IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 2, 2020, accepted December 10, 2020, date of publication December 15, 2020, date of current version December 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3044932

Energy-Aware Scheduling in Edge Computing Based on Energy Internet

QING ZHANG[1], **XIAOYONG LIN**[2], **YONGSHENG HAO**[3], **AND JIE CAO**[4]
[1]Computer Engineering College, Jimei University, Xiamen 361021, China
[2]School of Cultural Industries and Tourism, Xiamen University of Technology, Xiamen 361024, China
[3]Network Center, Nanjing University of Information Science and Technology, Nanjing 210044, China
[4]School of Management Engineering, Xuzhou University of Technology, Xuzhou 221018, China

Corresponding authors: Xiaoyong Lin (xylin_xm@163.com) and Yongsheng Hao (yongshenghao@yahoo.com)

ABSTRACT Edge computing has been widely researched when 5G network and cloud platforms work together for people's life. The limitation of energy provided by the battery of the edge device hinders its application. This paper focuses on task scheduling in edge computing combined with the Energy harvesting technology (EH) and Energy Internet (EI). The edge node collects green energy by EH. And nodes exchange energy by EI. Energy Internet obtains green energy from edge nodes. Compared to green energy, we call energy from the power grid (not from the energy of edge nodes by energy harvesting technology) brown energy. How to reduce brown energy consumption is one of the most important problem in our paper. Previous works have not examined the energy attenuation between nodes, neither have they studied the immigration routes of virtual machines (VMs). This paper analyzes VM scheduling and models the energy consumption of VM immigrations, offloading tasks, and green energy transfer in edge computing. The paper proposes a heuristic assumption that there is only one VM in the system, and then presents three heuristics for the system with multiple VMs. The simulation results show that the proposed - immigrated VMs with the minimum energy transferring attenuation ratio method (METAR) is effective in reducing brown energy and total energy consumption, and improving the utilization rate of green energy. Compared to the Energy-Efficiency problem solution (EE-PRO) and maximize task energy consumption scheduling (MTS), METAR average reduces by 28.23% and 49.50% in brown energy consumption. At the same time, METAR average decreases by 5.67% and 11.52% in execution time.

INDEX TERMS Edge computing, green computing, heuristics algorithm, simulation.

I. INTRODUCTION

With the rapid growth of smartphones, wireless technology [1], mobile devices, online applications [2], especially with the emergence of 5G network [3], edge computing is becoming increasingly important and popular [4] because it improves the quality of people's lives in almost all aspects including work, society [5] and economy [6]. Edge computing also facilitates research on Internet of things [7], [8], virtualization [9], healthcare systems [10], vehicular networks [11], and other industries. However, energy limitations and limited computational ability hinder the application of edge devices [6].

To cope with the challenges of edge computing, tasks are offloaded from local nodes to nearby edge nodes [5] (or remote cloud) connected by cables and reticles to improve processing ability and battery working time [12].

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

Fig. 1 presents a three-layer edge-computing framework. By using routers, gateways, base stations, and other kinds of access points, the system improves the computational ability and prolongs the working time of a mobile edge node by offloading some tasks from a local edge node to other edge nodes with more energy and higher processing ability [13]. The majority of studies have focused on where a task should be offloaded. However, the impact of VM immigration on the performance of the edge node is uncertain because different routes between edge nodes and VMs in nodes have different metrics, such as bandwidth, and energy transferring attenuation ratio (ETAR) [12]. These metrics affect task processing time and the energy consumption of VM immigration and task processing. In this paper, we call the energy by Energy Internet (EI) brown energy.

As shown in Figure 1, there are three layers in the green edge computing framework. In the first layer, users submit tasks by mobile devices, pads, smartphones, or other devices. The second layer not supports communication (or

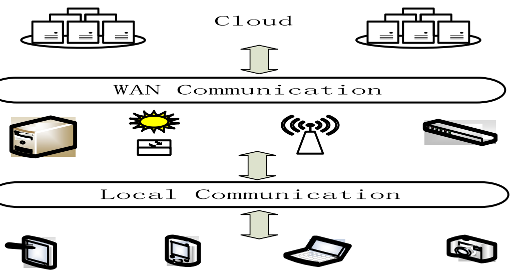


FIGURE 1. Three layers of edge computing.

improves processing ability) between edge nodes but also transfers energy between them with EI technology [14], [15]. The edge node in the third layer obtains energy by energy harvesting devices, and offloads the workload to the outside (other nodes or remote clouds) to the powerful systems (i.e. in the cloud), to extend the battery lifetime. EI technology makes it possible to transfer energy between edge nodes through cable or wireless [14], [15]. The Power over Ethernet (PoE) switch can transfer both data and energy. The selection of routes affects green energy loss during transferring because different routes differ in ETAR. The location of VM also affects energy consumption. On the one hand, different VM immigration routes consume different amounts of energy, and on the other hand, the task transferred through different routes requires different amounts of energy. This paper is the first study that combines EI technology, EH technology and edge computing. We take account of VMs immigration, energy transferring and task scheduling together in the paper. Tasks and green energy are scheduled by two rules: (1) add nodes (scheduling tasks and green energy on the node) to the VM with the minimum energy transferring attenuation ratio (METAR); (2) add nodes (scheduling tasks and green energy on the node) to the VM with the minimum energy consumption loss (ACL). Scheduling methods based on rules (1) and (2) are METAR and ACL, respectively. We also can use rule (1) and (2) alternately, called the method ALT.

The main contributions are listed as follows:

- We combine EI with edge computing to enhance edge computing in terms of energy consumption;
- We model edge computing with EI technology with multiple VMs and employ a method based on a global search to schedule tasks;
- Three heuristics are proposed to minimize the brown energy consumption based on three rules;
- Comparisons are made to evaluate the performance of the proposed methods and other methods.

The remainder of the paper is organized as follows. Section II introduces related work of edge computing and the EI technology used in edge computing. Section III presents the model used in this paper on green edge computing. Section IV provides a global search method to schedule resources and energy. In Section V, first, we suppose there is only one VM in the system, and present a heuristic algorithm to offload tasks and schedule energy; then, we propose three scheduling heuristics when the system has multiple VMs. In Section VI, simulations are conducted to verify the performance of the proposed method in energy consumption and other aspects.

II. RELATED WORKS

Section II(A) gives a summary of energy-aware scheduling in edge computing. Section II(B) gives an introduction of EI and EH technology used in edge computing. Section II(C) gives an example to illustrate the motivation of the paper.

A. ENERGY-AWARE EDGE COMPUTING

In edge computing, many factors affect the energy consumption of scheduling tasks. There are mainly four factors, which are local resources, remote resources, tasks, and the network [16]. Local resources provide processing power for tasks located locally. Resources with high energy utilization (energy power/computing speed) can always reduce the probability of transferring tasks to other nodes or remote clouds to reduce energy consumption. The resource determines the execution location according to the energy consumption in various places. The routes between local nodes and remote resources affect energy consumption for transferring input and output data. The energy consumption is also affected by the size of data (input and output files) and bandwidth when executed on nearby nodes or remote clouds. For example, if tasks with a small value of input and output files are executed on remote resources, energy computation may decrease (if remote resources have higher energy efficiency) [17]. In contrast, these tasks have high sizes in input and output files and low execution time, if they are executed in other nodes, thus increasing energy consumption and prolonging execution time.

To save energy of edge nodes, many methods have been utilized in the edge-computing environment. Offloading tasks may cost more time and local execution may require more energy in some cases (such as with large output files). Methods always attempt to make a tradeoff between time and energy. Zhang *et al.* [18] presented an energy-aware offloading scheme that jointly optimizes communication and computation resource allocation with limited energy and sensitive latency. They proposed an iterative search algorithm to optimize local computing frequency scheduling, channel allocation, power allocation, and computation offloading. L. Cui *et al.* [19] attempted to make a tradeoff between energy consumption and latency for user demands of various IoT applications. They formalized the multi-objective optimization problem (minimizing average energy consumption and average latency time) and applied the NSGA-II method to solve it. These scheduling methods and other important metrics are proposed to maximize energy efficiency. H. Sun *et al.* [20] defined the computation efficiency, which is the number of calculated data bits divided by the corresponding energy consumption. The scheduling target maximizes the sum of the computation efficiency among users with weighting factors. They solved the problem with iterative and gradient descent methods. Q. Wang *et al.* [21]

proposed a distributed algorithm consisting of four aspects: clock frequency configuration, transmission power allocation, channel rate scheduling, and offloading strategy selection. They employed an offloading selection strategy to determine where a task should be executed and then optimized the clock frequency for local execution, transmission power allocation, and queue delay in mobile edge computing. These methods [18]–[24] always formulated the energy-aware scheduling problem as a problem of multiple target scheduling and utilized some methods to solve this problem. The two most important goals are to minimize execution time and minimize energy consumption.

