BeCome: Blockchain-Enabled Computation Offloading for IoT in Mobile Edge Computing

Xiaolong Xu, Xuyun Zhang, Honghao Gao, Yuan Xue, Lianyong Qi*, Wanchun Dou

Abstract—Benefiting from the real-time processing ability of edge computing, computing tasks requested by smart devices in the IoT are offloaded to edge computing devices (ECDs) for implementation. However, ECDs are often overloaded or underloaded with disproportionate resource requests. In addition, during the process of task offloading, the transmitted information is vulnerable, which can result in data incompleteness. In view of this challenge, a blockchain-enabled computation offloading method, named BeCome, is proposed in this paper. Blockchain technology is employed in edge computing to ensure data integrity. Then, the nondominated sorting genetic algorithm III (NSGA-III) is adopted to generate strategies for balanced resource allocation. Furthermore, simple additive weighting (SAW) and multicriteria decision-making (MCDM) are utilized to identify the optimal offloading strategy. Finally, performance evaluations of BeCome are given through simulation experiments.

Index Terms—Blockchain, Edge Computing, Computation Offloading, IoT.

1 Introduction

In recent years, the traditional notion of the Internet as a network infrastructure covering terminals had faded gradually, while the Internet of Things (IoT) has emerged as an expansion paradigm of the Internet, accommodating objects that exist or could exist in the future [1] [2]. In the IoT, billions of physical objects are equipped with smart devices that gather information from surrounding conditions [3]. However, the speed of information processing is restricted due to the limited computation capacities of smart devices [4]. Mobile edge computing (MEC), which enables computing tasks to be performed at the edge of the network, has emerged as a complement to cloud computing [5]. Driven by the MEC technique, the computing tasks on smart devices are offloaded to edge computing devices (ECDs), which are in closer proximity to smart devices than the cloud platform. As there is a server to assist data processing in each ECD via offloading the workloads to ECDs, the data transmission time is sharply decreased, and the QoE (quality of experience) for users in the IoT is greatly improved.

- *X. Xu is with the School of Computer and Software, Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science and Technology, Nanjing, China. Email: njuxlxu@gmail.com*
- *X. Zhang is with the Department of Electrical and Computer Engineering, University of Auckland, New Zealand. E-mail: xuyun.zhang@auckland.ac.nz*
- *H. Gao is with theComputing Center, Shanghai University, Shanghai 200444, China. E-mail: gaohonghao@shu.edu.cn*
- *Y. Xue is with the School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, China. Email: xueyuannuist@gmail.com*
- *L. Qi is with the School of Information Science and Engineering, Qufu Normal University, China. E-mail: lianyongqi@gmail.com (Corresponding Author)*
- *W. Dou and X. Xu are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China. E-mail: douwc@nju.edu.cn*

Manuscript received ; revised

To achieve efficient offloading in the IoT, ECDs and the cloud platform are connected to a resource manager that monitors real-time load conditions globally. If an ECD is underloaded, which means that few tasks are executed on it, large quantities of energy are wasted to support the ECD's operation [6]. It is then necessary to offload the tasks on the original ECD to other ECDs with idle resources and set the original ECD into sleeping mode [7]. Based on the monitoring information and received offloading requests from smart devices, the resource manager selects a destination ECD to accommodate the offloaded tasks. On the other hand, if an ECD is overloaded, requests from smart devices are rejected and queued in the ECD until previous tasks are finished and the needed resources are available again. Therefore, it is suitable to offload the queuing computing tasks to the outsourced cloud while the offloading time is non-negligible and uncertain [8]. In this situation, the resource manager compares the queuing time with the offloading time and formulates a strategy due to the shorter time-consuming data processing method between queuing and offloading.

Nevertheless, computation offloading in MEC has drawbacks in terms of data security, including the security of transmitted data and data privacy protection for users [9] [10]. As the amount of data continues to increase, data integrity and security should be ensured to prevent users' privacy from being breached, with unpredictable consequences. Traditional data protection solutions are based on a centralized approach, such as using a trusted central entity [9] [11]. However, there is a single point of failure problem with the trusted central entity, making traditional solutions unsuitable in edge computing. Should the central entity that manages the integrity-preserving system be attacked, the security of the whole system in edge computing is under threat. Blockchain, which is a decentralized public ledger that stores transaction records, works as an appropriate solution. The blockchain records transactions as blocks that form a linked list data structure. Each newly generated block gets a hash value that joins the previous block to the current block, forming an irreversible chain [12]. This operation is broadcast to all miners, and when most miners return to the agreement, the newly created block is successfully added [13]. Considering that all users can verify the authenticity of the transaction, the attack is nearly impossible to achieve unless the attacker controls more than half of the clients, a feature that defines the decentralized nature of blockchain. Therefore, blockchain technology plays a big role in protecting data integrity in MEC.

Apart from data integrity, energy consumption, time consumption and load balance are also essential criteria to judge whether the offloading strategy is appropriate. However, it is a challenge to achieve trade-offs among minimizing ECDs' task offloading time, optimizing their energy consumption and maintaining load balance while considering data integrity. In this paper, a blockchain-enabled computation offloading method, named BeCome, is devised for the IoT, and the main contributions include the following:

- Devise a blockchain-enabled edge computing framework for the IoT to guarantee data integrity during the task offloading process.
- Adopt the nondominated sorting genetic algorithm III (NSGA-III) [14] to generate feasible resource allocation schemes through computation offloading using blockchain technology in mobile edge computing.
- Employ the simple additive weighting (SAW) and the multicriteria decision-making (MCDM) techniques to confirm the optimal schemes based on Virtual Machine migration technology.
- Conduct extensive experiments to test the validity of the proposed computation offloading method in a specific edge computing environment.

