Peng *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2021) 10:47
https://doi.org/10.1186/s13677-021-00262-6

Journal of Cloud Computing: Advances, Systems and Applications

RESEARCH Open Access

Security-Aware computation offloading for Mobile edge computing-Enabled smart city

Kai Peng[1*], Peichen Liu[1], Peng Tao[2] and Qingjia Huang[3]

Abstract

Smart city has obtained increasing attention from both academic and industry which has the potential to improve human living standards. A smart city consists of a great number of smart devices which are generating large amounts of data and emerging applications all the time. However, the computing capacity of smart devices are limited. Fortunately, the emergence of MEC can solve the above problem. However, the resources of edge servers in MEC are limited and the capabilities of edge servers are heterogeneous. It is important to improve the average resource utilization of all edge servers and keep load balancing of edge server cluster simultaneously. On the other hand, quite a few numbers of applications are delay-sensitive, it is necessary to ensure the security of these applications. In this paper, we consider joint optimization of mobile device and edge server in MEC-enabled smart city, improving the overall performance of the system. Technically, a new multi-objective computation offloading method is implemented to reduce time consumption, energy consumption, and keep load balancing of edge servers, as well as increase average resource utilization of edge servers while meeting the deadline constraint of delay-sensitive applications. Sufficient experiments have been conducted to prove the effectiveness and superiority of our proposed method in different scenarios.

Keywords: Smart city, MEC, Computation offloading, MOMBI

Introduction

The Internet of Things (IoT) is envisioned to greatly influenced our daily lives and improve socio-economic efficiency and effectiveness through a variety of applications. One of the most promising applications is smart city which utilizes IoT devices to manage cities without any manual intervention [1–4]. Smart cities have greatly changed people's lifestyles and improved people's quality of life with the help of innovative ideas and new technologies [5, 6].

The concept of smart city is originated from the vision of smart earth, which is defined formally in numerous ways [7–9]. The definition of smart city given by IBM is to use information and communication technology to analyze and integrate various critical information of the whole city, so as to make intelligent response of people's various needs which include people's livelihood, environmental protection, public safety, as well as industrial and commercial activities. Sensing devices and intelligent cameras, as well as computing processing units scattered in every corner of the city have facilitated many attractive applications (e.g., autonomous driving, in-vehicle virtual reality (VR), real-time positioning, machine learning, service recommendation) [10]-[17]. Additionally, these applications bring people a lot of convenience while also consuming a lot of computing resources. Meanwhile, they usually generate large amounts of data, which brings huge challenges to insufficient bandwidth [18, 19].

Nevertheless, mobile devices (MDs) in smart city have finite computing capacity, while the mobile users (MUs)'

*Correspondence: kai.peng@hqu.edu.cn
[1]College of Engineering, Huaqiao University, Quanzhou, People's Republic of China
Full list of author information is available at the end of the article

demand for services is usually unlimited [20]. Fortunately, mobile edge computing (MEC)-enabled smart city is regarded as a promising paradigm to solve the above issue, aiming to deploy a myriad of infrastructures (e.g., access points) and edge servers (ESs) in close proximity to mobile devices for the provision of elastic computing resources and low-latency services [21–23]. Among them, computation offloading is an efficient technology to augment capacity of MDs [24–27]. Meanwhile, quite a few numbers of applications are delay-sensitive, it is necessary to ensure the security of these applications [28, 29].

On the other hand, the computing resources in ES are also limited, and thus it is essential to improve the average resource utilization of all ESs [30]. Additionally, the ESs are heterogeneous, namely, they have different amounts of computing resources, which inevitably increases the difficulty of computation offloading [31]. It is unreasonable to offload application to an ES randomly. From the perspective of ES, it is necessary to consider the load balancing of each ES. In view of this, we consider the joint computation offloading in MEC-enabled smart city to improve the overall performance of MEC-enabled smart city [32]. The main contributions are summarized as follows.

- Both MD and ES have been taken into consideration in this study. More specifically, the time consumption and energy consumption of MDs, as well as resource utilization and load balancing of ES cluster are regarded as the optimization objectives.
- A multi-objective optimization model is built to represent the optimization problem, deadline constraint is considered to ensure the safety of delay-sensitive applications.
- We implement a new optimization method on the basis of many-objective metaheuristic based on the R2 Indicator to obtain the reasonable candidate strategies. And then simple additive weighting (SAW) and multi-criteria decision-making (MCDM) are utilized to obtain the optimal computation offloading strategy.
- We conduct a large number of experimental evaluations to verify the effectiveness and superiority of our proposed method in comparison with other methods in different scenarios.