In addition to the previously mentioned methods, some researchers have solved the problem by using other methods. S. Yang *et al.* [25] demonstrated how to place cloudlets on the network and allocate each requested task to cloudlets and a public cloud with the minimum total energy consumption without violating the delay requirement of each task. Data transfer consumed a substantial amount of energy for edge computing. L. Liu *et al.* [26] *et al.* utilized a local area network to collect data from users and conducted data compression to reduce the energy for data transfer. Besides, energy consumption is different when sending and receiving data. J. Zheng *et al.* [23] considered both downlink and uplink in communication and computing resource allocation and proposed an efficient joint downlink and uplink offloading algorithm for ultradense Het Nets. X. Zhao *et al.* [27] used multi-long short term memory (LSTM) based on a prediction model to predict the traffic for edge devices. Based on the prediction results, a mobile data offloading strategy based on cross-entropy is proposed to maximize the system throughput by determining which users to be offloaded to Wi-Fi systems. Z. Muhammad *et al.* [28] suggested game-theoretic resource management techniques to minimize infrastructure energy consumption and costs while meeting QoS of users. B. Ali *et al.* proposed [29] a Volunteer Supported Fog Computing (VSFC) that tries to explore the interplay of these two distributed computing domains and targets to reduce inherent communication delays of cloud computing, energy consumption, and network usage.

B. ENERGY HARVESTING, ENERGY INTERNET TECHNOLOGY COMBINED EDGE COMPUTING

Energy harvesting (EH) technology has been extensively applied in edge computing [16]. EH captures ambient renewable energy, including solar radiation, wind, and human motion energy [18]. EH is controllable, stable, and beneficial to edge devices [30]. Y. Mao [16] investigated a green MEC system with EH devices and developed the Lyapunov optimization-based dynamic computation offloading algorithm for the problem. The offloading method takes into account the execution latency and task failure and makes decisions on offloading considering the CPU-cycle frequencies for mobile execution, and the transmit power for computation offloading. To investigate the tradeoff between energy consumption and execution delay in an EH-supporting MEC

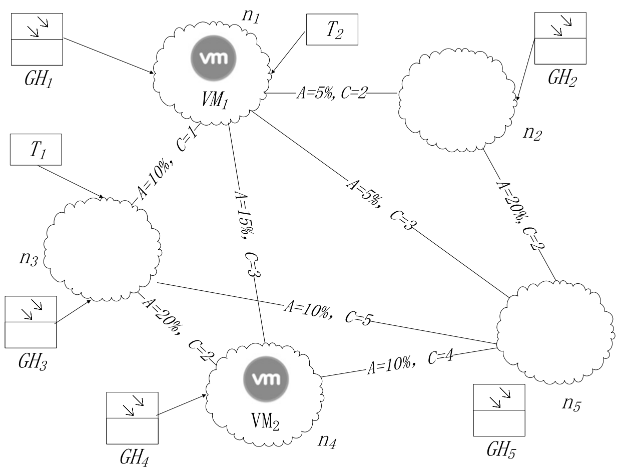


FIGURE 2. Edge computing supported by EI technology.

system, G. Zhang *et al.* [31] examined the stability of buffer queues and battery level as constraints, and formulated the offloading into an average weighted sum of energy consumption and execution delay minimization of a mobile device. These methods all use EH technology to obtain green energy and to ensure some metrics, such as time limit and energy consumption. In Figure 2, we call the energy from $GH_1 \sim GH_5$ green energy, other energy is brown energy.

EH technology only obtains green energy from outside and does not consider how to transfer energy from different devices. The Energy Internet (EI) transmits flexible and customizable energy between nodes in a power grid. In edge computing, some nodes get green energy (such as solar energy). EI transfers energy between edge devices, especially to some nodes with an amount of energy or lower system load. EI causes a new challenge in task allocation and schedule energy. Some nodes have much energy, we can transmit them to other nodes. On the contrary, some nodes need to get energy from other nodes. A task consumes different energy when it is offloaded to different nodes. All those bring a challenge to the new environment. Most works focus on how to offload tasks from edge devices to remote clouds. In the new environment, we also need to consider the route of energy transfer. Our targets are to maximize the utilization of green energy and reduce brown energy consumption. Different routes have different energy decay ratios and different energy losses for energy transfer, route selection influences energy consumption. To maximize the utilization of green energy and reduce brown energy consumption, L. Gu *et al.* [13] investigated the energy cost minimization by jointly considering VM immigration, task allocation, and green energy scheduling, and proposed a heuristic algorithm to solve the problem of offload tasks and energy transfer. However, they assumed that only one VM in the system. Most of the time, multiple VMs exist in the real system. We attempt to solve the offloading problem with EI technology. This paper focuses on task scheduling by considering VM immigration, green energy transfers, and targets to minimize brown energy.

C. MOTIVATION

An example is provided in Figure 2. There are five nodes ($n_1 \sim n_5$) and two VMs ($VM_1 \sim VM_2$) in the edge computing environment. Every node has a green energy harvesting device ($GH_1 \sim GH_5$). Every node can harvest green energy from solar energy. VM_1 and VM_2 are located on node n_1 and n_4. Some nodes have direct links connected with other nodes, while others are not. For example, n_2 and n_3 do not have any links with each other, but they can communicate via n_1. "A" and "C" on the links respectively represent the attenuation ratio of energy transfer and the number of hops between two nodes.

In Fig. 2, several routes between node n_2 and n_5, including: $\{n_2, n_5\}$, $\{n_2, n_1, n_5\}$, $\{n_2, n_1, n_3, n_5\}$,$\{n_2, n_1, n_4, n_5\}$ and $\{n_2, n_1, n_3, n_4, n_5\}$. The route $\{n_2, n_1, n_5\}$ has the minimum attenuation ratio (1-0.95*0.95) of energy transfer. The selection of routes is very important for VM immigration, task offloading, and energy transfer. Different routes may lose different energy of edge nodes. A bad route increases the energy consumption for VM immigration and task offloading. For example, task T_1 is submitted from n_3, to execute it, we can immigrate a VM from other nodes (n_1, n_4), or offload the task to the node with a VM (n_1, n_4). If the green energy collected by edge nodes is not enough for processing all tasks, we obtain energy from outside of the system, which is also known as brown energy. The brown energy always is obtained by the power grid and transferred by cable. How to reduce brown energy is one of the most important focuses of this paper.

III. SYSTEM MODEL

In this Section, we give models used in our paper. Table 1 illustrates the parameters used in the paper. Figure 2 provides an example of the edge-computing environment used in the paper.

A. NETWORK MODEL

We employ an indirect graph $G = (I, E)$ to denote the network of edge computing, which includes the set of nodes I and the set of network edges E. The system has N nodes and VMN VMs. The set $e_{i,j}$ $(i, j \in I)$ denotes whether there is a link between node n_i and n_j. If $e_{i,j}$ is equal to 1, there is a link between them; otherwise, no link exists between them. $d_{i,j}$ is the distance between node n_i and n_j. For the network, we always use hops between nodes as the distance between various nodes. We specifically set the distance as 0 when $i = j$. v_i is a binary variable, which indicates whether node n_i has a VM. $ea_{i,j}$ is the attenuation ratio of energy transfer from node n_i to n_j. We model the system as multiple slot times, each slot having the same attenuation value of energy transfer.

The energy consumption involved in the edge-computing environment is influenced by the following parameters: energy transfer attenuation ratio, green energy supply, energy consumption of task processing, energy consumption of offloading tasks and VM immigration.

TABLE 1. Notation used in the paper.

Notation	Meaning
$G = (I, E)$	To denote the network of edge computing
I	A set of nodes
E	A set of network edges
VMN	Number of VMs
N	Number of edge nodes
n_i, n_j	ith edge node and jth edge node
v_l	A binary variable that indicates whether n_l has a VM
ev_i	Energy consumption of VM immigration without attenuation for node n_i
$e_{i,j}$	A binary variable that indicates whether n_i has a link to n_j
$d_{i,j}$	Distance between n_i and n_j
$ea_{i,j}$	Energy transferring attenuation ratio from n_i to n_j
$minea_{i,j}$	Minimum energy transferring attenuation ratio from n_i to n_j (n_i is an adjacent node of n_j)
$metar_{i,j}$	Minimum energy transferring attenuation ratio from n_i to n_j (n_i is not an adjacent node of n_j)
$mt_{i,j}$	Energy transferring attenuation ratio between n_i and n_j
$ng_{i,s}$	New arriving green energy on node n_i during sth slot time.
$gel(i,j)$	Energy loss of transferring energy from node n_i to n_j
t	Task t
$egtp(t,j)$	Energy consumption for processing task on n_j (a VM on node n_j)
$egtd(t)$	Energy consumption for transferring data of task t without ETA
$egt(t,i,j)$	Total energy consumption for task t moved from n_i to n_j
v	VM v, also denoted as node v
$egvd(v)$	Energy consumption of VM immigration v without ETA
$egv(v,i,j)$	Total energy consumption of VM v immigrated from n_i to n_j
tec_j	Total energy consumption of node n_j located at VM
pe_j	Provided energy on node n_j
bec_j	Brown energy consumption of node n_j
s	sth slot time
$E_{i,s}$	Energy of node i during slot time s
$eegt(s,i,j)$	Energy consumption when tasks on node n_i are executed on node n_j during slot time s
$Diste^s_{i,j}$	Energy consumption of transferring energy from n_i to n_j
$Distv^s_{i,j}$	Energy consumption of VM immigration from n_i to n_j
$Dist^s_{i,j}$	Energy consumption from node n_i to n_j

B. ENERGY TRANSFERRING ATTENUATION RATIO BETWEEN TWO DIFFERENT NODES

If green energy in a node is insufficient for supplying the processing tasks of the VM in the node, we may schedule green energy from relatively nearby nodes that do not host a VM. Assume that green energy transfer from node n_i to n_j, we select the route which passes through the node-set $NS_i = \{n^l_{i,j} | 1 \leq l \leq L\}$, where L is the total number of nodes between node n_i and n_j. The minimum attenuation ratio of energy transfer between node n_i and n_j is as follows ($mt_{i,j}$):

$$mt_{i,j} = \prod (1 - minea_{x,y}) \tag{1}$$

where $x = n^l_{i,j}, y = n^l_{i,j} + 1,\ 1 \leq l \leq L$. Most of the time, there are several links between node n_i and n_j, we suppose that the number of routes is K. For the kth route, the relevant minimum attenuation ratio of energy transfer is $met^k_{i,j}$, then

$$metar_{i,j} = \min(met^k_{i,j} | 1 \leq k \leq K) \tag{2}$$

For example, in Figure 2, several routes traverse from node n_2 to n_5, including:$\{n_2,n_5\}$, $\{n_2, n_1, n_5\}$, $\{n_2, n_1, n_3, n_5\}$,

$\{n_2, n_1, n_4, n_5\}$ and $\{n_2, n_1, n_3, n_4, n_5\}$. The route $\{n_2, n_1, n_5\}$ has the minimum attenuation ratio (1-0.95*0.95) of energy transfer.