The remainder of this paper is organized as follows. In Section 2, a blockchain-enabled edge computing framework is designed. In Section 3, formalized definitions are introduced and complete mathematical modeling is described. In Section 4, a blockchain-enabled offloading method is proposed for the IoT in MEC. In Section 5, simulation experiments and comparison experiments are demonstrated. In Section 6, related work is summarized. In Section 7, the conclusion and future work are outlined.

2 Block Generation for Offloading in Edge Computing

Due to limited space in this paper, we have provided this section in Appendix A of a supplementary file.

3 System Model

3.1 Resource Model

Suppose there are Z types of ECD, denoted as $EC = \{ec_1, ec_2, \ldots, ec_Z\}$, and $\mathrm{ec}_z(z = \{1, 2, \cdots, Z\})$ has S data transmitters, denoted as $DT_z = \{dt_{z,1}, dt_{z,2}, \ldots, dt_{z,S}\}$. The amount of z-th type of ECD is set to N, which is denoted as $C_z = \{c_{z,1}, c_{z,2}, \ldots, c_{z,N}\}$. Suppose that there are M smart devices in the IoT. With loss of generality, suppose that a smart device accommodates only one task. Thus, the computing task from the m-th $(m = \{1, 2, \ldots, M\})$ smart device is denoted as k_m. Generally, the VM technology is powerful for ECD resource management in the IoT, and the physical resource is provisioned in the form of VM instances. Then the capacity of the n-th ECD $(n = \{1, 2, \ldots, N\})$ in ec_z is measured by the number of VM instances, denoted as $va_{z,n}$.

Definition 1 (Resource requirements of k_m) The resource requirements of k_m are defined by a 4-tuple, denoted as $r_m = (c_z, nv_m, st_m, ft_m)$, where c_z, nv_m, st_m and ft_m are the type of ECD employed by k_m, the number of VMs required for k_m, the occupied start time of k_m, and the finish time of k_m.

3.2 Time Consumption Model

To accurately predict time consumption of the IoT in MEC, we propose a time consumption model that is divided into two parts. Part one is the offloading time model, while part two is the task-queuing time model. With the decrease in offloading time, the QoE for smart device users is improved. Thus, the offloading time from smart devices to ECDs should be considered.

Assume M_m is a binary variable adopted to judge whether the computing task is migrated to the cloud platform.

$$M_m(t) = \begin{cases} 1, \text{if } k_m \text{ is migrated to ECD}, \\ 0, \text{Otherwise}. \end{cases} \tag{1}$$

The offloading time between the first data transmitter and destination ECD depends on the number of data transmitters that k_m goes through. The time consumption of offloading, k_m, is calculated by

$$T_m^{off}(t) = M_m(t) \cdot \frac{D_m}{\alpha_z} \cdot ZD_m^z, \tag{2}$$

where D_m is the data size of k_m and α_z is the data transmission rate between two data transmitters in ec_z. In addition, ZD_m^z represents the shortest path between the original data transmitter and the destination data transmitter in ec_z.

When k_m is offloaded to the cloud, the offloading time is calculated by

$$T_m^{off'}(t) = (1 - M_m(t)) \cdot \frac{D_m}{\omega}, \tag{3}$$

where ω is the data transmission rate between the cloud platform and the smart mobile device.

The total offloading time of all tasks is calculated by

$$T_{off}(t) = \sum_{m=1}^{M} T_m^{off}(t) + \sum_{m=1}^{M} T_m^{off'}(t). \tag{4}$$

When several tasks simultaneously access the same resource of ECD, the ECD appears occupied. Thus, the queuing time for accessing the resources also needs consideration. It is possible to queue the tasks from smart devices for processing [15]. The possibility that the ECD is in idle status is recorded as

$$P_0 = [\sum_{m=0}^{va_{z,n}-1} \frac{R_{z,n}^m}{m_z!} + \frac{R_{z,n}^{va_{z,n}}}{va_{z,n}!(1 - \frac{R_{z,n}}{va_{z,n}})}]^{-1}, \tag{5}$$

where $R_{z,n}$ indicates the utilization of $c_{z,n}$ and m_z is the number of VMs needed for task processing in ec_z.

Then, the possibility that there is a queue of length m_z is calculated by

$$P_n = \begin{cases} \frac{R_{z,n}^{m_z}}{m_z!} \cdot P_0, n < va_{z,n}, \\ \frac{R_{z,n}^{m_z}}{va_{z,n}! va_{z,n}{}^{m_z - va_{z,n}}} \cdot P_0, n \geq va_{z,n}. \end{cases} \quad (6)$$

When $n \geq va_{z,n}$, the possibility of k_m queuing in the ECD is calculated by

$$O_m(va_{z,n}, R_{z,n}) = \sum_{n=va_{z,n}}^{\infty} P_n. \quad (7)$$

The average waiting queue length L_q is calculated by

$$L_q = \frac{P_0 \cdot R_{z,n}^{m_z}}{va_{z,n}!} \sum_{n=va_{z,n}}^{\infty} (n - va_{z,n}) \cdot R_{va_{z,n}}{}^{n - va_{z,n}}, \quad (8)$$

where $R_{va_{z,n}} = \frac{R_{z,n}}{va_{z,n}}$.