The related work is introduced firstly. And then, the system model is described. Followed by is our proposed optimization method. Next, the experiments and result analysis are discussed. Finally, we present the conclusion and our future work.

Related work

Computation offloading for MEC Computation offloading is an effective technology to augment the computing capability of MDs in MEC. Liu et al. [33] investigate the multi-objective optimization problem concerning average price cost, average execution time, and average energy consumption in MEC environment. The objective of them is to maximize the revenue of each MU. Due to the limited capacity of MDs, it is hard to compute tasks with large amounts of data effectively. Therefore, a system model consisting of the execution time and energy consumption is established in [34]. Chen et al. [35] focus on the computation offloading for multiple MUs and multiple tasks. Correspondingly, two different kinds of greedy maximal scheduling algorithms are proposed to solve such an issue. Wu et al. focus on the tradeoff between limited computing capacity and high delay, as well as ensuring the data integrity during the computation offloading. Correspondingly, blockchain technology is used to implement secure computing offloading [36]. In addition, Feng et al. [37] propose a collaborative computation offloading and resource allocation strategy in blockchain-enabled MEC.

Computation offloading for Smart City Aiming at improving the overall edge computing system execution performance while preventing privacy leakage of the IoT devices during the process of service placement. A trust-oriented IoT service placement approach is proposed [38]. Taking into account of the heterogeneity of ES and remote cloud servers, Wu et al. [39] consider to jointly optimize the system utility and the bandwidth resource allocation in hybrid environment composed of ES and remote cloud. A distributed deep learning computation offloading method is developed to obtain optimized offloading solutions. With the aims of preventing the privacy leakage, improving offloading efficiency, as well as promoting utility of ESs, an intelligent computation offloading approach for smart city is proposed in [40].

Different from the above research, we investigate computation offloading in MEC-enabled smart city. Considering the resources of ESs are limited and the ESs are heterogeneous, the overall performance of the system needs to taken into consideration. In view of this, both the MD and the ES have been considered. In particular, we focus on latency-sensitive applications which have a critical impact on the performance of the system. Furthermore, the energy consumption, time consumption of MD, as well as the average resource utilization of ESs and load balancing of ES cluster are considered to optimize jointly.

System model and problem formulation

In this section, MEC-Enabled smart city system model and the multi-objective problem formulation are introduced.

System model

The architecture of MEC-enabled smart city is shown in Fig. 1. Remote cloud is often deployed in suburban areas

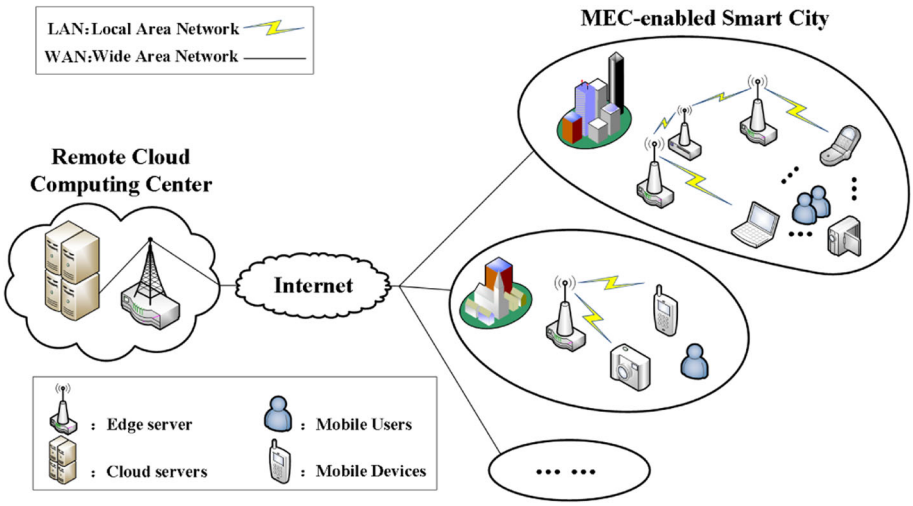

Fig. 1 MEC-Enabled Smart City

located geographically away from MDs with mighty computing power and storage capacity, providing powerful computing resources for MUs. Besides, ESs are deployed in urban areas, providing a certain amount of computing resources for MDs. MDs can communicate with the nearest ES via local area network (LAN). Meanwhile, MDs can also communicate with the remote cloud center through wide area network (WAN) directly.