$minea_{x,y}$ is the minimum attenuation ratio of energy transfer from node x to node y. We take $ea_{x,y}$ as the distance between node x and node y, whose values can be obtained by the method proposed by Dijkstra [32] Floyd method.

C. GREEN ENERGY LOSS

When the system decides to transfer green energy between different nodes, it always prefers to select the route with the smallest attenuation ratio of energy transfer. The total green energy loss when we move energy from node n_i to n_j during the sth slot time is:

$$gel(i,j) = ng_{i,s} * (1 - metar_{i,j}) \tag{3}$$

where $ng_{i,s}$ is the new arriving green energy on node n_i during the sth slot time.

D. ENERGY CONSUMPTION OF PROCESSING TASKS

The energy consumption for processing a task includes two parts: the energy for processing tasks on a VM and the energy consumption of moving the task (data and code) from the submitted node to a VM. For task t, assume that it is submitted from node i and is moved to node j, the total energy consumption $egt\ (t, i, j)$:

$$egt\ (t, i, j) = egtp\ (t, j) + \frac{egtd(t)}{1 - metar_{i,j}} \tag{4}$$

where $egtp\ (t, j)$ is the energy consumption for processing a task on node j (a VM on the node), and $egtd(t)$ is the energy consumption for transferring data of task t without energy transfer attenuation.

E. ENERGY CONSUMPTION OF VM IMMIGRATIONS

For VM v, we assume that it is submitted from node i and immigrated to node j. The total energy consumption $egv\ (v, i, j)$ is:

$$egv\ (v, i, j) = \frac{egvd(v)}{1 - metar_{i,j}} \tag{5}$$

where $egvd\ (v)$ is the energy consumption of VM immigration v from node i to node j without energy transfer attenuation.

F. BROWN ENERGY CONSUMPTION

For node j (with a VM on it), the total energy consumption tec_j is computed as follow:

$$tec_j = \sum_t egt\ (t, i, j) + egv\ (v, i, j) \tag{6}$$

$ng_{i,s}$ is the new arriving green energy on node n_i during sth slot time. The provided energy on node j is:

$$pe_j = \sum\nolimits_i gel(i, j) + ng_{i,s} \tag{7}$$

The brown energy consumption of node j is:

$$bec_j = \begin{cases} tec_j - pe_j & if\ pe_j < tec_j \\ 0 & if\ pe_j \geq tec_j \end{cases} \tag{8}$$

G. COMPREHENSIVE ANALYSIS OF SCHEDULING TASKS AND IMMIGRATING VM

In this subsection, we refer to the energy provided by any node as the positive energy, which includes the green energy obtained in the last slot time ($lg_{i,s}$) and the newly arrived green energy ($ng_{i,s}$). We refer to the energy used for VM immigration as negative energy (ev_i). Hence, $E_{i,s}$ denotes the energy of node i during the slot time s and is expressed as follows:

$$E_{i,s} = lg_{i,s} + ng_{i,s} \tag{9}$$

If the energy on node i is transferred to node j, the energy lost $Diste^s_{i,j}$ is

$$Diste^s_{i,j} = E_{i,s} * metar_{i,j} \tag{10}$$

If a VM exists on node i, it is immigrated to node j, the energy consumption is $egvd\ (i)$ while the attenuation ratio of energy transfer is zero. The energy consumption $Distv^s_{i,j}$ is

$$Distv^s_{i,j} = egvd(i)/(1 - metar_{i,j}) \tag{11}$$

Because the route with the minimum attenuation ratio of energy transfer, always has the minimum energy loss for the energy transfer and VM immigration. The energy transferring and VM immigration may choose the same route to reduce energy consumption. If no VM exists on node i, $Distv^s_{i,j} = 0$.

If tasks on node i are submitted to node j (A VM on node j), then the energy consumption of processing tasks $Distt^s_{i,j}$ is

$$Distt^s_{i,j} = eegt\ (s, i, j) \tag{12}$$

where variable $Dist^s_{i,j}$ denotes the energy consumption from node i to node j as follows:

$$Dist^s_{i,j} = Diste^s_{i,j} + Distv^s_{i,j} + Distt^s_{i,j} \tag{13}$$

IV. SYSTEM ANALYSIS

In this section, first, we propose the scheduling method without considering the complexity of the system. We refer to the method as the offloading method without considering algorithm complexity (OFFWC). As the VM immigration always requires some time (1∼5 minutes), we assume that there is only a chance for VM immigration during one slot time. The scheduling method consists of two major steps: (1) finding all possible routes for VM immigration and energy transfer; and (2) finding all possible task offloading solutions. We select the solution with the minimum value in brown energy consumption.

A. FINDING ALL POSSIBLE ROUTES

There are VMN VMs and N edge nodes in the system. The challenge is to obtain VMN different nodes in N different edge nodes. The number of possible solutions to immigrate VMs is:

$$C_N^{VMN} = N^{VMN} \tag{14}$$

The set $vmp\,[VMIM]\,[N]$ is applied to denote the positions of the VMs after the VM immigrations. One row in vmp is a policy for VM immigration. In the VM immigrations, we always select the route with the minimization energy transferring attenuation ratio to reduce energy consumption. According to the method in Section III, the total energy consumption of VM immigrations is conserved in the set $egmi\,[VMIM]$, and the related green energy consumption and the brown energy consumption is $egmig\,[VMIM]$ and $egmib\,[VMIM]$, respectively. Here $((1 \leq temp \leq VMIM))$,

$$egmi\,[temp] = egmig\,[temp] + egmib\,[temp] \tag{15}$$

In a time slot, the time is too short to migrate a VM twice, so, we only take into account one immigration in the scheduling.

B. FINDING ALL POSSIBLE TASK OFFLOADING SOLUTIONS

For time slot s, there are T_s tasks and VMN VMs in the scheduling. The problem is how to assign the T_s tasks on $2{*}VMN$ VMs. We allocate the tasks by the route with the attenuation ratio of energy transfer to reduce energy consumption. The number of possible allocation methods TAM is:

$$TAM = (2 * VMN)^{T_s} \tag{16}$$

Every row is an allocation method, and the total energy consumption is $egt\,[TAM]$. The related green energy consumption and the brown energy consumption is $egtg\,[TAM]$ and $egtb\,[TAM]$. Here $((1 \leq temp \leq TAM))$,

$$egt\,[temp] = egtg\,[temp] + egtb\,[temp] \tag{17}$$

From Section IV(A) and IV(B), we obtain all possible solutions to allocate resources. We have three targets: (1) to minimize the total energy consumption, (2) to minimize the brown energy consumption, and (3) to maximize the total green energy.

$$tar_1 = \sum\nolimits_1^{VMIN} egmi\,[temp] + \sum\nolimits_1^{TAN} egt\,[temp] \tag{18}$$

$$tar_2 = \sum\nolimits_1^{VMIN} egmib\,[temp] + \sum\nolimits_1^{TAN} egtb\,[temp] \tag{19}$$

$$tar_3 = \sum\nolimits_1^{VMIN} egmig\,[temp] + \sum\nolimits_1^{TAN} egtg\,[temp] \tag{20}$$

The three targets have weights: α_1, α_2 and α_3, respectively. So, target tar is:

$$Tar = \alpha_1 * tar_1 + \alpha_2 * tar_2 - \alpha_3 * tar_3 \tag{21}$$

We select the method that has the minimum Tar. The complexity of the method is:

$$\mathrm{O}\,(VMIM * TAM) = \mathrm{O}(N^{VMN} * VMN^{T_s}) \tag{22}$$

The complexity of this method is too high. Therefore, we give three heuristics to decrease the complexity of the method in the following section.

V. HEURISTICS FOR ENERGY-EFFICIENT TASK ALLOCATION IN EDGE COMPUTING

In this section, we firstly present a scheduling method when there is only one VM in the system. And then, we propose three heuristic scheduling methods when multiple VMs exist. Finally, we give the complexity of the three proposed heuristics in Section V(C).