In fact, the arrival time of the task is regular and has a peak time, which is consistent with the law of negative exponential distribution. Thus, the arrival time as well as the corresponding service time of the task obeys the negative exponential distribution, and the average queuing time for tasks in the ECD is calculated by

$$T_m^{queue}(t) = \frac{1}{va_{z,n} \cdot \theta - \zeta} \cdot O_m(va_{z,n}, R_{z,n}). \quad (9)$$

where the interval of the task arrival time obeys the negative exponential distribution of parameter ζ and the service time in the ECD is subject to the negative exponential distribution of parameter θ.

Then the total time consumption is calculated by

$$T(t) = T_{off}(t) + \sum_{m=1}^{M} T_m^{queue}(t). \quad (10)$$

3.3 Energy Consumption Model

The data transmitter energy consumption required for k_m is calculated by

$$E_m^{DT}(t) = e_z^{DT} \cdot SL_m + \frac{D_m}{\delta_z} \cdot \beta_z \cdot SL_m, \quad (11)$$

where e_m^{DT} represents the baseline energy consumption of the data transmitter serving ec_z and D_m represents the data size of k_m. In addition, δ_z is the data transmission rate between data transmitters serving ec_z, β_z is the energy consumption rate of data transmitter serving ec_z, and SL_m is the number of data transmitters that k_m goes through.

Thus, the entire energy consumption of data transmitters is calculated by

$$E_{DT}(t) = \sum_{m=1}^{M} E_m^{DT}(t) \cdot M_m(t). \quad (12)$$

For the sake of judging whether k_m is migrated to $c_{z,n}$, let G_m be a decision binary variable.

$$G_m(t) = \begin{cases} 1, \text{if } k_m \text{ is migrated to } c_{z,n}, \\ 0, \text{Otherwise}. \end{cases} \quad (13)$$

The energy consumption of active VMs in $c_{z,n}$ is calculated by

$$E_{z,n}^{active}(t) = \sum_{i=1}^{va_{z,n}} \varepsilon_{z,n,i} \cdot \phi_{z,n,i}, \quad (14)$$

where $\varepsilon_{z,n,i}$ is the power of $v_{z,n,i}$ in active mode and $\phi_{z,n,i}$ represents the running time of $v_{z,n,i}$.

The total energy consumption of active VMs is calculated by

$$E_{total}^{active}(t) = \sum_{z=1}^{Z} \sum_{n=1}^{N} E_{z,n}^{active}(t). \quad (15)$$

According to this fact, the VMs without task processing have a baseline energy consumption. Then, the time consumption of VMs in idle mode is determined by the maximum operating cycle time of $v_{z,n,i}$. Thus, the maximum operating time in $c_{z,n}$ is calculated by

$$H_{z,n}(t) = \max_{i=1}^{va_{z,n}} \{\phi_{z,n,i}\}. \quad (16)$$

Therefore, the energy consumption of idle VMs in $c_{z,n}$ is calculated by

$$E_{z,n}^{idle}(t) = \sum_{i=1}^{va_{z,n}} \varphi_{z,n,i} \cdot (H(t) - \phi_{z,n,i}), \quad (17)$$

where $\varphi_{z,n,i}$ is the power rate of $v_{z,n,i}$ in idle mode.

The total energy consumption of all idle VMs is calculated by

$$E_{total}^{idle}(t) = \sum_{z=1}^{Z} \sum_{n=1}^{N} E_{z,n}^{idle}(t). \quad (18)$$

Moreover, active ECDs consume baseline energy that is calculated by

$$E_{ECD}(t) = \sum_{z=1}^{Z} \sum_{n=1}^{N} \gamma_{z,n} \cdot H_{z,n}(t), \quad (19)$$

where $\gamma_{z,n}$ is the baseline power rate of $c_{z,n}$.

Based on the analysis above, total energy consumption for IoT in mobile edge computing is calculated by

$$E(t) = E_{DT}(t) + E_{total}^{active}(t) + E_{total}^{idle}(t) + E_{ECD}(t). \quad (20)$$

3.4 Load Balance Analysis

Let $\eta_{z,n,i}$ be a binary variable to judge whether k_m is occupied v_i in the $c_{z,n}$.

The occupy condition of VMs in ECDs reflects the resource utilization of ECDs. Therefore, the resource utilization of $c_{z,n}$ is calculated by

$$R_{z,n}(t) = \frac{1}{va_{z,n}} \cdot \sum_{m=1}^{M} \sum_{i=1}^{va_{z,n}} \eta_{z,n,i}(t). \quad (21)$$

The occupation status of $c_{z,n}$ is measured by

$$\lambda_{z,n}(t) = \begin{cases} 1, \text{if } c_{z,n} \text{ is occupied}, \\ 0, \text{Otherwise}. \end{cases} \quad (22)$$

The total number of employed ECDs in ec_z is calculated by

$$\mu_z(t) = \sum_{n=1}^{N} \lambda_{z,n}(t). \quad (23)$$

Based on the above analysis of resource utilization, the average resource utilization of ec_z is calculated by

$$R_z(t) = \frac{1}{\mu_z(t)} \cdot \sum_{n=1}^{N} R_{z,n}(t). \tag{24}$$

The level of load balance variance for $c_{z,n}$ is embodied by the reciprocal of the variance of the difference between $c_{z,n}$ and ec_z, denoted as

$$L_{z,n}(t) = \frac{1}{\left(R_{z,n}(t) - R_z(t)\right)^2}. \tag{25}$$

Finally, the level of load balance for ec_z is calculated by

$$L_z(t) = \frac{\mu_z(t)}{\sum\limits_{n=1}^{N} L_{z,n}(t) \cdot \lambda_{z,n}(t)}. \tag{26}$$

4 BeCome Design

4.1 ECD Resource Monitoring Using Ledgers in Blockchain

In this section, in a bid to supervise and monitor the resource utilization of the ECDs and the unoccupied VM instances for the IoT in mobile edge computing, a dynamic VM allocation ledger is presented as follows.