Let ST be the set of computation offloading strategies, denoted as $ST=\{st_{n,i}|0 \le i \le |ST_n|, 1 \le n \le N\}$, where $st_{n,i}$ represents offloading strategy of the i-th application in $n-th$ MD, which is measured as

$$st_{n,i} = \begin{cases} 0, & \text{if st}_{\text{n,i}} \text{ is executed locally,} \\ 1, ..., S, & \text{if st}_{\text{n,i}} \text{ is offloaded to ES,} \\ S+1, & \text{if st}_{\text{n,i}} \text{ is offloaded to cloud.} \end{cases} \quad (1)$$

where 0 represents the computation task $tk_{n,i}$ is executed locally, 1 represents the task is offloaded to 1−th ES. Accordingly, s represents the task is offloaded to s-th ES. And $s+1$ represents the task is offloaded to cloud center. In addition, $tk_{n,i}$ represents the i-th application of n-th MD.

Time consumption model

In this subsection, the time consumption model is introduced. The total time consisting four parts, namely, the execution time consumption which is represented as $T^E()$, the waiting time consumption which is represented as $T^W()$, and the propagation time which is represented as $T^P()$, as well as the transmission time consumption which is represented as $T^T()$.

Executing time

The executing time refers to the time cost of performing tasks, which is calculated as

$$T^E(st_{n,i}) = \begin{cases} \frac{wk_{n,i}}{f_{end}}, & st_{n,i} = 0, \\ \frac{wk_{n,i}}{f_{edge}}, & st_{n,i} = 1, 2, ..., S \\ \frac{wk_{n,i}}{f_{cloud}}, & st_{n,i} = S+1, \end{cases} \quad (2)$$

where $wk_{n,i}$ represents the task workload. The computation frequency of MD is represented as f_{end} and the computation frequency of ES is represented as f_{edge}, and f_{cloud} represents the computation frequency of the cloud center.

Waiting time

Due to the resources of ES are limited, the computing resources of ESs are represented by a set of virtual machines(VMs). When the tasks which have been offloaded to the ES exceed the number of VMs, some of the tasks need to queue up wait for the ES to calculate the previous tasks, which causes time consumption in the waiting queue. Firstly, we use a double-tuple $v_b^s = (work, tn)$ to represent b-th VM in s-th ES, where $work$ represents a set of total tasks of the k-th VM and the tn represents the number of tasks in VM. Secondly, according to the computation offloading strategy $st_{n,i}$, $tk_{n,i}$ is offloaded to the minimum occupancy VM in e-th ES. Then, the $work$ is updated by $work = work + wk_{n,i}$ and the $tn = tn + 1$. Repeat this until the remaining applications are scheduled to the corresponding VM. Finally, $T^W(st_{n,i})$ is calculated as

$$T^W(st) = \sum_{n=1}^{N} \sum_{i=1}^{|tk_n|} T^e(pre(st_{n,i})) \cdot \wp_{n,i,q} \quad (3)$$