Algorithm 1 H-One(v, I) // v Is The Node Located a VM

```
1: $mintar = +\infty$, $tarn = null$;
// $mintar$ is minimizing target value of $Tar$ (Formula 21), $tarn$ is VM immigration node;
2: **For** every node ($n$) in $I - v$ // select node $n$ as the VM immigration target
3:   **For** every node ($m$) in $I - v - n$
4:     **If** $Dist^t_{m \to v} < Dist^t_{m \to n}$
5:       Add node $m$ to set $A$;
6:     **Else**
7:       Add node $m$ to set $B$;
8:     **EndIF**
9:   **EndFor**
10:  $Tar_1$ = Schedule($A$, $v$);
11:  $Tar_2$ = Schedule($B$, $n$);
12:  **If** $mintar > (Tar_1 + Tar_2)$
13:    $mintar = (Tar_1 + Tar_2)$;
14:    $tarn = n$;
15:  **EndIf**
16: **EndFor**
```

A. HEURISTICS FOR ONLY ONE VM IN EDGE COMPUTING

Assume that there is only one VM in the system and is located on node v. There are two major problems related to the scheduling: (1) Which node is selected as the VM immigration targets? (2) How should tasks and energy be scheduled to the initial node and the immigration node of a VM? This is the target of Algorithm 1. In Algorithm 1, we attempt to check the value of the target function of formula (21) to determine which node should be the immigration node of the VM. We assume that every target has the same weight, $\alpha_1 = \alpha_2 = \alpha_3$. We select the immigration target node by selecting the node that has the minimum value for the target function in Formula (21).

First, we assume that node (n) is selected as the immigration target node of the VM (line 2, Algorithm 1, same in the following paragraph). According to the energy consumption between two nodes (formula 22), nodes are divided into two sets: A and B. Nodes with a small value in energy consumption to node v are set in set A, and the remaining nodes are set in set B. Line 10 gives the scheduling target values when tasks in A are executed on node v. Line 11 gives the scheduling target values when tasks in B are executed on node n. We select the nodes with minimum target function (lines 10∼15).

Algorithm 2 gives details of scheduling tasks in set A to node v. First, we sort nodes in A by the ascending order of the energy consumption from node i to v ($Dist^t_{i \to v}$) (line 1, in Algorithm 2; same in the following paragraph). Then, we offload tasks from nodes in A to node v (line 2); at the same time, we transfer green energy from nodes in A to node v as requested (line 3). We only transfer energy just enough for processing tasks on node v. During energy transfer, we select nodes by the ascending order of attenuation ratio of energy

Algorithm 2 Schedule(X, v) // v Is the Node Located a VM, and X Is a set of Nodes Whose Tasks Should Be Executed on Node v; Returns the Target Value of Tar

```
1: Sort nodes in $X$ in the ascending order of $Dist^{s}_{x,v}$ (x ∈ X)
2: Offloading tasks on nodes in $X$ to node $v$;
3: Transferring green energy from nodes in $X$ to node $v$ as request;
4: Obtain brown energy from the system.
5: Calculate the target value of $Tar$ as formula (16).
```

transfer (formula 2) (line 3). If energy obtained from set A is not enough for processing all tasks in A, we obtain brown energy (line 4). Algorithm 2 returns the value of the target function (line 5).

Algorithm 2 obtains the target value of Tar when the tasks on nodes in set X are executed on the VM v. We first sort nodes in X in the ascending order of $Dist^{s}_{x,v}$ (x ∈ X) (line 1, in Algorithm 2; same in the following paragraph), and offload tasks from nodes in X to node v (line 2). Simultaneously, we transfer the energy from nodes in X to node v (line 3). Last, we obtain the brown energy (line 4) and the value of the target value Tar (line 5).

B. HEURISTICS FOR SEVERAL VMS IN EDGE COMPUTING

Algorithm 1 assumes that there is only one VM in the system. However, there are multiple VMs in the system. In this subsection, we attempt to solve the scheduling problem with several VMs in the system. Set V denotes the nodes which a VM is located:

$$V = \{v_{temp} | temp \in [1, VMN\} \quad (23)$$

In Section V(A), we give the scheduling method when there is only one VM in the system. When there are several VMs, if we split the system into multiple sets and each set has one VM, we can repeatedly use "H-One(v)" to schedule tasks and transfer energy. Thus, our scheduling target becomes how to divide nodes into multiple sets.

First of all, we take every node with a VM as the initial node of a set, and then add the exiting nodes from those initial nodes to the related sets by two rules: (1) add nodes to the set with the minimum energy transferring attenuation ratio (METAR) (Formula 2); (2) add nodes to the set with the energy consumption loss (ECL) (Formula 23). Algorithm 3 and Algorithm 4 give the heuristic based on rule (1) and rule (2), respectively.

Algorithm 3 initialize VMN sets (every set has a VM) and add nodes to every set by adding nodes to the set which has METAR. In Algorithm 3, lines 1∼3 (Algorithm3, same in the following paragraph) initialize VMN sets and each set has a VM. Line 4 obtains exiting nodes (LN) that do not belong to any sets. $metar_{ln,vt}$ (line 10) is the minimum attenuation ratio of energy transfer from node n_{ln} to node n_{vt}. For every $S[temp]$ in S, if a node (has not been allocated to any sets) has the minimum METAR (lines 10∼12), we add the node to $S[temp]$ (lines 14). $seln$ records the node with the minimum

Algorithm 3 METAR($VMNS[temp]$) // VMN Is the Number of VMs

```
1: For every node $v_{temp}$ in V
2:   Add $v_{temp}$ into the set $S[temp]$
3: End
4: $LN = I - V$;
5: While $LN$ is not null
6:   For every set $S[temp]$ in $S$
7:   $vt = v_{temp}$;  //$vt$ is the current node that locates a VM
8:   $minmet = +\infty$;
9:    For every node $ln$ in $LN$
10:     If $minmet > metar_{ln,vt}$
11:      $minmet = metar_{ln,vt}$, $seln = ln$;
12:     EndIf
13:    EndFor
14:    Add node $seln$ to $S[temp]$
15:   EndFor
16: EndWhile
17: For every set $S[temp]$ in $S$
18:   H-One($v_{temp}$, $S[temp]$);
19: End
```

METAR. We examine each set individually, and each set gets a new node each time (lines 6∼15). We get all sets and schedule every set by Algorithm 1 (lines 17∼19).

Algorithm 4 initializes VMN sets (every set has a VM) and adds nodes to every set by adding nodes to the set that has the energy consumption loss. In Algorithm 4, lines 1∼3 initial VMN sets and each set initializes with a VM, which is similar to Algorithm 3. Line 4 obtains the nodes that do not belong to any sets. $minmet$ denotes the minimum attenuation ratio of energy transfer and $seln$ is the selected node. For every $S[temp]$ in S, if a node (has not been allocated to any sets) has the minimum ECL (lines 10∼12), then we add the node to $S[temp]$ (lines 14). We examine each set individually, and each set gets a new node each time (lines 6∼15). We get all sets and schedule every set by using Algorithm 1 (lines 17∼19).

In addition to Algorithm 3 and 4, we can alternately divide sets by using two rules. We add a node to every set by using rule 1 (lines 7∼17) and then by using rule 2 (lines 9∼29). We repeat these steps until all nodes are allocated to sets. Algorithm 5 gives the details when we alternately use two rules.

Algorithm 3 initialize VMN sets (every set has a VM) and add nodes to every set by adding nodes to the set which has (1) METAR, and (2) the energy consumption loss alternately. In Algorithm 5, we first initialize VMN sets that they are initialized with a VM (lines 1∼3, Algorithm 5; same in the following paragraph). LN is a set of nodes without VMs. Second, for each node in LN, we alternately use the minimization policy for METAR (lines 7∼17) and minimum energy transfer attenuation ratio (lines 19∼29). In line 8, vt denotes the current node where a VM is located. $minmet$ is the node with the minimum METAR, and $seln$ is the relative

Algorithm 4 ACL ($VMNS[temp]$) // VMN Is the Number of VMs

```
1: For every node $v_{temp}$ in $V$
2:   Add $v_{temp}$ into the set $S[temp]$
3: End
4: $LN = I - V$;
5: While $LN$ is not null
6:   For every set $S[temp]$ in $S$
7:   $vt = v_{temp}$; //$vt$ is the current node with a VM
8:   $minmet = +\infty$;
9:    For every node $ln$ in $LN$
10:     If $minmet > Dist^{t}_{ln\to vt}$
11:      $minmet = Dist^{t}_{ln\to vt}$, $seln = ln$;
12:     EndIf
13:    EndFor
14:    Add node $seln$ to S[$temp$]
15:   EndFor
16: EndWhile
17: For every set $S[temp]$ in $S$
18:   H-One($v_{temp}$, $S[temp]$);
19: End
```

node. Last, line 15 adds the selected node to *seln* to $S[temp]$. We apply the minimization policy of the attenuation ratio of energy transfer policy (lines 19∼29). Conversely, *minmet* means the minimum energy transferring attenuation ratio.

C. COMPLEXITY ANALYSIS

From Table 1, we know that the number of nodes and the number of VMs is N and VMN.