Definition 2 (VM allocation ledger). The VM allocation ledger of $c_{z,n}$, denoted as $vat_{z,n}$, is adopted to record the VM allocation when the $c_{z,n}$ receives the offloading request.

Let VAT be the set of VM allocation ledgers, i.e., $VAT = \{vat_{z,n} | 1 \leq z \leq Z, 1 \leq n \leq N\}$. The number of allocations contained in $vat_{z,n}$ is represented as $a_{z,n}$. Let a 4-tuple be an allocation record for the i-th $\left(1 \leq i \leq a_{z,n}\right)$ allocation, which is defined by $ar_{z,n}^{\,i} = \left(rt_{z,n}{}^{i}, ra_{z,n}{}^{i}, st_{z,n}{}^{i}, ot_{z,n}{}^{i}\right)$, where $rt_{z,n}^{\,i}$ represents the responsive task request in $ar_{z,n}^{\,i}$, $ra_{z,n}^{\,i}$ represents the resource allocated for $rt_{z,n}^{\,i}$, and $st_{z,n}^{\,i}$ and $ot_{z,n}^{\,i}$ represent the requested begin time and requested execution time separately.

The VM allocation ledger is updated dynamically. At the arrival of the request, if there are idle VM instances in the destination ECDs, the resource provider assigns the required resources at the requested begin time. Simultaneously, a new record is produced, and the VM allocation ledger is updated correspondingly. Furthermore, if the running VMs are offloaded between ECDs, the VM allocation ledger is also updated, that is, the occupied time is altered.

Algorithm 1 elaborates how the VM allocation ledger is updated.

Algorithm 1 Updating VM allocation ledger

```
Require: $E, VA$
Ensure: $DE$
 1: if $VRT == in$ then
 2:     $a_{z,n}$ = $a_{z,n}$+ 1
 3:     $a = a_{z,n}$
 4:     Generate the record $ar_{z,n}^{\,a}$ based on $vr$
 5:     Add $ar_{z,n}^{\,a}$ into $vat_{z,n}$
 6: end if
 7: if $VRT == out$ then
 8:     for $i$ = 1 to $a_{z,n}$ do
 9:         if $rt_{z,n}^{\,i} == tr$ then
10:             $ot_{z,n}^{\,i} = t - st_{z,n}^{\,i}$
11:         end if
12:     end for
13: end if
14: return $vat_{z,n}$
```

4.2 VM Migration-Based Allocation Using NSGA-III

Dynamic migration provides the policy to transmit running VMs from one ECD to another, without interrupting the execution of the computing tasks. With the ECD resource monitoring presented in Section 4.1, the number of unoccupied VMs on an ECD at any moment in time is evaluated, based on which a VM migration-based resource allocation methodology is presented in this section.

In this paper, to record the offloading strategies of the computing tasks effectively, the definition of the computation offloading table is defined.

NSGA-III performs global searches quickly and with guaranteed quality [14]. The mutation operation also allows the algorithm to converge faster. Considering that the aim of this paper is to solve the multi-objective optimization problem, compared with the traditional genetic algorithm, NSGA-III finds appropriate solutions more efficiently.

Firstly, the ECDs are encoded, and the computation offloading strategies are represented by genes. The genes compose a chromosome that represents a solution of the allocation problem. In this paper, the fitness functions are divided into three categories: time consumption, energy consumption and load balance, presented in (10), (20) and (26), respectively.

The parameters utilized in our method are determined for further selection, including the number of chromosomes B and the iteration time I. Let $j_{b,m}$ be the m-the gene of the b-the chromosome, and the b-th chromosome is represented as $J_b = (j_{b,1}, j_{b,2}, ..., j_{b,M})(b = 1,2, ..., B)$. The standard single-point crossover operation and the usual mutation operation are conducted to create new chromosomes.

After crossover and mutation, B new chromosomes are created; we perform nondominated sorting for the $2B$ solutions based on the three fitness values. Firstly, we search for the minimum fitness values and then subtract the three fitness functions to obtain the updated functions. The intercepts of the axes are represented by the corresponding fitness values, and the axes constitute a 3-dimension hyperplane. Then, the fitness values of each solution are all normalized. After normalization, the value of each solution is in the domain [0, 1). We divide each axis into ψ subsections, and then D reference points are distributed on the 3-dimension hyperplane. To associate the reference points with the normalized solutions, the individuals in the l-th front are sorted based on the number of the associated reference points. Finally, the solutions that are connected with the most points are selected. The process does not stop until the number of individuals in the offspring population equals B.

Algorithm 2 elaborates the process of generating the offloading strategies for computing tasks.

Algorithm 2 Generating candidate computation offloading strategies using NSGA-III

```
Require: The initialized population $X$
Ensure: The optimal offloading strategies in the last iteration $X^*$
1: $i = 1$
2: for $i < I$ do
3:     for the chromosome in $X$ do
4:         Evaluate objective functions by (10), (20) and (26)
5:     end for
6:     Conduct the crossover and mutation operations
7:     for the $2S$ individuals do
8:         Do non-dominated sorting
9:     end for
10:    Select the individuals primarily
11:    if not all individuals in $l$-th front are selected then
12:        Conduct association operation
13:        Classify the individuals in the $l$-th front
14:        Select the $w$individuals
15:    end if
16: end for
17: return $X^*$
```

4.3 Migration Confirmation Using SAW and MCDM

If the time consumption increases, the solution becomes less optimal. Hence, the time consumption is a negative criterion. Furthermore, the energy consumption and the load balance are also negative criteria.