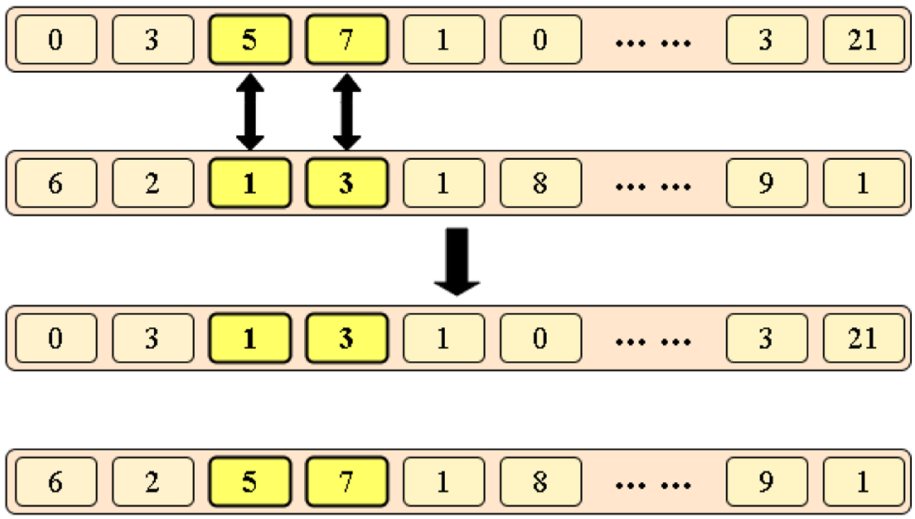

Fig. 2 Crossover operation

where $\wp_{n,i,w}$ is to estimate whether $tk_{n,i}$ is scheduled to the waiting queue in the q_w, and the $T^e(pre(st_{n,i}))$ represents the execution time of previous task.

Propagation time

The propagation time consumption depends on strategy $st_{n,i}$, represented by

$$T^P(st_{n,i}) = \begin{cases} 0, & st_{n,i} = 0, \\ L_{edge}, & st_{n,i} = 1, 2, ..., S \\ L_{cloud}, & st_{n,i} = S+1, \end{cases} \tag{4}$$

where L_{edge} represents the propagation latency between MUs and ESs, which represents LAN. L_{cloud} represented propagation latency between MUs and cloud center, which represents WAN.

Transmission time

Let $T^T(st_{n,i})$ be the transmission latency which can be calculated by

$$T^T(st_{n,i}) = \begin{cases} 0, & st_{n,i} = 0, \\ \frac{wk_{n,i}}{B_{edge}}, & st_{n,i} = 1, 2, ..., S \\ \frac{wk_{n,i}}{B_{cloud}}, & st_{n,i} = S+1, \end{cases} \tag{5}$$

If a task is executed locally by MD, there is no time consumption of transmission. Correspondingly, B_{edge} and B_{cloud} represent bandwidth of ES and cloud, respectively. B_{edge} is the bandwidth of a LAN, and B_{cloud} equals to the bandwidth of WAN. The corresponding bandwidth can be expressed as

$$B = \begin{cases} B_{edge} = be, & st_{n,i} = 1, 2, ..., S \\ B_{cloud} = bc, & st_{n,i} = S+1, \end{cases} \tag{6}$$

Total time consumption

The total time consumption $T(st)$ is obtained by the offloading strategy of i-th application in n-th MD, calculated as

$$T(st) = \sum_{n=1}^{N} \sum_{i=1}^{|tk_n|} \left(T^E(st_{n,i}) + T^T(st_{n,i}) + T^P(st_{n,i}) + T^W(st_{n,i}) \right) \tag{7}$$

Energy consumption model

There are four kinds of energy consumption, namely, the execution energy consumption $E^E()$, the waiting energy

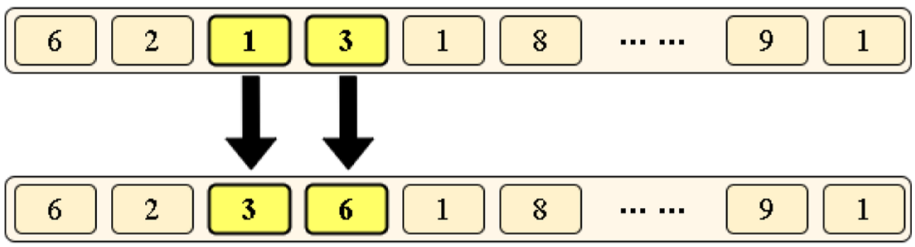

Fig. 3 Mutation operation

Table 1 Parameters Setting

Parameter	Value
The active power of MD	0.6 W
The idle power of MD	0.01 W
The transmitted power of MD	0.5 W
The propagation time of LAN	2 ms
The propagation time of WAN	20 ms
The bandwidth of LAN	200 kps
The bandwidth of WAN	150 kps
The computing frequency of MD	500 Mhz
The computing frequency of the remote cloud	2000 MHz
The computing frequency of the ESs	800 MHz
The number of ESs in multi-user experiment	5
The number of ESs in multi-application experiment	5
The number of ESs in large-scale application experiment	20
The value of the time constraint for each application	4.8
The maximum number of VMs in ES	25
The minimum number of VMs in ES	15

consumption $E^W()$, the propagation energy consumption $E^P()$, and the transmission energy consumption $E^T()$.

