it is required to establish an efficient method to find the best solution. Recently, numerous approaches have existed to solve NP-hard problems, such as Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Ant Colony Optimization (ACO), and Grey Wolf Optimization (GWO). Specifically, the PSO inspired by search behaviours and hunting is widely used in hybrid algorithms based on a swarm intelligent technique because of its high global optimization efficiency, fast convergence speed and easy implementation. On the other hand, GWO has been adopted in solving various real-world problems, such as resource allocation problems, scheduling problems, forecasting, and many others as presented in Refs. [44–47].

Therefore, in this study, Energy-efficient Task Offloading Strategy (ETOS) is proposed to solve the optimization problem. The proposed approach is a hybrid method based on a swarm intelligent technique established through PSO and GWO. The motive behind using a hybrid approach is to improve the search capacity within the solution space and enhance the convergence performance towards a better solution. It is known from the literature review that PSO is characterized by high global optimization efficiency, fast convergence speed and easy implementation. Also, the resilient strength of GWO dwells in the capability to explore the search space achieved through imitation of the leadership-hierarchy of grey-wolves. Therefore, in this study, we leverage the combined strengths of the two methods to improve the quality of results, speed of iterative convergence, and capability to search global optimum.

The PSO algorithm [48] is an intelligent and stochastic swarm algorithm. Its motivation is to learn search and hunting from animal behaviour to determine the global optimization problem in which each member of the group is called particle. In this study, the researchers considered a mobile-application scenario where N tasks with resource-intensive and low-latency requirements must be unloaded into MEC servers for processing. Each mobile device is considered as a particle. Consequently, the proposed problem can be represented as $N-$dimensional vectors. Thus, the placement vector can be represented by $\lambda_z(t) = (\lambda_{z1}, \lambda_{z2}, \ldots \lambda_{zm}, \ldots, \lambda_{z\eta})$ where λ_{zm} denotes the m^{th} position of the z^{th} particle (device) representing the probable solution vector. Also, the velocity vector can be represented as $V_z(t) = (v_{z1}, v_{z2}, \ldots v_{zm}, \ldots, v_{z\eta})$ with v_{zm} indicating the velocity of z^{th} particle in m^{th} dimension. Likewise, the best location vector for each device is indicated by $\text{pBest}_z^l = (pB_{z1}, pB_{z2}, \ldots, pB_{z\eta})$ and the best position vector of the whole population is represented as $gBest = (gB_1, gB_2, \ldots, pB_\eta)$ respectively. In PSO method, the location of each member of the crowd in the search space should be updated using the following mathematical models

$$V_z(t+1) = V_z(t) + \theta_1\eta_1(pB_{zi} - \lambda_z(t)) + \theta_2\eta_2(g_{best} - \lambda_z(t)) \tag{12}$$

$$\lambda_z(t+1) = \lambda_z(t) + V_z(t+1) \tag{13}$$

The GWO algorithm [49] is a swarm intelligent method motivated by the hunting and leadership hierarchy whereby the crowd can be divided into four dissimilar groups known as alpha (α), beta (β), delta (δ), and omega used for mimicking the leadership hierarchical structure. The α is the highest level in the structure which is responsible for overall decision making. The β is the second high level in the structure known as assistant wolves that assist α in making decisions and guide the lower levels in the structure. The δ is the third ranked grey wolf which assists the higher levels and it plays a great role in observing the boundaries of their region, guarding the group, and hunting. The omega is the lowermost rank which must obey the members of higher levels. According to Mirjalili et al. [49], the algorithm implements three key hunting: searching, encircling, and attacking.

Let $\lambda(t)$ and $\lambda_p(t)$ denote the location of the wolf and prey at time t, the mathematical models for encircling procedure can be expressed as

$$d = \left|\theta.\lambda_p(t) - \lambda(t)\right| \tag{14}$$

$$\lambda(t+1) = \lambda_p(t) - v.d \tag{15}$$

where θ and v are vectors.

In the hunting process, it is assumed that α, β and δ have a better understanding of the optimal solution $\lambda_p(t)$, which is the location of the prey. Also the location of wolves can be updated with respect to the location of α, β and δ. Therefore, the hunting behavior can be modeled mathematically as

$$\begin{aligned} d_\alpha &= |\theta_1.\lambda_\alpha(t) - \lambda(t)|, \\ d_\beta &= \left|\theta_2.\lambda_\beta(t) - \lambda(t)\right|, \\ d_\delta &= |\theta_3.\lambda_\delta(t) - \lambda(t)| \end{aligned} \tag{16}$$

$$\begin{aligned} \lambda_1 &= \lambda_\alpha(t) - v_1.(d_\alpha), \\ \lambda_2 &= \lambda_\beta(t) - v_2.(d_\beta), \\ \lambda_3 &= \lambda_\delta(t) - v_3.(d_\delta) \end{aligned} \tag{17}$$

$$\lambda(t+1) = \frac{\lambda_1 + \lambda_2 + \lambda_3}{3} \tag{18}$$

where $\theta_j = 2\eta_2$ and $\eta_2 \in [0, 1]$, which is generated randomly for $j = 1,\ 2,$ and 3. Similarly, $v_j = 2h.\eta_1 - h$ where $\eta_1 \in [0, 1]$, which is generated randomly. According to Ref. [49], h progressively declines from 2 to 0. Let V be the value generated randomly in the range $[-2v, 2v]$, if $|V| < 1$, the members of the crowd are forced to approach the prey, which is the optimal point. When $|V| > 1$, the members of the crowd are forced to deviate from the prey.