For Algorithm 3, the complexity of Lines 1∼3 is O(VMN), the complexity of Lines 5, 6 and 9 is O(N), VMN and O(N), respectively. So, the complexity of METAR is:

$$\begin{aligned} O\,(\text{METAR}) &= O\,(N) + O(N * VMN * N) \\ &= O(N * VMN * N) \end{aligned}$$

Same for other methods:

$$\begin{aligned} O\,(\text{ACL}) &= O(N{*}VMN * N) \\ O\,(\text{ALT}) &= O(N{*}VMN * N) \end{aligned}$$

VI. SIMULATIONS

In this section, first, we give the detail related to the simulation environment is subsection VI(A). Then, we compare the proposed method and other methods in subsection VI(B). Finally, we discuss the reason for different performance in subsection VI(C).

A. SIMULATION ENVIRONMENT

According to the relevant references [13], [33]–[36], the parameters used in the paper are listed in Table 2. "Rand(a, b)" returns a random value between a and b. For example, the attenuation ratio of energy transfer has a range of [0, 0.25], which means that its maximum and minimum values are 0 and 0.25, respectively. The average number of tasks that arrive in every slot is changed from 8 to 20 with

Algorithm 5 ALT ($VMNS[temp]$) // VMN Is the Number of VMs

```
1: For every node $v_{temp}$ in $V$
2:   Add $v_{temp}$ into the set $S[temp]$;
3: End
4: $LN = I - V$;
5: While $LN$ is not null
6:   If $check == 0$
7:    For every set $S[temp]$ in $S$
8:    $vt = v_{temp}$;  //$vt$ is the current node with a VM
9:    $minmet = +\infty$;
10:     For every node $ln$ in $LN$
11:      If $minmet > metar_{ln,vt}$
12:       $minmet = metar_{ln,vt}$, $seln = ln$;
13:      EndIf
14:     EndFor
15:     Add node $seln$ to $S[temp]$
16:    EndFor
17:   $check = 0$;
18:   Else
19:    For every set $S[temp]$ in $S$
20:    $vt = v_{temp}$;  //$vt$ is the current node with a VM
21:    $minmet = +\infty$;
22     For every node $ln$ in $LN$
23:      If $minmet > Dist^{t}_{ln\to vt}$
24:       $minmet = Dist^{t}_{ln\to vt}$, $seln = ln$;
25:      EndIf
26:     EndFor
27:     Add node $seln$ to S[$temp$]
28:    EndFor
29:   $check = 1$;
30:   EndIf
31:   For every set $S[temp]$ in $S$
32:    H-One($v_{temp}$, $S[temp]$);
33:   End
34: EndWhile
```

a step of 2. The energy consumption of VM immigration is a rand in [3, 30]. The power rate of an active VM is 50 W and that of an idle VM is 30 W. In the following section, the applied simulation parameters are in Table 2 unless otherwise specified.

In the experiments, we consider a network as Figure 3, as shown in Figure 3. There are 11 nodes in the system. We evaluate the performance of energy consumption of VM immigration (ECVMI), energy lost (EL: for offloading tasks and green energy transferring), total energy consumption (TEC), and brown energy (BE) in different cases: (1) different arrival rate of tasks; (2) different green energy generation rates; (3) different energy attenuation ratio (range), and (4) different number of VMs. The most important metric is BE (brown energy consumption). BE is the standard to assess the performance of those methods. Other metrics explain the reasons for the various requirements of BE. We will compare the proposed methods with

TABLE 2. Parameters used in the simulation.

Parameter	Values
Task processing energy consumption	Rand(3, 30)
VM immigration energy consumption without energy transferring attenuation	Rand(3, 10)
VM immigration time	[1~5]
Task offloading energy consumption without energy transferring attenuation	Rand(1, 10)
Green energy generation rate every time slot	[0, 5]
Power rate of active VM	50W
Power rate of idle VM	30W
Task arrival rate	15
Energy transferring attenuation ratio	[0, 0.25]
Number of VMs	3

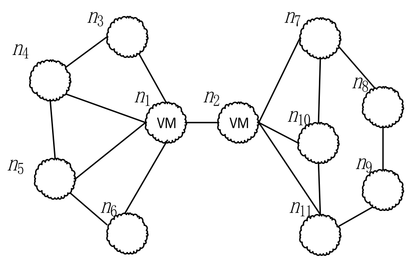


FIGURE 3. The test topology.

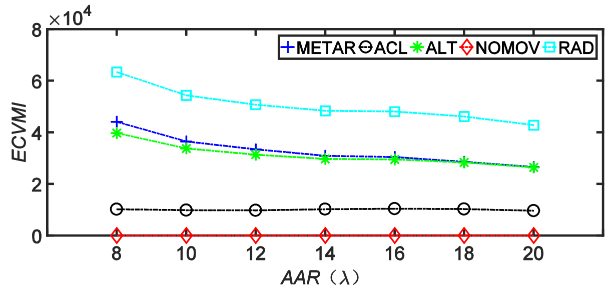


FIGURE 4. *ECVMI* with different *AARs*.

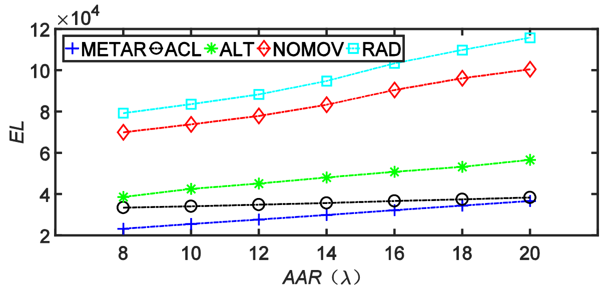


FIGURE 5. *EL* with different *AARs*.

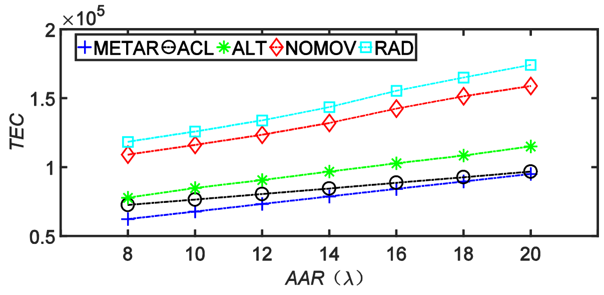


FIGURE 6. *TEC* with different *AARs*.

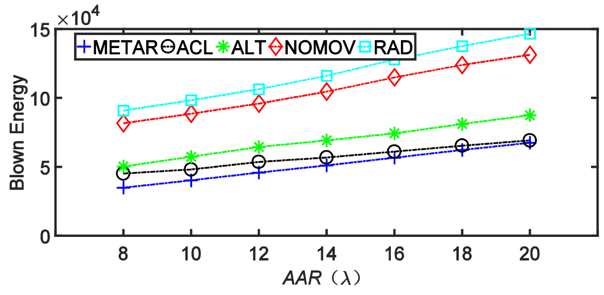


FIGURE 7. Brown energy with different *AARs*.

RAD and NOMOV. RAD randomly selects a note as the VM immigration target. NOMOV does not immigrate VM in the scheduling.

B. EXPERIMENTS

1) EFFECT OF THE ARRIVAL RATE

In this section, we evaluate the methods with different AARs. We attempt to determine the influence of AAR on different metrics of various scheduling methods. Figures 4∼7 show the performance of different methods in *ECVMI*, *EL*, *TEC,* and *BE*, respectively. The *AAR* is changed from 8 to 20 with a step of 2.

Generally, all methods exhibit an increasing trend with an increase in AAR in *EL*, *TEC,* and brown energy and a decreasing trend in *ECVMI*. The value of RAD always the largest, followed by that of NOMOV, ACL, ALT, and METAR. When the system has a higher AAR, it may offload more tasks to VMs, which increases the *EL* (Figure 5), and finally increases the *TEC* (Figure 6) and *BE* (Figure 7). NOMOV and RAD always have the largest values in *EL*, *TEC,* and *BG*. RAD also has the largest value in *ECVMI*. In METAR, ACL, and ALT, METAR performs best because it always has the smallest value in *BE* (Figure 7). We also discover that all methods (except ACL) exhibit a slight decrease with the increase in the ARRs because additional tasks easily identify the route for offloading tasks and immigrating VMs with a lower ETAR. ACL always prefers the route with minimum ETAR, thus, it is not substantially different for various AARs. Compared with *BE* of ACL, ALT, NOMOV and RAD, METAR is, on average reduced by 0.16 (e+05), 0.46 (e+05), 1.38 (e+05) and 1.67 (e+05), which is approximately 10.94%, 26.30%, 51.56% and 56.41 %, respectively.

2) EFFECT OF THE GREEN ENERGY GENERATION RATE

Figures 8∼11 show the performance of different methods in *ECVMI*, *EL*, *TEC,* and *BE* with different *GEGR*s. The *GEGR* is changed from [0, 3] to [0, 7].

For *EL*, *TEC,* and *BG*, the order of the scheduling methods from best to worst is METAR, ACL, and ALT. *BE* is the most important metric. To *BE* of RAD, METAR, ACL, ALT and NOMOV are reduced, by 6.91 (e+04), 6.26(e+04), 5.14(e+04) and 1.89 (e+04) in *BE* on average. With the increase of *GEAR*s, all methods get more green energy, thus reducing *TEC*, *BE*, and enhancing *EL*.