Executing energy consumption

Executing energy consumption represents the energy consumption of task executing which can be obtained by multiplying the power by the execution time. And it can be calculated as

$$E^E(st_{n,i}) = \begin{cases} \frac{wk_{n,i}}{f_{end}} \cdot p^a, & st_{n,i} = 0, \\ \frac{wk_{n,i}}{f_{edge}} \cdot p^i, & st_{n,i} = 1, 2, ..., S \\ \frac{wk_{n,i}}{f_{cloud}} \cdot p^i, & st_{n,i} = S+1, \end{cases} \tag{8}$$

where the execution time can be calculated by Eq. (2), p^a represents active energy, and p^i represents idle energy.

Waiting energy consumption

The waiting energy consumption refers to the energy consumption generated by tasks of MD in a waiting queue. When the waiting time is calculated, it is easier to obtain the waiting energy consumption. In addition, the waiting time consumption is calculated by Eq. (3), and the waiting energy consumption is equal to waiting time multiplying ideal energy consumption.

$$E^W(st) = T^W(st_{n,i}) \cdot p_i \tag{9}$$

Propagation energy consumption

In this part, the propagation energy consumption calculated by

$$E^P(st_{n,i}) = \begin{cases} 0, & st_{n,i} = 0, \\ EC(st_{n,i}), \cdot p^i & st_{n,i} = 1, 2, ..., S \\ EC(st_{n,i}), \cdot p^i & st_{n,i} = S+1, \end{cases} \tag{10}$$

Transmission energy consumption

Energy consumption of transmission is represented by $E^T()$. The transmission energy consumption of MDs is related to the transmission time and can be calculated as

$$E^T(st_{n,i}) = \begin{cases} \frac{wk_{n,i}}{B_{end}} \cdot p^t, & st_{n,i} = 0, \\ \frac{wk_{n,i}}{B_{edge}} \cdot p^t, & st_{n,i} = 1, 2, ..., S \\ \frac{wk_{n,i}}{B_{cloud}} \cdot p^t, & st_{n,i} = S+1, \end{cases} \tag{11}$$

where p^t is represented the energy consumption of transmission.

Total energy consumption

Based on the above analysis, the total energy consumption $E(st)$ can be calculated as

$$E(st) = \sum_{n=1}^{N} \sum_{i=1}^{|tk_n|} \left(E^E(st_{n,i}) + E^T(st_{n,i}) + E^P(st_{n,i}) + E^W(st_{n,i}) \right) \tag{12}$$

Resource utilization model

The computing resources of ESs are limited. Visualization technology [41] is used to represent the resources of ES. Thus, Resource utilization can be calculated by the usage of VMs. The VMs in the ESs are instantiated as multiple VM instances with heterogeneous computing capacity.

The resource utilization can be measured by the number of the active VM instances in the VM pool. It is assumed that vm_m is the number of VM instances of the m-th ES e_m. The resource utilization could be expressed as

$$C_m = \frac{1}{vm_m} \cdot \sum_{i=1}^{I} \sum_{j=1}^{J} r_{i,j} \cdot O_{i,j}^{m} \tag{13}$$

where $r_{i,j}$ is the VM instances occupied by $st_{n,i}$ and $O_{i,j}^{m}$ is a binary flag to determine whether the task $st_{n,i}$ is performed by the ES s_m, which is calculated by

$$O_{i,j}^{m} = \begin{cases} 1, & \text{if } \mathrm{st_{n,i}} \text{ is performed by the } \mathrm{s_m}, \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

A binary flag F_m is utilized to represent the status of the ES e_m which is calculated by

$$F_m = \begin{cases} 1, & \text{if } \mathrm{e_m} \text{ is employed}, \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

hereafter, the number of the employed ESs can be calculated by

$$EE = \sum_{m=1}^{M} F_m \tag{16}$$

Above all, the average resource utilization can be calculated by

$$ARU(st) = \frac{1}{EE} \sum_{m=1}^{M} C_m \tag{17}$$

Load balancing model

Different from the average resource utilization, load balancing is a negative indicator. When its value is lower, the state of the ES cluster is much better. Additionally, based on the average resource utilization illustrated in Eq. (17), the average load balancing value can be obtained by

$$LB(st) = \frac{1}{UK} \cdot \sum_{w=1}^{W} [\, UV(st_{n,i})) - UC(st_{n,i})]^2 \tag{18}$$