In the proposed ETOS, searching and attacking processes are controlled by inertial weight I_ω. Therefore d_α d_β, and d_δ can be updated using the newly adapted expression as follows [50]:

$$d_\alpha = |\theta_1.\lambda_\alpha(t) - I_\omega * \lambda(t)|,$$

$$d_\beta = \left|\theta_2.\lambda_\beta(t) - I_\omega * \lambda(t)\right|, \tag{19}$$

$$d_\delta = \left|\theta_3.\lambda_\delta(t) - I_\omega * \lambda(t)\right|$$

Similarly, the velocity and location of each particle in the global search domain can be updated according to the following newly formulated equations

$$V_z(t+1) = I_\omega * \begin{pmatrix} V_z(t) + \theta_1\eta_1(\lambda_1 - \lambda_z(t)) \\ +\theta_2\eta_2(\lambda_2 - \lambda_z(t)) \\ +\theta_3\eta_3(\lambda_3 - \lambda_z(t)) \end{pmatrix} \tag{20}$$

$$\lambda_z(t+1) = \lambda_z(t) + V_z(t+1) \tag{21}$$

where $z = 1, 2, 3, \ldots, I_\omega$ is the inertia weight, $V_z(t)$ is the speed of a particle z at a time t, $\lambda_z(t)$ is the position of a particle$z$ at a time t, θ_i is the uniformly distributed random numbers, and η_i is the accelerating factor.

Finally, in order to evaluate whether the solution is improved, the fitness function method is used whereby the fitness equation can be expressed as

$$F(x,y) = \sum_{k=1}^{N}\left(\sum_{i=1}^{Q} x_k E_L + \sum_{s=1}^{S} y_k E_{off}\right) + P(x,y) \tag{22}$$

where $P(x,y)$ is the penalty function. In order to establish the penalty function of the fitness equation (22), the method proposed by Dai et al. [17] is applied to adjust the penalty co-efficient function to solve the capacity-associated constraints. Therefore, the penalty function can be expressed as

$$P(x,y) = \xi(x,y) + \beta(x) \tag{23}$$

where

$$\xi(x,y) = \sum_{i=1}^{Q}\xi_1 \times \left(\sum_{k=1}^{N} x_k - 1\right)^2 + \sum_{s=1}^{N}\xi_2 \times \left(\sum_{k=1}^{N} y_k - 1\right)^2 \tag{24}$$

$$\beta(x) = \sum_{i=1}^{Q}\left(\beta_1 \times \left(\frac{1}{\sum_{k=1}^{N}\sum_{s=1}^{S} x_k\alpha_s - \mathscr{K}}\right)^2\right) + \sum_{\omega=1}^{F}\left(\beta_2 \times \left(\frac{1}{\sum_{\omega=1}^{F} B_{k\omega} - \mathscr{B}}\right)^2\right) + \sum_{i=1}^{Q}\left(\beta_3 \times \left(\frac{1}{\mathrm{D_{off}} - D_k}\right)^2\right) \tag{25}$$

where $\xi_1, \xi_2, \beta_1, \beta_2$ and β_3 represent penalty variables. The penalty variables represent the weights of the constraint expressions in the initial constrained objective function (6). Particularly, ξ_1 and ξ_2 represent weights of constraints (7) and (8), while β_1, β_2 and β_3 represents the weights of (9), (10) and (11). The purpose of the penalty variable is to guarantee the execution of the proposed algorithm in the domain of the feasible region. Therefore, the expression (22) is the new objective function obtained through combination of the original objective function (6) with the weighted constraint expressions. The motive behind the combination is to convert the constrained expression (6) to an unconstrained expression (22).

Algorithm 1is the pseudo-code of the proposed ETOS strategy that find the optimal processing plan for each task in terms of energy cost. The proposed ETOS algorithm obtains all necessary information and starts the execution process. First, the necessary parameters are set, and the algorithm determines the preliminary sub-carrier assignment during the first run of iteration (Algorithm 1 line 1–3). The initial velocity and location of each particle/device are set randomly (Algorithm 1 line 4), and the corresponding fitness value of each device is acquired from equation (22) (Algorithm 1 line 5). Moreover, the algorithm sets the vector p$Best_z^l$ which keeps track of individual particles' best location, and *gBest* vector which keeps track of the particle with the best result of all particles in the population (Algorithm 1 line 6). The offloading decision is made based on the task processing costs in terms of delay and energy preferences (Algorithm 1 line 7–12). Second, the algorithm starts iterative execution in the loop until the termination criteria are met (Algorithm 1 line 13–28). For each particle, the individual best value and the global best value are updated (Algorithm 1 line 15–22). Furthermore, the velocity ($V_z(t+1)$) and location ($\lambda_z(t+1)$) are updated using the established expressions (20) and (21) (Algorithm 1 line 23–26). Finally, the algorithm returns the optimized energy utilization results (Algorithm 1 line 31).