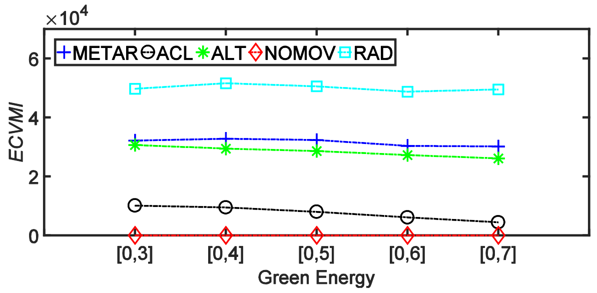


FIGURE 8. ***ECVMI* for with different *GEGRs*.**

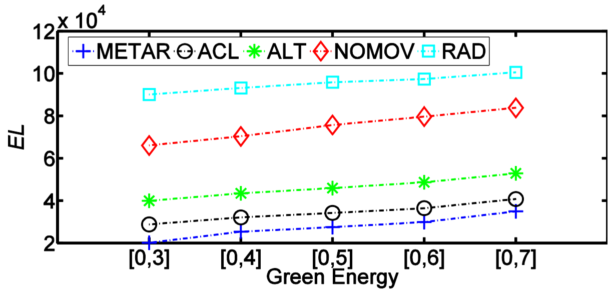


FIGURE 9. ***EL* for with different *GEGRs*.**

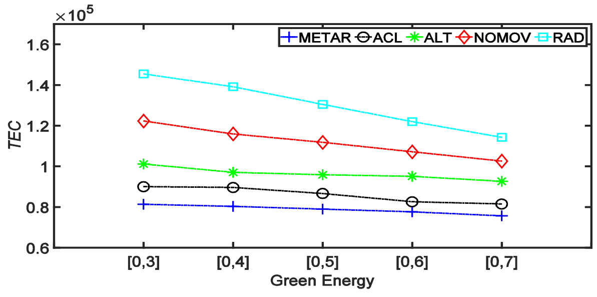


FIGURE 10. ***TEC* for with different *GEGRs*.**

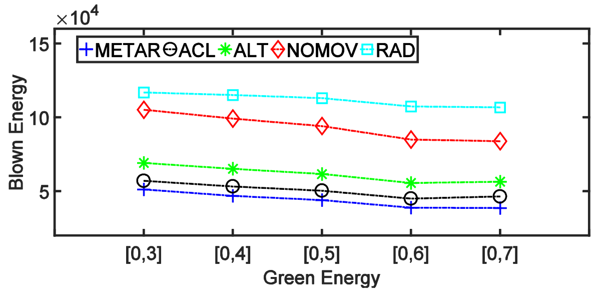


FIGURE 11. **Brown energy with different *GEGRs*.**

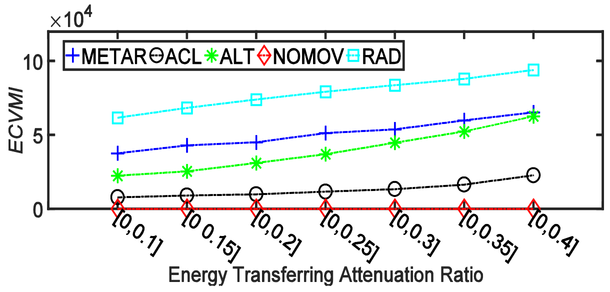


FIGURE 12. ***ECVMI* with different *ETARs*.**

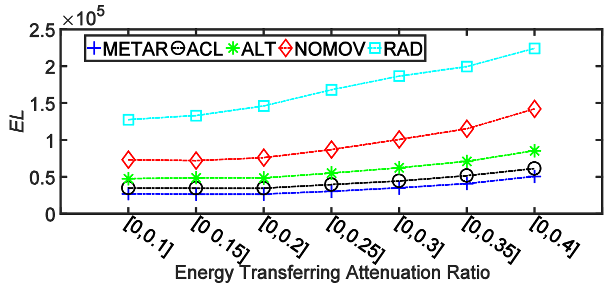


FIGURE 13. ***EL* with different *ETARs*.**

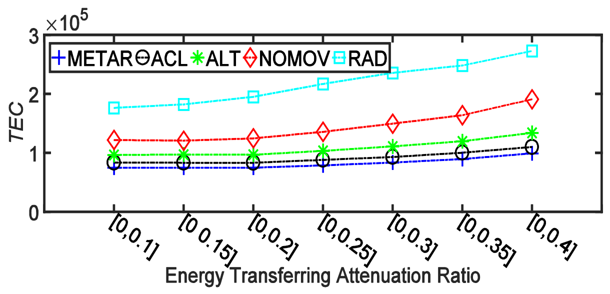


FIGURE 14. ***TEC* with different *ETARs*.**

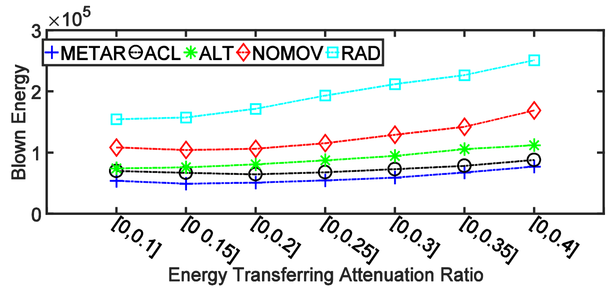


FIGURE 15. **Brown energy with different *ETARs*.**

3) EFFECT OF ENERGY TRANSFERRING ATTENUATION RATIO

Figures 12∼15 are used to investigate the performance of different methods with various metrics under various *ETAR*s. The *ETAR* is changed in the scope of [0, 0.1], [0, 0.15], [0, 0.2], [0, 0.25], [0, 0.3], [0, 0.35] and [0, 0.4], respectively.

Generally, all methods improve these four indicators by enhancing *ETAR*, as higher *ETAR* results in higher energy loss during green energy transfer and VMs migration, thus improving *EL* and ultimately improving the other three indicators. The decreasing order of *ECVMI*, *EL,* and *TEC* is METAR, ACL, ALT, NOMOV, and RAD, respectively. Compared with *BE* of ACL, ALT, RAD and NOMOV METAR average reduces by 0.51(e+04), 2.51(e+04), 3.34(e+04), 9.90(e+04) on average.

4) EFFECT OF NUMBERS OF VMS

Figs 16∼19 are used to evaluate the performance of four methods in *ECVMI*, *EL*, *TEC,* and BE when the number of VMs is 2, 3, and 4, respectively. Compared with BE of NOMOV, ECL, and ALT, the METAR is reduced by 0.98

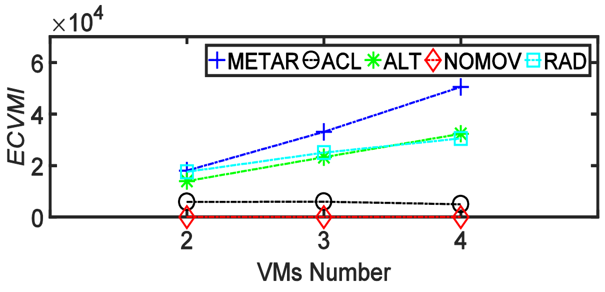


FIGURE 16. ***ECVMI*** **with different numbers of VMs.**

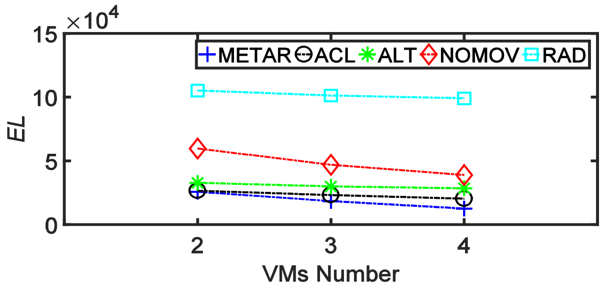


FIGURE 17. ***EL*** **with different numbers of VMs.**

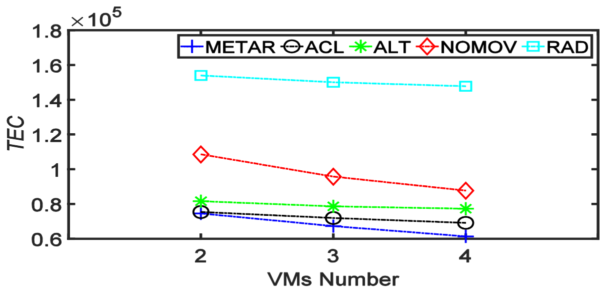


FIGURE 18. ***TEC*** **with different numbers of VMs.**

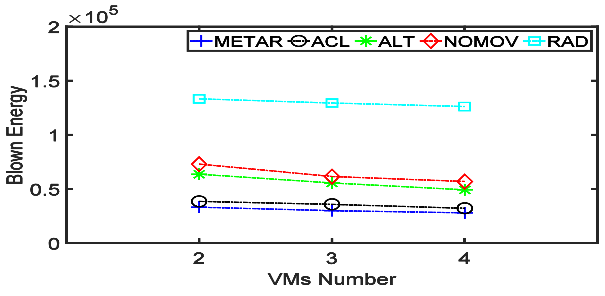


FIGURE 19. **Brown energy with different numbers of VMs.**

(e+05), 0.02 (e+05) and 0.31 (e+05) on average, respectively. METAR performs best because it not only reduces the energy consumption for immigrating VMs, but also tries to reduce the energy consumption of task offloading and green energy transfer.