Problem formulation

The aim of this paper is to minimize the time consumption and energy consumption of MDs and load balancing of ESs, and maximize the resource utilization of ES. The multi-objective optimization issue can be formulated as

$$\mathrm{Min}\,\{\mathrm{T(st)}\}, \mathrm{Min}\,\{\mathrm{E(st)}\}, \mathrm{Min}\,\{\mathrm{LB(st)}\} \tag{19}$$

$$\mathrm{Max}\,\{\mathrm{UV(st)}\} \tag{20}$$

$$\mathrm{s.t.\ st_{n,i}} \in \{0, 1, 2, ..., \mathrm{E} + 1\} \tag{21}$$

$$T(st) \leqq deadline \tag{22}$$

where $T(st)$ represents the time-constraint which means that the total time consumption should not overstepping the given deadline.

Algorithm design

In this section, we describe our proposed method, namely, a multi-objective collaborative optimization for smart

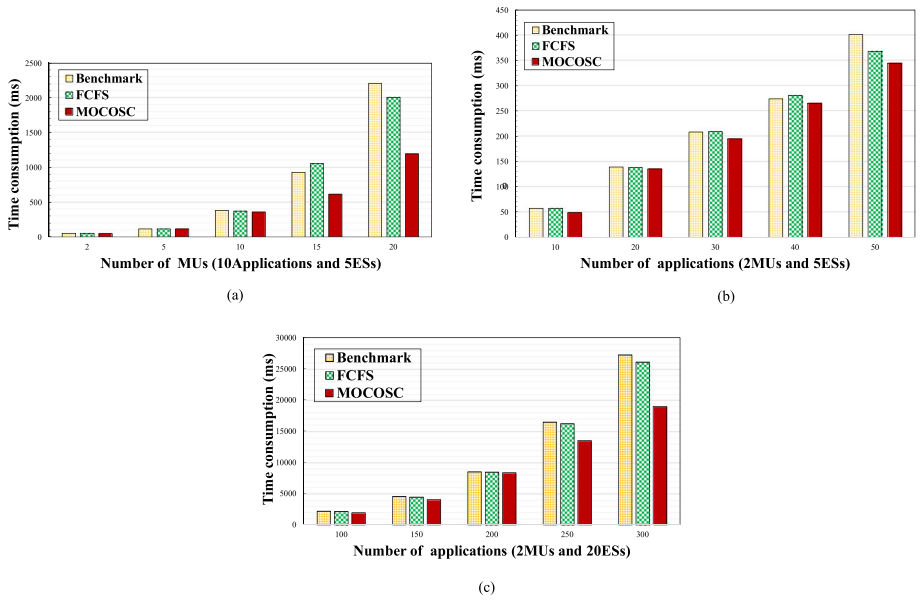

Fig. 4 Comparison of time consumption

city (MOCOSC)based on the multi-objective optimization genetic algorithm MOMBI [42]. Different from the traditional multi-objective optimization algorithm, the MOMBI uses the R2 indicator instead of a non-dominated sort. R2 indicator focuses on the Pareto front rather than the edge, which can achieve an elite solution set. By correcting the inherent deviation of the R2 indicator, MOCOSC achieves different sets of Pareto approximate solutions as computation offloading strategies.

The main steps of MOCOSC are shown as follows. Firstly, some initialization work needs to be completed. Secondly, new populations are created by crossover and mutation operations. Then, solutions are ranked by R2 ranking and selected to the next population using tournament selection. Finally, the optimal solution is selected by using SAW [43] and MCDM [44] in the set of the solution in the Pareto front.

Initialization

Firstly, some parameters are initialized, such as the size of population N_p, the size of archive set N_{arc}, the probability of crossover P_c, the probability of mutation P_m, the number of iterations G_{max}, and current iteration index G_{cur}.

After setting the above parameters, the algorithm is ready to run. The first population should be generated randomly, represented by pop_1, and create an empty archive arc_1 to keep archive set of population.

Crossover and mutation

The aim of crossover operation is to retain the characteristics of the parent population. In the crossover operation, through the crossover operation at the cross point, a new offspring population is generated while maintaining the characteristics of the parent population. In crossover points, the crossover operation is to exchange the corresponding value of two offloading strategies. An example of crossover operation is shown in Fig. 2. It can be seen that the crossover operation occurs between 1 and 5, as well as between 7 and 3.