Algorithm 1: Energy-efficient task offloading Algorithm

Input: $C_k, s_k, f_{dev}, f_{ser}, D_k$

Output: The optimized energy consumption
1: Initialize $I_\omega, a, \eta, B_v, iter$
2: Z = mobile_devices_initialization()
3: Set initial sub-carrier assignment vector
4: Set $V_z(t)$ and $\lambda_z(t)$ randomly
5: Get fitness value of agents
6: Set p$Best_z^l$ and $gBest$ vectors
7: **for** each task $k \in \mathcal{J}$ received at the BS $i \in \Phi$, **do**
8: compute $D_L, D_{off}, E_L, E_{off}$
9: if $D_L \leq D_{off}$ and $E_L \leq E_{off}$
10: process task k locally in mobile device
11: else
12: offload tasks k to MEC server
13: set iter = 2
14: **while**$(iter < MaxNumOfIter)$ **do**
15: **for** each search agent **do**
16: fitness value = p$Best_z^l$
17: ***if*** $fitness\ value > pBest$
18: $pBest = fitness\ value$
19: **end if**
20: $B_v = pBest$
21: **end for**
22: $gBest \leftarrow Best\ value\ in\ B_v$
23: **for** each particle z in the whole population **do**
24: update velocity using (20)
25: update position using (21)
26: **end for**
27: iter = iter + 1
28: **end while**
29: **end if**
30: **end for**
31: Return B_v, $gBest$

The optimized results achieved by the proposed algorithm are potential information required by BS/AP to efficiently allocate resources, such as power and sub-carriers, for task offloading. Therefore, the BS/AP will reply to the service requester (end user), which will make the offloading choice based on the results. Specifically, the scenario is as follows: the Mobile Device (MD) owning task decides which task should be computed locally or offloaded to more resourceful MEC servers. Then, the MDs inquire required resources from the BS currently connected (home BS). Upon inquiry reception at the BS, the controller server, which maintains the Resource Allocation Table (RAT) and is responsible for task scheduling, allocates the optimal MEC server within the cooperative space. Then, the MDs offload the task to the selected optimal MEC server

to process the task.

The complexity of the proposed Algorithm 1 lies in computing the cost of a candidate solution depending on the current location of the MDs and the computations needs to update each particle's velocity based on expression (20) and location (21). Algorithm 1 optimizes energy consumption and resource allocation through two phases. Initially, the algorithm initializes the device's locations represented as a dimensional vector with size N. Therefore, the time complexity of the initialization process is $O(N)$. Then, the search agents explore the solution space to find the best solution that satisfies the optimization objective. Therefore, the complexity for computing the cost of candidate solution for each iteration is $O(N*\Theta)$ where N is the dimension of the problem, and Θ is the population size. The following section presents the discussion of the simulation results.

5. Simulation results and discussion

This section discusses the simulation results to assess the possible benefits from the proposed approach using various parameters, such as the size of input data, processing capacity, number of MEC servers, and data uploading rate. In the simulation, the coverage radius (R) of the BS is set to 500 m. Each user can inquire about services from the closest BS in an area covered by radius R. Similar to Ref. [51], users are distributed randomly following poisson distribution by poisson parameter $\mu > 0$ and the distribution variable $\delta = 1, 2, 3, \ldots$, with an average of 50 users at each BS. Throughout the simulation, the number of search agents are set to 30, number of iterations varies between 10 to 500, $\theta_1 = \theta_2 = \theta_3 = 0.5$, and $I_\omega = 0.9751$. The values of η_i are randomly generated and evenly distributed in the range 0 and 1. All over, it is considered that resources for each MEC server are evenly distributed within 8 GB–500 GB for storage and 100MIPS to 1000 MIPS for processing. The energy consumption per CPU cycle $\mathscr{E}_d$ and $\mathscr{E}_{ser}$ considered to be $1x10^{-24}$ and $1x10^{-26}$ [34]. Furthermore, the noise variance (σ^2) is set to $-$ 102dBm. Other simulation parameters are presented in Table 2 below.

The proposed approach was evaluated using various parameters, and the metrics used for assessment include the average energy utilization, total energy utilization, energy gain, response delay, and the offloading utility. From the simulation results, the performance was assessed in comparison with the following existing baseline approaches.

1) *Optimal Enumeration Offloading Strategy (OEOS)*: This method searches for the optimal solution through enumeration of all potential solutions and chooses the optimal server to offload the task [52].
2) *Random Offloading Approach based on Dynamic Programming with Hamming distance termination (ROA-DPH):* The ROA-DPH makes offloading decision randomly and freely select the server to offload the tasks based on resource availability. ROA-DHP uses a random-based approach and hamming-distance termination principle to obtain a better offloading decision [53].
3) *Adaptive Task Offloading (ATO):* The MDs dynamically offload jobs to the higher layers with more computing resources to minimize execution cost [54]
4) *Local:* All tasks executed by MDs.

Table 2
Variables, value and description of each variable used in the simulation.