According to the maximum and minimum of each objective function value in the population, the three objective functions are normalized first, denoted as $V(T)$, $V(E)$ and $V(L)$. In addition, the weights of the fitness functions are determined in advance to evaluate the utility values of the solutions. The utility value of the b-th chromosome is denoted by

$$V(J_b) = \mu_1 V(T) + \mu_2 V(E) + \mu_3 V(L), \tag{27}$$

$$\mu_1 + \mu_2 + \mu_3 = 1. \tag{28}$$

Therefore, the optimal one with the highest utility value is selected as the most optimal strategy.

5 Experimental Evaluation

5.1 Simulation Setup

The specified parameter settings for evaluating BeCome are expounded in Table 1.

TABLE 1: Parameter Settings

Parameter description	Value
The total number of ECDs N	20
The latency between two ECDs α	2000Mbps
The latency between the Cloud and the device ω	1000Mbps
The power ε for active VMs	50W
The power ε for idle VMs	30W
The power γ for ECDs	200W
The data size for each computing task	[0, 200]MB
The total number of ECDs	20
The number of VMs in each ECD	11

To assess the efficiency of BeCome, the offloading methods, i.e., the benchmark and the best fit decreasing (BFD) methods, are employed. The key ideas of these two contrastive methods are elaborated as follows.

• Benchmark: The key idea of benchmark is that the task is migrated to the surrounding ECD.

• Best fit decreasing (BFD): The computing tasks are sorted in decreasing order by the resource they require. Then the task to be offloaded is migrated to the ECD that has the smallest resource but enough for the current task.

The simulation of computation offloading is implemented on a desktop PC with Inter Core i7-8700 3.20 GHz processors and 16 GB RAM.

5.2 Comparison Analysis

5.2.1 *Comparative analysis on ECD resource utilization*

After offloading all the tasks to the ECDs by benchmark, BFD and BeCome, the occupation of the VM instances is obtained. Fig. 1 shows the comparison of the ECD resource utilization through benchmark, BFD and BeCome with diverse task sizes. Consequently, from Fig. 1 we can see that BeCome achieves better resource utilization than BFD and benchmark. In other words, BeCome reduces the number of unemployed VM instances more than the other offloading methods. Moreover, when the ECDs are overloaded, all methods achieve full resource utilization because all VM instances in all ECDs are occupied.

5.2.2 *Comparative analysis on load balancing rate*

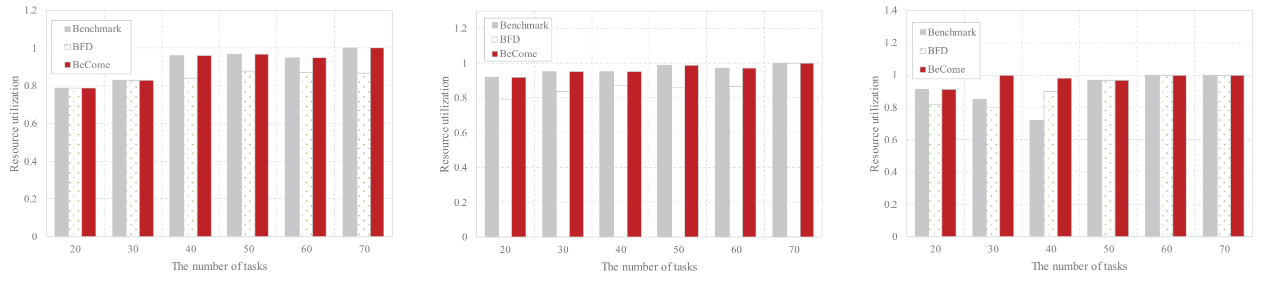
The load balancing rate can show the offload situation of the offloading methods; thus, the comparision of the load balancing rate is exhibited in Fig. 2. It is intuitive that BeCome achieves a better load balancing rate when the ECDs are underloaded. When the ECDs are overloaded, the load balancing rate is 0, which has no meaning to be compared.

5.2.3 *Comparative analysis on offloading time*

Fig. 3 shows the comparison of the offloading time by benchmark, BFD and BeCome with diverse task sizes. When the ECDs are underloaded, BeCome consumes more offloading time from Fig. 3a because when optimizing resource utilization and energy consumption, more offloading time is required to find a suitable ECD to process the tasks, which may sacrifice a small time cost instead. However, when the ECDs are overloaded, from Fig. 3b and Fig. 3c, the flexibility of BeCome to dynamically determine whether tasks are waiting for the available ECDs or just offloadeded to the cloud infrastructure saves more time.

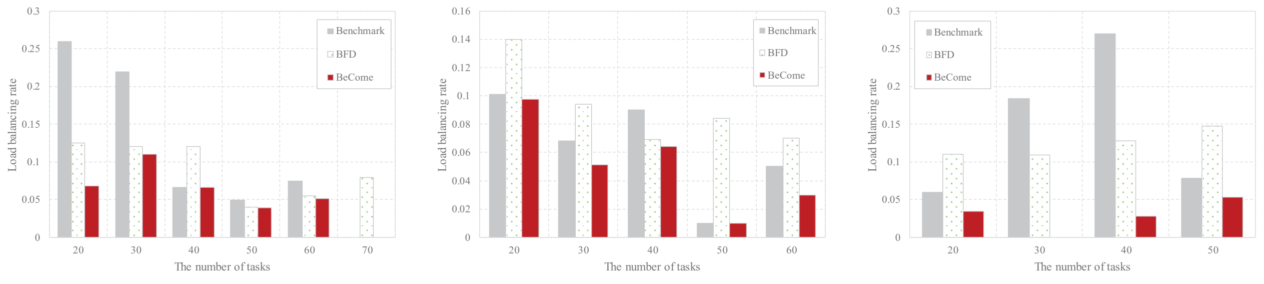
5.2.4 *Comparative analysis on energy consumption*