The aim of mutation operation is to maintain the diversity of the population. An example of mutation operation is shown in Fig. 3. The offloading strategy is mutated from 1 to 3 which means the 3−th ES is selected to provide services rather than 1−th ES. Similarly, the 3 on the right side of the figure is mutated to 6 indicating that the 6−th ES will replace 3−th to provide services.

After these two operators, the new offspring N_p^new is generated and served as the population for the next iteration.

R2 ranking and reference points

In this section, we introduce R2 ranking of MOMBI. R2 ranking is based on R2 indicator. R2 indicator divides the non-dominated layer by measuring the utility function. The utility function represents the value of each objective which is obtained by the weighted Tchebycheff method. The weighted Tchebychef is defined as

$$WT(A, W) = -\frac{1}{|W|}\sum_{\omega\in W}\min_{\alpha\in A}\mu(\alpha) \tag{23}$$

where W represents the weight of each fitness function, A is the set of Pareto approximation and μ represents the utility functions.

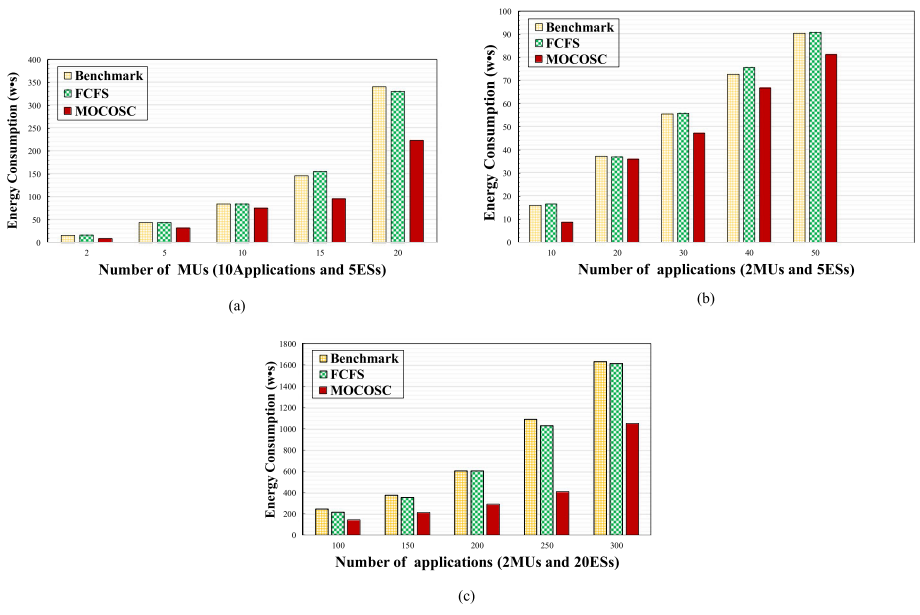

Fig. 5 Comparison of energy consumption

In Eq. (23), the utility functions μ need to be updated in each ranking. The utility function is updated by the reference point and the objective function. The updating method of the reference point can be obtained by

$$f_t^*(st_{n,i}) = \frac{f_t(st_{n,i}) - z_t^{min}}{z_t^{max} - z_t^{min}} \tag{24}$$

where z_t^{max} and z_t^{min} are the maximum and minimum objective function value in pop_t.

Combing with Eqs. (23) and (24), the R2 ranking of MOMBI can be defined as

$$rank_p = \bigcup_{w \in W} \min_{a \in A/B_k} \{ \max_{i \in 1,\ldots,t} w_i | \frac{f_t(st_{n,i}) - z_t^{min}}{z_t^{max} - z_t^{min}} | \} \tag{25}$$

Selection

In this section, the selection operation of MOMBI is described. The goal of selection operation is to select the solution with optimal characteristics as the parent population for the next generation population. MOMBI uses the tournament selection to select solutions. Firstly, two solutions with the same probability are selected. Then, the better one is selected as the next generation parent population by comparing their objective function values. To repeat the above steps until enough next-generation populations are generated.

Optimal selection

After the maximum iteration, a set of Pareto solutions is generated, and SAW [43] and MCDM [44] are exploited to select the optimal solution as the computation offloading strategy. According to the core idea of these two methods, the normalized form of the four goals can be expressed as follows.