Variables	Values
Number of MEC servers (S)	10
Uplink transmission power ($P_{d,s}$)	20 dBm
System Bandwidth (B)	20 MHz
MEC execution rate (f_{ser})	[2,5] GHz
MDs execution rate (f_{dev})	1GHz
Input data size (s_k)	20 $\sim$ 100MB
Workload (C_k)	$(1 \sim 2.5) \times 10^7$ (CPU cycles)
Sub-carrier average bandwidth	15 kHz

Fig. 3 illustrates the comparison of energy consumption between the remote execution (all tasks offloaded to MEC servers) and the local execution (tasks processed in MDs) under various MEC execution rates and data size. The ratio of energy consumption for local execution is 1. From the figure, it is obvious that the ratio of energy consumption will decrease when the MEC execution rate increases because when the execution rate of MEC server is large, sufficient resources can be provided to handle large workload with fewer servers, so as to reduce processing time and overall energy consumption. For example, the ratio differences (local $-$ remote) of energy consumption for data sizes 20MB, 50MB, and 80MB are 0.42, 0.26, 0.12 when the MEC execution rate is small, e.g., 2×10^9 Hz while they are 0.62, 0.419, 0.22 when the MEC execution rate is large, e.g., 7×10^9 Hz. Also, when the data size is small (for example, 20 MB), the remote execution consumes less energy compared to the medium data size (50 MB) and large data size (80 MB), resulting in a small ratio because the energy consumed for transmission and execution is less for small data size. Conversely, when the offloaded file is large, more energy is required for transmission and processing which will increase energy consumption. For example, the ratio of energy consumption (data size = 80 MB, MEC execution rate = 1×10^9 Hz) is larger than 1 because when the data size is large (80 MB), more resources are required for data transmission and processing. Therefore, when the MEC execution rate is low (1×10^9 Hz), offloading task to MEC consumes more energy than processing locally in a mobile device, which causes the ratio of energy consumption to be larger than 1. Moreover, when the number of MDs uploading tasks to servers with richer resources becomes larger, it may lead to a high probability of interference resulting in high energy consumption. Therefore, it is critical to optimize the selection of MEC servers to unload tasks according to application preferences and available resources to achieve minimum energy cost.

Fig. 4 demonstrates the complexity comparison analysis between proposed ETOS and OEOS when the number of tasks changes and the task-input size is fixed. Although OEOS guarantees optimal solution for offloading decisions, the main shortcoming of this algorithm resides in its exponential time-complexity $O(3^k)$, where k is the number of tasks. Consequently, OEOS is inappropriate in application scenarios with a bigger number of tasks requiring intensive resources for execution. Therefore, in this paper, we use OEOS as a benchmark algorithm to the proposed ETOS. The analysis of the simulation results proves that the proposed ETOS executes faster than OEOS in both cases when, the task-input size is set to 24 MB and 48 MB respectively, as shown in Fig. 4.

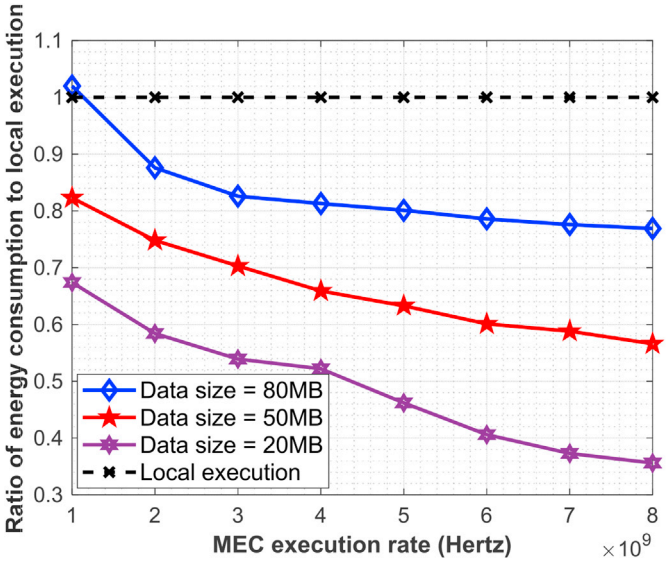

Fig. 3. Ratio of energy consumption to local execution versus MEC execution rate under various data size.

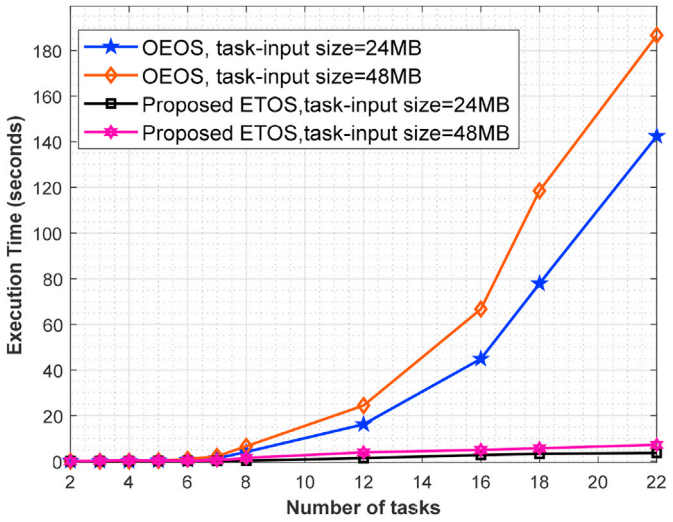

Fig. 4. Comparisons of the execution time between OEOS and proposed ETOS when the number of tasks changes.