C. COMPARISON BETWEEN THE PROPOSED METHODS AND OTHER METHODS

From Section VI(C), we find that METAR always performs the best in all metrics under various values of AARs, GEARs,

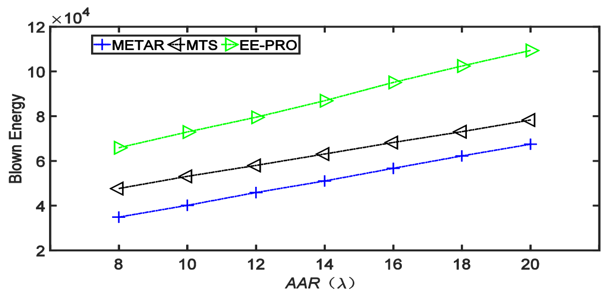


FIGURE 20. **Brown energy with different** ***AARs.***

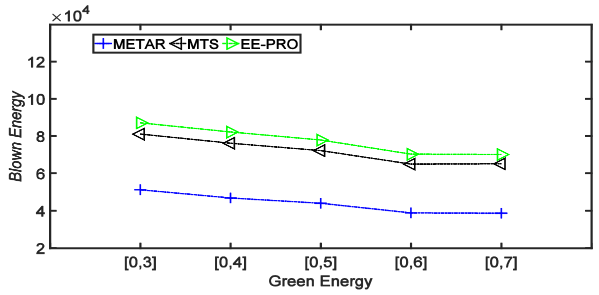


FIGURE 21. **Brown energy with different** ***GEARs.***

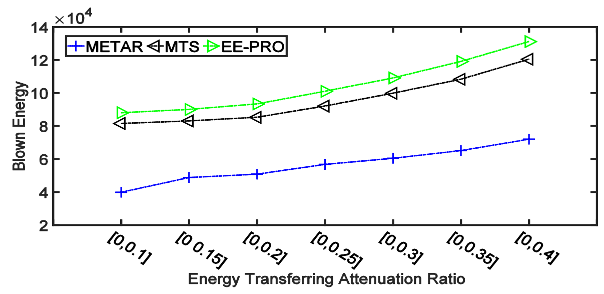


FIGURE 22. **Brown energy with different** ***ETARs.***

ETARs, and the number of VMs. Brown energy is the most important value in all those metrics. Therefore, we will compare the value of METAR with the most advanced methods in BE. MTS [13] is an approximate optical heuristics algorithm that considers VM immigration, task scheduling, and energy scheduling. But, MTS assumes that there is only one VM, and we extend it to multiple VMs. We also compare EE-PRO [36] with our methods. We suppose that the processing ability of a VM is [0.8, 1.2] standard machines (standard machine: with 8 Cores, 8G memory, 1T hard disk). The execution time of a task on a standard machine is about [1, 100] time unit (seconds).

Figs 20∼23 give the brown energy of METAR, MTS and EE-PRO under different parameters: AARs. GEARs, ETARs, and the number of VMs. Generally speaking, METAR performs best under all environments, followed by MTS and EE-PRO, no matter what kinds of parameters are changed. Compared to MTS and EE-PRO, METAR average reduces by 28.23% and 49.50% in brown energy consumption.

Figures 24∼25 give the average waiting time (AWT) and average execution time (AET) when the AAR is changed

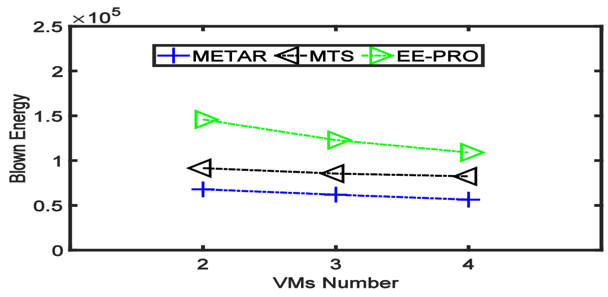


FIGURE 23. Brown energy with different *VMs*.

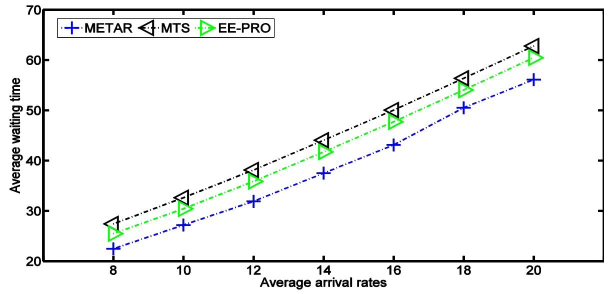


FIGURE 24. Average waiting time.

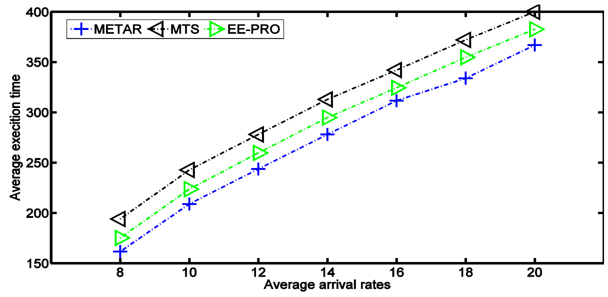


FIGURE 25. Average execution time.

from 8 to 20 with a step of 2 (for other parameters, the values of three methods also have the same trends). In the three scheduling methods, METAR always has the minimum value in AET and AWT, followed by EE-PRO and MTS. Compare to the AET of MTS and EE-PRO, METAR average reduces by 5.67% and 11.52%. Compare to the AWT of MTS and EE-PRO, METAR average reduces by 9.92% and 11.31%.

D. DISCUSSIONS

From the above simulations, AAR, *GEGR*, and the number of VMs affect the performance of those methods. All methods have an increasing trend with the increase of the AAR in *EL*, *TEC* and brown energy, and METAR always performs the best regardless of AARs. All methods have a stable performance in *ECVMI*, and show a slight increase in *EL*, *TEC,* and *BE* with an enhancement in *GEGR*. The reason is these the five methods need to transfer green energy and lose more energy (Figure 9), thus improving *TEC* and *BE*. We also discover that NOMOV performs better than RAD, because randomly offloading tasks and immigrating VMs wastes more green energy and requires more VMs immigration energy. Other methods intelligently select the green energy transfer route to keep metrics stable in the scheduling. From the view of *GEGR*, METAR performs the best in four metrics. All methods increase in *EL*, *TEC,* and *BE* and decrease in *ECVMI* with the increase in the number of VMs, because a larger number of VMs, as more VMS always ensure the likelihood of each method to select a transfer route (green energy) and migration route (for VMs) with lower ETAR, thus reducing EL.

We also compare METAR with MTS and EE-PRO. We also found that METAR consumes the least brown energy under any condition that we have tested. METAR performs better because it also focuses on improving the utilization rate of green energy and selecting the route with smaller *ETAR*. We select the route with the smallest value in *ETAR* for transferring green energy, immigrating VMs, and offloading tasks. After getting the VM immigration target, the energy transferring routes, and the task offloading route, we can search the possible in a small scope to get the relatively good scheduling by a heuristic algorithm (Algorithms 1 and 2).

VII. CONCLUSION AND FUTURE WORK

In this paper, we discuss the problem encountered in a mobile environment with EI technology. We propose three methods to schedule tasks and immigrate VMs: METAR, ACL and ALT. Simulations are used to evaluate the methods, by considering four parameters and four metrics. The four parameters are (1) different task arrival rates; (2) different green energy generation rates; (3) different energy attenuation ratios (ranges), and (4) different number of VMs. We give comparisons on four metrics: *ECVMI*, *EL*, *TEC* and *BE*. The simulation results show that METAR performs better than other methods in reducing brown energy. So, when we transmit energy and immigrate VMs, we prefer to select the route with the smallest ETAR would reduce the total energy consumption and keep other metrics (such as average waiting time, average execution time) in a good performance. Furthermore, we compare METAR with MTS and EE-PRO, the simulation results show that METAR has a good performance in BE, AWT and AET.

If we can use some meteorological models [37] to forecast the green energy, it may help the scheduling of edge devices. The consolidation technique [38] also can be used to reduce the energy consumption under our environment that would be a new research of our paper. Edge computing has been widely applied in many fields, such as meteorological stations, environment monitoring devices, and medical healthcare devices. We also hope to evaluate the methods and conduct more research in specific areas.

REFERENCES

[1] J. Wang, P. Zhao, S. C. H. Hoi, and R. Jin, "Online feature selection and its applications," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 3, pp. 698–710, Mar. 2014.
[2] B. Zhou, A. V. Dastjerdi, and R. N. Calheiros, "An online algorithm for task offloading in heterogeneous mobile clouds," *ACM Trans. Internet Technol.*, vol. 18, no. 2, p. 23, 2018.
[3] L. Yang, H. Zhang, M. Li, J. Guo, and H. Ji, "Mobile edge computing empowered energy efficient task offloading in 5G," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6398–6409, Jul. 2018.

[4] A. Kiani and N. Ansari, "Edge computing aware NOMA for 5G networks," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1299–1306, Apr. 2018.