The comparison of energy consumption is performed in Fig. 4. It shows that BeCome produces less energy than the comparative methods, as BeCome employs fewer ECDs and wastes fewer idle VM instances, which constitute a large part of total energy consumption in ECDs. Furthermore, with the enlargement of the task scale, the consumed energy is increased, which may mean that BeCome may compensate it shortcomings in dealing with large computing task scales.



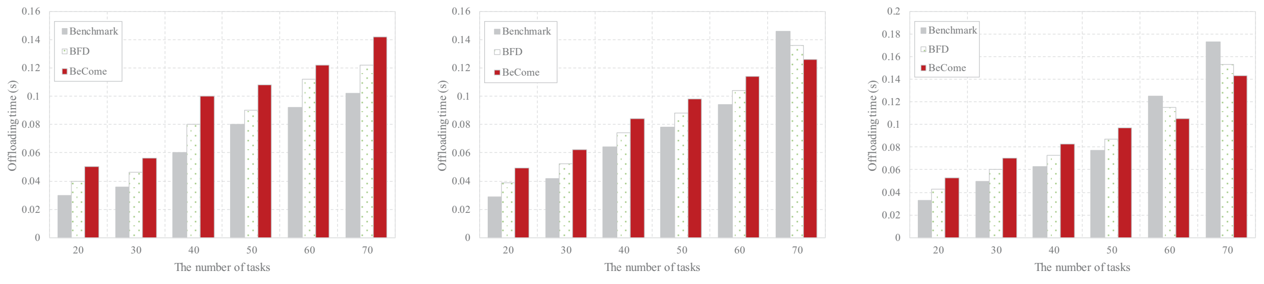
(a) Type-1 ECD (b) Type-2 ECD (c) Type-3 ECD

Fig. 1: Comparison on average resource utilization of ECDs by Benchmark, BFD and BeCome for each ECD type.



(a) Type-1 ECD (b) Type-2 ECD (c) Type-3 ECD

Fig. 2: Comparison on load balancing rate of ECDs by Benchmark, BFD and BeCome for each ECD type.



(a) Type-1 ECD (b) Type-2 ECD (c) Type-3 ECD

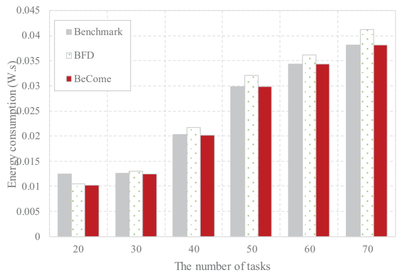
Fig. 3: Comparison on offloading time by Benchmark, BFD and BeCome at different task scales for each ECD type.

6 Related Work

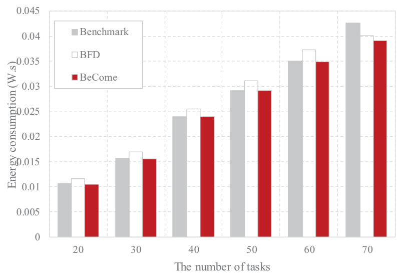
The IoT has birthed a new and key technical field faced with growing demands from users for internetworking technologies [16] [17]. In the IoT environment, the wide application of edge computing brings better performance in reducing delay, saving energy and other aspects [18] [19] [20]. Faced with the requirements of IoT technologies for real-time processing and feedback, edge computing, as a new computing paradigm, has a better capability to solve problems of high transmission delay, low battery life, high bandwidth expenditure and privacy leakage in the IoT realm [21] [22].

In consideration of the energy consumption of data in the IoT, Roy et al. proposed a data routing method to transmit data to base stations via a mobile data collector in order to reduce power consumption and occupation time in the sensing area [23]. In consideration of transmission delay, a novel probabilistic-based channel assignment mechanism was studied by Salameh et al. to minimize the negative impacts brought by delay constraints in [24]. Nevertheless, neither of the two methods took joint consideration of energy consumption and transmission delay. In the realm of cloudlet and MEC, in [25], Dolui et al. discussed and compared the multiple features of the novel paradigms, including fog, cloudlet and mobile edge.

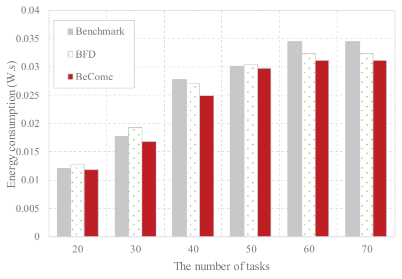
To address the disadvantages of the overloaded mobile edge computing system, Satria proposed two different recovery schemes in [26]. One scheme is that an overloaded



(a) Type-1 ECD



(b) Type-2 ECD



(c) Type-3 ECD

Fig. 4: Comparison of energy consumption by Benchmark, BFD and BeCome at different task scales in each ECD type.

ECD offloads its work to other neighboring devices within transfer range. Aother scheme is that a neighboring ECD's user devices, which are linked with the overloaded ECD, act as ad hoc relay nodes to bridge the two disconnected devices. In [27], Yu et al. designed a scalable and dynamic load balancer (SDLB) in which the core algorithm is minimal hashing, to determine the migration destination of each task.