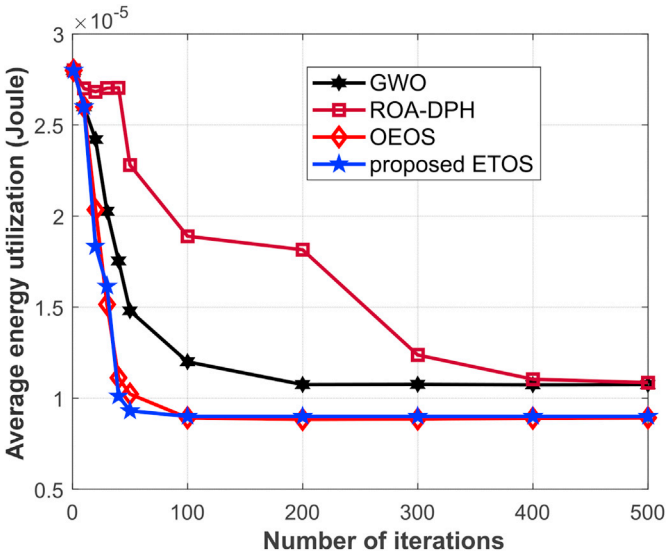

Fig. 5. Average energy utilization versus the number of iterations.

When the number of tasks is small (2–6 tasks), the execution time of OEOS and the proposed ETOS is nearly the same. However, when the number of tasks increases further, the proposed ETOS exhibits faster execution speed than OEOS, with performance gap growing rapidly. This demonstrates the superiority of the proposed ETOS over OEOS especially on application scenarios having a large number of tasks.

In Fig. 5, the performance evaluation is compared with the benchmark algorithms [47,49] in terms of statistical results and optimal energy utilization for processing user requests.

As shown in Fig. 5, compared to the GWO and ROA-DPH methods, the proposed ETOS algorithm performs the near-optimal solution (OEOS) with less than 5% difference from the optimal strategy and provides the best minimum energy utilization in less iterations (50). This is because, the penalty function is established to adjust the fitness function to find the best solution of the optimization problem which considers both resource availability and cost of task computation. Similarly, the established penalty function is applied to speed up the iterative convergence and enhance the correctness of the results.

Fig. 6 presents the impact of task size on energy consumption. Obviously, as the task size increases, the difference in ratio of energy consumption decreases. This is because when the input data is large, it requires more resources for uploading tasks to the MEC and processing which consumes more energy than small task size. Although, all strategies exhibit similar trend of decreasing difference in energy consumption ratio on increasing task size, the proposed ETOS performs closer to the optimal scheme (OEOS) and outperforms the two baseline schemes ROA-DPH and ATO, as shown in Fig. 6. This is essential because of the efficient energy utilization achieved by the proposed strategy, which in return improves data uploading rates and maximizes the energy saved by the system.

Fig. 7 demonstrates the comparison in total energy utilization versus processing capacity required to accomplish the task. It is observed that in all approaches, the total energy utilization increases with the required processing capacity. In view of Local execution, the energy utilization is higher compared to all schemes because MDs are constrained in both computing and energy resources. Therefore, offloading resource-intensive jobs to more resourceful MEC is a promising solution to save more energy in MDs. In view of the proposed ETOS, the total energy utilization is closer to OEOS and lower compared to ROA-DPH, ATO, and Local because the jobs are offloaded to the MECs that exploit cooperative task processing. This, in return, maximizes utilization of resources at the edge of the mobile network, thereby saving more energy that could be used for job transmission through the backhaul link to the remote

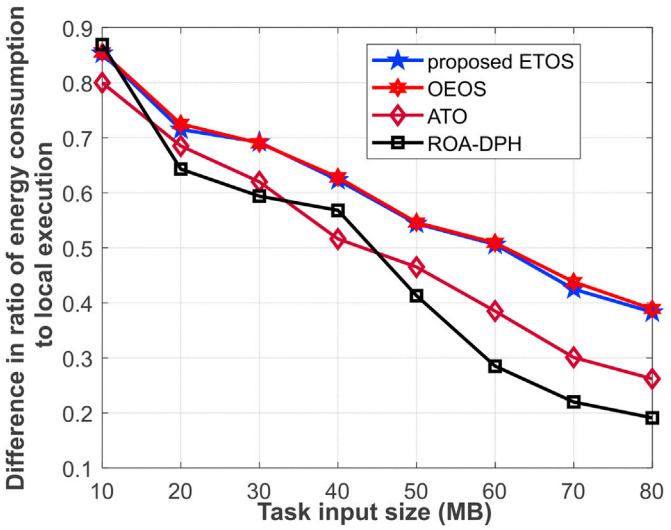

Fig. 6. Comparison of energy consumption for four approaches when the task input size varies.

servers. In addition, the offloading decision strictly considers the capacity constraints, proximity, and time constraint, as such minimizes the transmission energy through a long distance which improves energy consumption.