[5] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Gener. Comput. Syst.*, vol. 78, pp. 680–698, Jan. 2018.

[6] M. Dias de Assuncao, A. da Silva Veith, and R. Buyya, "Distributed data stream processing and edge computing: A survey on resource elasticity and future directions," *J. Netw. Comput. Appl.*, vol. 103, pp. 1–17, Feb. 2018.

[7] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan. 2018.

[8] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the Internet of Things: A case study," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1275–1284, Apr. 2018.

[9] R. Morabito, V. Cozzolino, A. Y. Ding, N. Beijar, and J. Ott, "Consolidate IoT edge computing with lightweight virtualization," *IEEE Netw.*, vol. 32, no. 1, pp. 102–111, Jan./Feb. 2018.

[10] M. Chen, W. Li, Y. Hao, Y. Qian, and I. Humar, "Edge cognitive computing based smart healthcare system," *Future Gener. Comput. Syst.*, vol. 86, pp. 403–411, Sep. 2018.

[11] K. Wang, H. Yin, W. Quan, and G. Min, "Enabling collaborative edge computing for software defined vehicular networks," *IEEE Netw.*, vol. 32, no. 5, pp. 112–117, Sep./Oct. 2018.

[12] Y. Hao, J. Cao, Q. Wang, and J. Du, "Energy-aware scheduling in edge computing with a clustering method," *Future Gener. Comput. Syst.*, vol. 117, pp. 259–272, Apr. 2021, doi: 10.1016/j.future.2020.11.029.

[13] L. Gu, J. Cai, D. Zeng, Y. Zhang, H. Jin, and W. Dai, "Energy efficient task allocation and energy scheduling in green energy powered edge computing," *Future Gener. Comput. Syst.*, vol. 95, pp. 89–99, Jun. 2019.

[14] C. Zou, "Energy Internet technology," in *New Energy*. Singapore: Springer, 2020, doi: 10.1007/978-981-15-2728-9_5.

[15] W. Su, A. Q. Huang, *The Energy Internet*. London, U.K.: Woodhead Publishing, 2019, doi: 10.1016/B978-0-08-102207-8.12001-6.

[16] Y. Mao, J. Zhang, Z. Chen, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[17] D. G. Roy, D. De, A. Mukherjee, and R. Buyya, "Application-aware cloudlet selection for computation offloading in multi-cloudlet environment," *J. Supercomput.*, vol. 73, no. 4, pp. 1672–1690, 2017.

[18] J. Zhang, X. Hu, Z. Ning, and E. C. H. Ngai, "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2633–2645, Aug. 2018.

[19] L. Cui, C. Xu, S. Yang, J. Z. Huang, J. Li, X. Wang, Z. Ming, and N. Lu, "Joint optimization of energy consumption and latency in mobile edge computing for Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4791–4803, Jun. 2019.

[20] H. Sun, F. Zhou, and R. Q. Hu, "Joint offloading and computation energy efficiency maximization in a mobile edge computing system," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 3052–3056, Mar. 2019.

[21] Q. Wang, S. Guo, J. Liu, and Y. Yang, "Energy-efficient computation offloading and resource allocation for delay-sensitive mobile edge computing," *Sustain. Comput., Informat. Syst.*, vol. 21, pp. 154–164, Mar. 2019.

[22] M. T. Kabir and C. Masouros, "A Scalable energy vs. latency trade-off in full-duplex mobile edge computing systems," *IEEE Trans. Commun.*, vol. 67, no. 8, pp. 5848–5861, Aug. 2019.

[23] J. Zheng, L. Gao, H. Wang, X. Li, P. Xu, and L. Wang, "Joint downlink and uplink edge computing offloading in ultra-dense HetNets," *Mobile Netw. Appl.*, vol. 24, pp. 1452–1460, May 2019.

[24] T. Bahreini, H. Badri, and D. Grosu, "Energy-aware capacity provisioning and resource allocation in edge computing systems," in *Proc. Conf. Edge Comput.* Cham, Switzerland: Springer, 2019, pp. 31–45.

[25] S. Yang, F. Li, M. Shen, X. Chen, and X. Fu, "Cloudlet placement and task allocation in mobile edge computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5853–5863, Jun. 2019.

[26] L. Liu, X. Chen, Z. Lu, and L. Wang, "Mobile-edge computing framework with data compression for wireless network in energy Internet," *Tsinghua Sci. Technol.*, vol. 24, no. 3, pp. 271–280, Jun. 2019.

[27] X. Zhao, K. Yang, Q. Chen, D. Peng, H. Jiang, X. Xu, and X. Shuang, "Deep learning based mobile data offloading in mobile edge computing systems," *Future Gener. Comput. Syst.*, vol. 99, pp. 346–355, Oct. 2019.

[28] M. Zakarya, L. Gillam, H. Ali, I. Rahman, K. Salah, R. Khan, O. Rana, and R. Buyya, "EpcAware: A game-based, energy, performance and cost efficient resource management technique for multi-access edge computing," *IEEE Trans. Services Comput.*, early access, Jun. 26, 2020, doi: 10.1109/TSC.2020.3005347.

[29] B. Ali, M. A. Pasha, S. U. Islam, H. Song, and R. Buyya, "A volunteer supported fog computing environment for delay-sensitive IoT applications," *IEEE Internet Things J.*, early access, Sep. 21, 2020, doi: 10.1109/JIOT.2020.3024823.

[30] B. Li, Z. Fei, J. Shen, X. Jiang, and X. Zhong, "Dynamic offloading for energy harvesting mobile edge computing: Architecture, case studies, and future directions," *IEEE Access*, vol. 7, pp. 79877–79886, 2019.

[31] G. Zhang, W. Zhang, Y. Cao, D. Li, and L. Wang, "Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4642–4655, Oct. 2018.

[32] E. W. Djikstra, "A note on two problems in connection with graphs," *Numer. Math.*, vol. 1, pp. 269–271, 1959.

[33] X. Xu, Y. Li, T. Huang, Y. Xue, K. Peng, L. Qi, and W. Dou, "An energy-aware computation offloading method for smart edge computing in wireless metropolitan area networks," *J. Netw. Comput. Appl.*, vol. 133, pp. 75–85, May 2019.

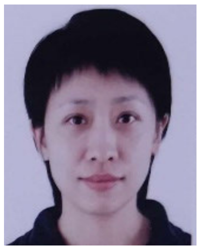
[34] L. Chen, X. Li, H. Ji, and V. C. M. Leung, "Computation offloading balance in small cell networks with mobile edge computing," *Wireless Netw.*, vol. 25, pp. 4133–4145, May 2018.

[35] X. Xu, Q. Liu, Y. Luo, K. Peng, X. Zhang, S. Meng, and L. Qi, "A computation offloading method over big data for IoT-enabled cloud-edge computing," *Future Gener. Comput. Syst* , vol. 95, pp. 522–533, Jun. 2019.

[36] H. Feng, S. Guo, A. Zhu, Q. Wang, and D. Liu, "Energy-efficient user selection and resource allocation in mobile edge computing," *Ad Hoc Netw.*, vol. 107, Oct. 2020, Art. no. 102202.

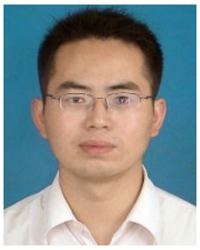
[37] Y. Hao, J. Cao, T. Ma, and S. Ji, "Adaptive energy-aware scheduling method in a meteorological cloud," *Future Gener. Comput. Syst.*, vol. 101, pp. 1142–1157, Dec. 2019.

[38] A. A. Khan, M. Zakarya, R. Buyya, R. Khan, M. Khan, and O. Rana, "An energy and performance aware consolidation technique for containerized datacenters," *IEEE Trans. Cloud Comput.*, early access, Jun. 5, 2019, doi: 10.1109/TCC.2019.2920914.



QING ZHANG received the M.E. degree from the Wuhan University of Science and Technology, in 2001. She is currently an Instructor with the Computer Engineering College, Jimei University, and an Associate Editor of blue book *Annual Report on Future Media in China* (2018). She has presided over the industry-university cooperative education project of the Ministry of Education "Construction of Interactive Design Courses for Software Engineering Majors," participated in a number of major and key projects in the Fujian Province Social Sciences, Nation Radio, and Television Administration. Her main research interests include data science, mobile computing, and human–computer interaction.

XIAOYONG LIN is currently a Professor with the Xiamen University of Technology, the Director of the Future Media Development Research Center, Fujian University of Arts and Social Sciences Research Base, and an Editor of blue book *Annual Report on Future Media in China*. He has presided over a number of major and key projects in Fujian Province Social Sciences, Nation Radio, and Television Administration. He has published three monographs and more than 30 academic articles. His main research interests include future media, Internet, and new media.



YONGSHENG HAO received the M.S. degree in engineering from Qingdao University, in 2008. He is currently a Senior Engineer with the Network Center, Nanjing University of Information Science and Technology. He has published more than 30 papers in international conferences and journals. His current research interests include distributed and parallel computing, mobile computing, grid computing, web service, particle swarm optimization algorithm, and genetic algorithm.

JIE CAO received the Ph.D. degree in computer science from Northeastern University, in 2001. He is currently a Professor with the Xuzhou University of Technology. His current research interests include distributed and parallel computing and decision systems.