The increasing volume of resource-intensive applications in MEC leads to severe safety challenges, as it is hard to distinguish between users and attackers [28] [29] [30]. Blockchain technology is employed to secure the stored data and prevent unauthorized changes. In [11], blockchain is integrated with smart contracts to preserve the safety of data storage and sharing. In addition, reputations of vehicles are taken into consideration to preserve the quality of data sharing. In [29], to preserve the privacy of users, blockchain technology is utilized to construct peer-to-peer access to a secure video system. With the aim of decreasing delays, the size of a block is adapted dynamically to enhance the performance of the blockchain.

However, few studies offer comprehensive consideration of data security, time cost of the offloading process, power consumption of ECDs and load balance. It is still a significant challenge to devise an optimized offloading strategy according to the three objectives mentioned above, especially on data security and privacy protection.

7 Conclusion and Future Work

Aiming to decrease the task offloading time and the energy consumption of ECDs while achieving load balance and data integrity in the IoT, we designed the BeCome method. Furthermore, NSGA-III is adopted to produce several feasible task offloading schemes. In addition, SAW and MCDM are adopted with the aim of assessing and selecting the suitable schemes as well as achieving the multi-objective optimization. In the future, we are committed to extend BeCome to the real scene of the IoT, and the number of tasks on each smart device will be determined according to the actual situation.

Acknowledgment

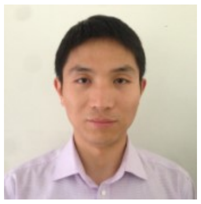
This research is supported by the National Key Research and Development Program of China (No. 2017YFB1400600), the National Science Foundation of China (no. 61702277, no. 61872219 and no. 61672276), the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD) fund, and Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology (CICAEET).

References

[1] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of things (iot): A vision, architectural elements, and future directions, Future generation computer systems 29 (7) (2013) 1645–1660.
[2] X. Zhang, K.-K. R. Choo, N. L. Beebe, How do i share my iot forensic experience with the broader community? an automated knowledge sharing iot forensic platform, IEEE Internet of Things Journal PP (99) (2019) 1–1.
[3] G. Sun, V. Chang, M. Ramachandran, Z. Sun, G. Li, H. Yu, D. Liao, Efficient location privacy algorithm for internet of things (iot) services and applications, Journal of Network and Computer Applications 89 (2017) 3–13.
[4] A. Zanella, N. Bui, A. Castellani, L. Vangelista, M. Zorzi, Internet of things for smart cities, IEEE Internet of Things journal 1 (1) (2014) 22–32.
[5] C. Long, Y. Cao, T. Jiang, Q. Zhang, Edge computing framework for cooperative video processing in multimedia iot systems, IEEE Transactions on Multimedia 20 (5) (2018) 1126–1139.
[6] T. X. Tran, D. Pompili, Joint task offloading and resource allocation for multi-server mobile-edge computing networks, IEEE Transactions on Vehicular Technology 68 (1) (2019) 856–868.
[7] K. Zhang, Y. Mao, S. Leng, Y. He, Y. Zhang, Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading, IEEE Vehicular Technology Magazine 12 (2) (2017) 36–44.
[8] Z. Ning, X. Kong, F. Xia, W. Hou, X. Wang, Green and sustainable cloud of things: Enabling collaborative edge computing, IEEE Communications Magazine 57 (1) (2019) 72–78.
[9] X. Zhang, R. Li, B. Cui, A security architecture of vanet based on blockchain and mobile edge computing, in: 2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN), IEEE, 2018, pp. 258–259.
[10] P. Danzi, A. E. Kalør, Č. Stefanović, P. Popovski, Delay and communication tradeoffs for blockchain systems with lightweight iot clients, IEEE Internet of Things Journal 6 (2) (2019) 2354–2365.
[11] J. Kang, R. Yu, X. Huang, M. Wu, S. Maharjan, S. Xie, Y. Zhang, Blockchain for secure and efficient data sharing in vehicular edge computing and networks, IEEE Internet of Things Journal PP (99) (2018) 1–1.
[12] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, Z. Han, When mobile blockchain meets edge computing, IEEE Communications Magazine 56 (8) (2018) 33–39.
[13] G. Kumar, R. Saha, M. K. Rai, R. Thomas, T.-H. Kim, Proof-of-work consensus approach in blockchain technology for cloud and fog computing using maximization-factorization statistics, IEEE Internet of Things Journal PP (99) (2019) 1–1.