Fig. 8 shows the comparison of the proposed ETOS approach with three baseline approaches ROA-DPH, ATO, and OEOS in terms of the offloading utility by considering the effects of the data transmission rate. It is observed that the offloading utility increases with transmission rate for all approaches with a slight difference between the schemes when the transmission rate is low. In each case, the proposed ETOS performs near-optimal solution (OEOS), and it outperforms other baseline approaches of ROA-DPH and ATO in terms of offloading utility. This is because the proposed optimization strategy leverages the opportunity of MEC servers located closer to end users for task execution which is potential for minimizing latency and resource consumption such as energy and bandwidth compared to remote cloud. This in returns maximizes uploading rate of MDs which increases offloading utility, as shown in Fig. 8. In addition, the cooperative capability of MEC servers has brought opportunities for plenty resources that influence more task offloading to the MEC to minimize resource consumption by MDs.

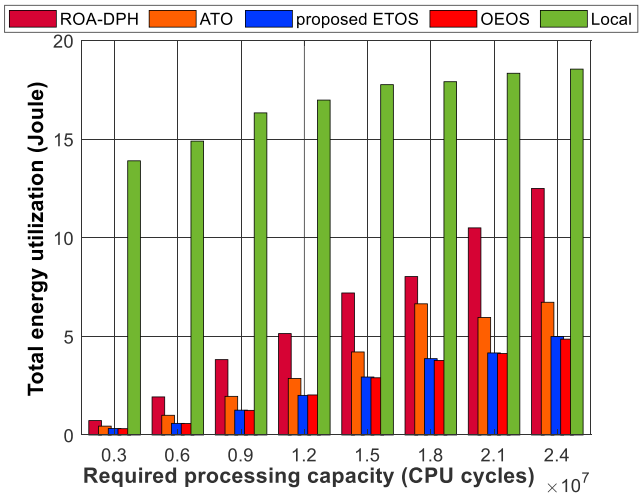

Fig. 7. Total energy utilization versus processing capacity required to accomplish the tasks.

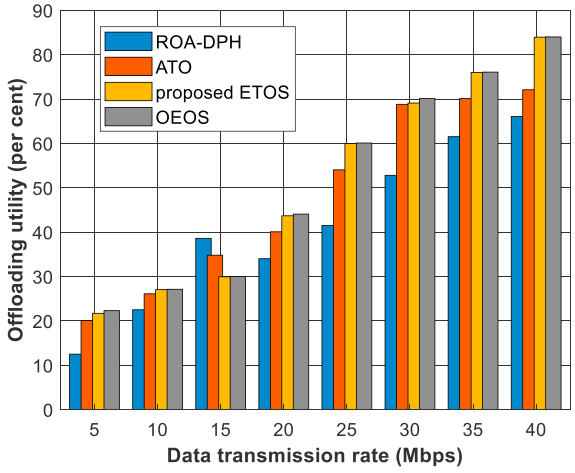

Fig. 8. Offloading utility versus data transmission rate.

6. Conclusion and future works

This paper addresses the task offloading problem focusing on cooperative MEC architecture. Initially, the task offloading is modeled as an optimization problem with the objective of reducing the overall energy consumption. Then, the energy-efficient task offloading algorithm based on swarm intelligent optimization techniques is proposed to ensure that minimal energy cost is achieved. Furthermore, the performance of the proposed algorithm is evaluated using simulation experiments with various parameters. The simulation results verify that the proposed approach is computational-efficient compared to benchmark methods. Moreover, the proposed strategy considerably outperforms other baseline approaches, such as OEOS, ROA-DPH, ATO, and Local execution in terms of energy consumption, execution time, and offloading utility.

In the future, the researchers envisage extending the scope of the work to an online scenario and evaluating the algorithm in the real environment of mobile edge computing.

Declaration of competing interest

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.

Acknowledgements

The work was supported by the Chinese Scholarship Council (CSC) under MOFCOM (No. 2017MOC010907), any opinions, findings, and conclusions are those of the authors and do not necessarily reflect the views of the above agency.

References

[1] Z.Q. Jaber, M.I. Younis, Design and implementation of real time face recognition system (RTFRS), Int. J. Compt. Appl. 94 (12) (2014) 15–22.
[2] S. Wang, Y. Zhang, H. Wang, Z. Wang, X. Wang, T. Jiang, Large scale measurement and analytics on social groups of device-to-device sharing in mobile social networks, Mobile Network. Appl. 23 (4) (2017) 1–13.
[3] Y. Zhang, J. Ren, J. Liu, C. Xu, H. Guo, Y. Liu, A Survey on emerging computing paradigms for big data, Chin. J. Electron. 26 (2017) 1–12.
[4] C.L. Irene, Y.L. Susan, The Internet-of-Things: review and research directions, Int. J. Res. Market. 34 (2017) 3–21.
[5] J. Ren, G. Yu, Y. Cai, Y. He, Latency optimization for resource allocation in mobile-edge computation offloading, IEEE Trans. Wireless Commun. 17 (2018) 5506–5519.
[6] B. Li, Z. Peng, P. Hou, et al., Reliability and capability based computation offloading strategy for vehicular ad hoc clouds, J. Cloud Comput. 8 (2019) 1–14.
[7] B. Zhou, A.V. Dastjerdi, R.N. Calheiros, S.N. Srirama, R. Buyya, A context sensitive offloading scheme for mobile cloud computing service, in: Proceeding of the 2015 IEEE 8th International Conference on Cloud Computing, IEEE, New York, 2015, pp. 869–876.
[8] Cisco Visual networking index, Global Mobile Data Traffic Forecast Update 2017–2022 White Paper, 2022. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.pdf. (Accessed 14 August 2021).
[9] G. Skourletopoulos, C.X. Mavromoustakis, P. Chatzimisios, G. Mastorakis, E. Pallis, J.M. Batalla, Towards the evaluation of a big data-as-a-service model: a decision theoretic approach, in: Proceeding of the 2016 IEEE INFOCOM, IEEE, San Francisco, 2016, pp. 877–883.
[10] K. Kaur, S. Garg, G.S. Aujla, N. Kumar, J.P.C. Rodrigues, M. Guizani, Edge computing in the industrial internet of things environment: software-defined-networks-based edge-cloud interplay, IEEE Commun. Mag. 56 (2018) 44–51.
[11] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, W. Zhao, A survey on internet of things: architecture, enabling technologies, security and privacy, and applications, IEEE Internet Things J. 4 (2017) 1125–1142.
[12] F. Bonomi, R. Milito, P. Natarajan, J. Zhu, Fog computing: a platform for internet of things and analytics, in: N. Bessis, C. Dobre (Eds.), Big Data and Internet of Things: A Roadmap for Smart Environments, Springer, 2014, pp. 169–186.
[13] Q. Fan, N. Ansari, Towards workload balancing in fog computing empowered IoT, IEEE Transactions on Network Science and Engineering 7 (1) (2020) 253–262.
[14] M. Patel, J. Joubert, J.R. Ramos, N. Sprecher, S. Abeta, Neal, et al., Mobile Edge Computing, Technical White Paper 1, 2014, pp. 1–36.
[15] M.P.J. Mahenge, C. Li, C.A. Sanga, Mobile edge computing: cost-efficient content delivery in resource-constrained mobile computing environment, Int. J. Mobile Comput. Multimed. Commun. 10 (3) (2019) 23–46.
[16] Q. Jia, R. Xie, T. Huang, J. Liu, Y. Liu, Caching resource sharing for network slicing in 5G core network: a game theoretic approach, J. Organ. End User Comput. 31 (4) (2019) 1–18.
[17] S. Dai, M. Liwang, Y. Liu, Z. Gao, L. Huang, X. Du, in: L. Zhu, S. Zhong (Eds.), Hybrid Quantum-Behaved Particle Swarm Optimization for Mobile-Edge Computing Offloading in Internet of Things, Springer Nature, 2018, pp. 350–364.
[18] X. Lyu, H. Tian, Adaptive receding horizon offloading strategy under dynamic environment, IEEE Commun. Lett. 20 (2016) 78–81.
[19] S. Guo, J. Liu, Y. Yang, B. Xiao, Z. Li, Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing, IEEE Trans. Mobile Comput. 18 (2019) 319–333.
[20] G.C. Kagadis, C. Kloukinas, K. Moore, J. Philbin, P. Papadimitroulas, C. Alexakos, P.G. Nagy, D. Visvikis, W.R. Hendee, Cloud computing in medical imaging, Med. Phys. 40 (7) (2013) 1–11.
[21] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies, The case for VM-based cloudlets in mobile computing, IEEE Pervasive Computing 8 (2009) 14–23.
[22] M. Chiang, Fog Networking: an Overview on Research Opportunities, 2016 arXiv: 1601.00835.
[23] A.R. Khan, M. Othman, S.A. Madani, S.U. Khan, A survey of mobile cloud computing application models, IEEE Communications Surveys and Tutorials 16 (2014) 393–413.
[24] F.B. Jemaa, G. Pujolle, M. Pariente, QoS-aware VNF placement optimization in edge-central carrier cloud architecture, in: Proceeding of the 2017 IEEE Global Communications Conference, 2017, pp. 1–7. Washington.
[25] M. Ryden, K. Oh, A. Chandra, Nebula: distributed edge cloud for data intensive computing, in: Proceeding of the 2014 IEEE International Conference on Cloud Engineering, 2014, pp. 57–66. Boston.
[26] L. Tong, Y. Li, W. Gao, A hierarchical edge cloud architecture for mobile computing, in: Proceeding of the 2016, IEEE International Conference on Computer Communications, San Francisco, 2016, pp. 1–9.
[27] European Telecommunications Standards Institute, GS MEC 001, 2016. March.