[14] K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints, IEEE Transactions on Evolutionary Computation 18 (4) (2013) 577–601.
[15] Z. Ning, J. Huang, X. Wang, J. Rodrigues, L. Guo, Mobile edge computing-enabled internet of vehicles: Toward energy-efficient scheduling, IEEE Netw. PP (99) (2019) 1–1.
[16] S. Sarkar, S. Chatterjee, S. Misra, Assessment of the suitability of fog computing in the context of internet of things, IEEE Transactions on Cloud Computing 6 (1) (2018) 46–59.
[17] X. Wang, L. T. Yang, H. Li, M. Lin, J. Han, B. O. Apduhan, Nqa: A nested anti-collision algorithm for rfid systems, ACM Transactions on Embedded Computing Systems (TECS) 18 (4) (2019) 32.
[18] A. C. Baktir, A. Ozgovde, C. Ersoy, How can edge computing benefit from software-defined networking: A survey, use cases, and future directions, IEEE Communications Surveys & Tutorials 19 (4) (2017) 2359–2391.
[19] N. Moustafa, K.-K. R. Choo, I. Radwan, S. Camtepe, Outlier dirichlet mixture mechanism: Adversarial statistical learning for anomaly detection in the fog, IEEE Transactions on Information Forensics and Security 14 (8) (2019) 1975–1987.
[20] J. Zhang, N. Xie, X. Zhang, K. Yue, W. Li, D. Kumar, Machine learning based resource allocation of cloud computing in auction, Computers, Materials & Continua 56 (1) (2018) 123–135.
[21] F. Casino, K.-K. R. Choo, C. Patsakis, Hedge: Efficient traffic classification of encrypted and compressed packets, IEEE Transactions on Information Forensics and Security PP (99) (2019) 1–1.
[22] X. Xie, T. Yuan, X. Zhou, X. Cheng, Research on trust model in container-based cloud service, Computers, Materials & Continua 56 (2) (2018) 273–283.
[23] S. S. Roy, D. Puthal, S. Sharma, S. P. Mohanty, A. Y. Zomaya, Building a sustainable internet of things: Energy-efficient routing using low-power sensors will meet the need, IEEE Consumer Electronics Magazine 7 (2) (2018) 42–49.
[24] H. A. B. Salameh, S. Almajali, M. Ayyash, H. Elgala, Spectrum assignment in cognitive radio networks for internet-of-things delay-sensitive applications under jamming attacks, IEEE Internet of Things Journal 5 (3) (2018) 1904–1913.
[25] K. Dolui, S. K. Datta, Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing, in: Global Internet of Things Summit (GIoTS), 2017, IEEE, 2017, pp. 1–6.
[26] D. Satria, D. Park, M. Jo, Recovery for overloaded mobile edge computing, Future Generation Computer Systems 70 (2017) 138–147.
[27] Y. Yu, X. Li, C. Qian, Sdlb: A scalable and dynamic software load balancer for fog and mobile edge computing, in: Proceedings of the Workshop on Mobile Edge Communications, ACM, 2017, pp. 55–60.
[28] J. Xu, S. Wang, B. Bhargava, F. Yang, A blockchain-enabled trustless crowd-intelligence ecosystem on mobile edge computing, IEEE Transactions on Industrial Informatics PP (99) (2019) 1–1.
[29] M. Liu, F. R. Yu, Y. Teng, V. C. Leung, M. Song, Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing, IEEE Transactions on Wireless Communications 18 (1) (2019) 695–708.
[30] Z. Yang, Y. Huang, X. Li, W. Wang, F. Wu, X. Zhang, W. Yao, Z. Zheng, L. Xiang, W. Li, et al., Efficient secure data provenance scheme in multimedia outsourcing and sharing, Computers, Materials & Continua 56 (1) (2018) 1–17.

Xiaolong Xu received his Ph.D. degree from Nanjing University, China, in Dec. 2016. He is currently an assistant professor with the school of computer and software, Nanjing University of Information Science and Technology. He worked as a research scholar at Michigan State University, USA, from Apr. 2017 to May 2018. He has published more than 60 peer review papers in international journals and conferences, including IEEE TCC, IEEE TBD, IEEE TETCI, IEEE ICWS, WWW, SPE, JNCA, FGCS, CCPE, CI, ICSOC, etc. He received the best paper award of IEEE CBD 2016. His research interests include mobile computing, edge computing, IoT, cloud computing and big data.



Xuyun Zhang is a lecturer in the Department of Electrical & Computer Engineering at the University of Auckland, New Zealand. Prior to his current appointment, he worked as a postdoctoral fellow in the Machine Learning Research Group of NICTA (currently Data61, CSIRO) in Australia. He received his PhD degree from University of Technology, Sydney (UTS, Australia) in 2014, as well as his ME and BS degrees in Computer Science from Nanjing University (China) in 2011 and 2008, respectively. His primary research interests include IoT and smart cities, big data, cloud computing, scalable machine learning & data mining, data privacy & security, and Web service technology.

Honghao Gao is currently an associate professor of School of Computer Engineering and Science, Shanghai University, China. He was a JSPS Research Fellow with the Department of Human Informatics and Cognitive Sciences, Waseda University, Japan from Nov. 2012 to Jan. 2013. He was a visiting scholar in Department of Computer Science at the University of California at Santa Barbara supported by China Scholarship Council from Dec. 2015 to Dec. 2016. He has been engaged in the extensively research works in the fields of computer science, service computing, Business process management, and database technology. His current research interests include social computing, data mining and analytics, group behavior modeling and simulating, and service recommendation.

Yuan Xue is currently working towards his B.S. degree in Computer Science and Technology at School of Computer and Software in Nanjing University of Information Science and Technology. His research interests include Blockchain and Mobile Edge Computing.

Lianyong Qi received his PhD degree in Department of Computer Science and Technology from Nanjing University, China, in 2011. Now, he is an associate professor of the School of Information Science and Engineering, Chinese Academy of Education Big Data, Qufu Normal University, China. He has already published more than 50 papers including JSAC, TCC, TBD, FGCS, JCSS, CCPE, ICWS and ICSOC, etc. His research interests include services computing, big data and IoT.

Wanchun Dou received the Ph.D. degree in mechanical and electronic engineering from the Nanjing University of Science and Technology, China, in 2001. He is currently a Full Professor of the State Key Laboratory for Novel Software Technology, Nanjing University. From April 2005 to June 2005 and from November 2008 to February 2009, he respectively visited the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, as a Visiting Scholar. Up to now, he has chaired three National Natural Science Foundation of China projects and published more than 60 research papers in international journals and international conferences. His research interests include workow, cloud computing, and service computing.