[28] J. Kwak, Y. Kim, J. Lee, S. Chong, Dream: dynamic resource and task allocation for energy minimization in mobile cloud systems, IEEE J. Sel. Area. Commun. 33 (2015) 2510–2523.
[29] Y. Mao, J. Zhang, K.B. Letaief, Dynamic computation offloading for mobile-edge computing with energy harvesting devices, IEEE J. Sel. Area. Commun. 34 (2016) 3590–3605.
[30] S. Sardellitti, G. Scutari, S. Barbarossa, Joint optimization of radio and computational resources for multi-cell mobile edge computing, IEEE transactions on signal and information processing over networks 1 (2014) 89–103.
[31] Y. Mao, J. Zhang, S.H. Song, K.B. Letaief, Power-delay tradeoff in multi-user mobile-edge computing systems, in: Proceeding of the 2016 IEEE Global Communication Conference (GLOBECOM), IEEE, Washington, 2016, pp. 1–6.
[32] J. Ren, Y. Zhang, K. Zhang, Exploiting mobile crowdsourcing for pervasive cloud services: challenges and solutions, IEEE Commun. Mag. 53 (2015) 98–105.
[33] M.P.J. Mahenge, C. Li, C.A. Sanga, Collaborative mobile edge and cloud computing: tasks unloading for improving users' quality of experience in resource-intensive mobile applications, in: Proceeding of the 2019 IEEE ICCCS, IEEE, Singapore, 2019, pp. 322–326.
[34] K. Cheng, Y. Teng, W. Su, A. Liu, X. Wang, Energy-efficient joint offloading and wireless resource allocation strategy in multi-MEC server systems, in: Proceeding of the 2018 International Conference on Communication, IEEE, 2018, pp. 1–6. Kansas City.
[35] Z. Wang, W. Liang, M. Huang, Y. Ma, Delay-energy Joint Optimization for Task Offloading in Mobile Edge Computing, 2018 arXiv: 1804.10416v1 [cs.NI].
[36] X. Cao, F. Wang, J. Xu, R. Zhang, S. Cui, Joint Computation and Communication Cooperation for Mobile Edge Computing, 2016 arXiv: 1704.06777v2 [cs.IT].
[37] J. Liu, Y. Mao, J. Zhang, K.B. Letaief, Delay-optimal computation task scheduling for mobile-edge computing systems, in: Proceeding of the 2016 IEEE ISIT, IEEE, Spain, 2016, pp. 1451–1455.
[38] C. You, K. Huang, H. Chae, B. Kim, Energy-efficient resource allocation for mobile-edge computing offloading, IEEE Trans. Wireless Commun. 16 (2017) 1397–1411.
[39] F. Wang, J. Xu, X. Wang, S. Cui, Joint offloading and computing optimization in wireless powered mobile-edge computing system, in: Proceeding of the 2016 IEEE ICC, IEEE, Paris, 2016, pp. 1–6.
[40] C. You, K. Huang, H. Chae, H, Energy efficient mobile cloud computing powered by wireless energy transfer, IEEE J. Sel. Area. Commun. 34 (2016) 1757–1770.
[41] A.P. Miettinen, J.K. Nurminen, Energy efficiency of mobile clients in cloud computing, in: Proceeding of the 2nd USENIX Conference HotCloud, 2010, pp. 1–7. Boston.
[42] Y. Wen, W. Zhang, H. Luo, Energy-optimal mobile application execution: taming resource-poor mobile devices with cloud clones, in: Proceeding of the 2012 IEEE INFOCOM, IEEE, Orlando, 2012, pp. 2716–2720.
[43] G. Nardini, A. Virdis, G. Stea, Modeling X2 backhauling for LTE-advanced and assessing its effect on CoMP coordinated scheduling, in: Proceeding of the 2016 1st International Workshop on Link and System Level Simulations, 2016, p. 46. Vienna.
[44] E. Emary, H.M. Zawbaa, C. Grosan, A.E. Hassenian, Feature subset selection approach by gray-wolf optimization, in: Proceeding of the Afro-European Conference for Industrial Advancement, Advances in Intelligent Systems and Computing, Springer, Berlin, 2015, pp. 1–13.
[45] Y. Yusof, Z. Mustaffa, Time series forecasting of energy commodity using grey wolf optimizer, in: Proceeding of the International Multi Conference of Engineers and Computer Scientists, 2015, pp. 1–6. Hong Kong.
[46] A.A. El-Fergany, H.M. Hasanien, Single and multi-objective optimal power flow using grey wolf optimizer and differential evolution algorithms, Elec. Power Compon. Syst. 43 (13) (2015) 1548–1559.
[47] G.M. Komaki, V. Kayvanfar, Grey wolf optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time, Journal of Computational Science 8 (2015) 109–120.
[48] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceeding of the IEEE International Conference on Neural Networks, IEEE, Perth, Australia, 1995, pp. 1942–1948.
[49] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Software 69 (2014) 46–61.
[50] N. Singh, S.B. Singh, Hybrid algorithm of particle swarm optimization and grey wolf optimizer for improving convergence performance, Journal of Applied Mathematics, Hindawi (2017) 1–16.
[51] G. George, A. Lozano, M. Haenggi, Distribution of the number of users per base station in cellular networks, IEEE Wireless Communications Letters 8 (2) (2019) 520–523.
[52] H. Guo, J. Liu, H. Qin, Collaborative mobile edge computation offloading for IoT over fiber-wireless networks, IEEE Network 32 (1) (2018) 66–71.
[53] H. Shahzad, T.H. Szymanski, A dynamic programming offloading algorithm for mobile cloud computing, in: Proceeding of the IEEE Canadian Conference on Electrical and Computer Engineering, IEEE, Vancouver, 2016, pp. 1–5.
[54] X. Liu, S. Guo, Y. Yang, Task offloading with execution cost minimization in heterogeneous mobile cloud computing, in: Proceeding of the 13th Mobile Ad-Hoc and Sensor Network International Conference, 2017, pp. 509–522. Beijing.