Time consumption of i-th application in n-th MD can be normalized as

$$V_s(T(st_{n,i})) = \left\{ \begin{array}{ll} \frac{T_{max} - T(st_{n,i})}{T_{max} - T_{min}}, & T_{max} - T_{min} > 0 \\ 1, & T_{max} - T_{min} = 0 \end{array} \right\} \tag{26}$$

Similarly, the energy consumption can be normalized as

$$V_s(E(st_{n,i})) = \left\{ \begin{array}{ll} \frac{E_{max} - E(st_{n,i})}{E_{max} - E_{min}}, & E_{max} - E_{min} > 0 \\ 1, & E_{max} - E_{min} = 0 \end{array} \right\} \tag{27}$$

The resource utilization can be normalized as

$$V_s(ARU(st_{n,i}) = \left\{ \begin{array}{ll} \frac{ARU_{max} - ARU(st_{n,i})}{ARU_{max} - ARU_{min}}, & ARU_{max} - ARU_{min} > 0 \\ 1, & ARU_{max} - ARU_{min} = 0 \end{array} \right\} \tag{28}$$

The load balancing can be normalized as

$$V_s(LB(st_{n,i})) = \left\{ \begin{array}{ll} \frac{LB_{max} - LB(st_{n,i})}{LB_{max} - LB_{min}}, & LB_{max} - LB_{min} > 0 \\ 1, & LB_{max} - LB_{min} = 0 \end{array} \right\} \tag{29}$$

In order to select the optimal solution, we need to consider four objectives together. Correspondingly, we use α_t, α_e, α_u and α_l to represent weight of each objective, respectively. And the constraint of four vectors is defined as $\alpha_t + \alpha_e + \alpha_u + \alpha_l = 1$. And then, we define the maximum utility value of chromosome in the optimal population V_{max} which is expressed as

$$\begin{aligned} V_{max} = \max [&\alpha_t \cdot V_s(T(st)) + \alpha_e \cdot V_s(E(st)) \\ &+ \alpha_u \cdot V_s(ARU(st) + \alpha_l \cdot V_s(LB(st))] \end{aligned} \tag{30}$$

Method overview

Algorithm 6 illustrates the procedure of MOCOSC, the goal of MOCOSC is to reduce the time and energy consumption of MDs, and the load balancing while increasing the resource utilization of ESs. Firstly, the first generation population of offloading strategies is initialised as POP_1 (Line 1). Secondly, evaluate four objectives of population by Algorithm 1-4. Algorithm 1 and Algorithm 2 calculate the time and energy consumption of MDs, and Algorithm 3 and Algorithm 4 calculate the resource utilization and load balancing of ESs (Lines 2-5). Then, calculate reference points of objective functions (Line 6). Afterwards, R2 ranking is executed to divide non-dominated solution set by Algorithm 5. After performing these steps, a set of reference points and a population are generated, and the reference points and populations are iterated. Then, the tournament selection is executed to find best solution (Line 9), and the mutation and crossover are performed to generate next generation (Lines 7-8). Repeat the following steps until the maximum number of iterations is reached. Namely, evaluate population and update the reference point and execute the R2 ranking by Algorithm 5 (Lines 10-13). Finally, the population size is reduced to N_{pop} (Line 14). After that, MOCOSC adopt SAW and MCDM method to select an optimal offloading strategy (Line 17).

Comparison and analysis of experimental results

In this section, we conduct experiments to prove the effectiveness and superiority of MOCOSC. Firstly, the experimental setting is described. Followed by is the experimental evaluation and discussions.

Experimental setting

In our experiment, by default, it is supposed there are 5 ESs providing service, each of which has heterogeneous computing capacity. We design three sets of experiments. The first one is an experiment with different numbers of MUs, which is set from 2 to 20 and each MU has 10

Algorithm 1 Time consumption calculation

Input: Offloading strategy **st**
Output: Time consumpiton **T(st)**

for n= 1 to N **do**
  for i= 1 to $|TK_n|$ **do**
    $T^E(st_{n,i})$ is calculated by Eq. (2)
    $T^W(st_{n,i})$ is calculated by Eq. (3)
    $T^P(st_{n,i})$ is calculated by Eq. (4)
    $T^T(st_{n,i})$ is calculated by Eq. (5)
    $T(st_{n,i})$= $T^E(st_{n,i})$+ $T^T(st_{n,i})$+ $T^P(st_{n,i})$+ $T^W(st_{n,i})$
    If($T(st_{n,i}) > T_{ddl}$)
      Regenerate solution by Algorithm 1
  end for
end for
return T(st)

Algorithm 2 Energy consumption calculation

Input: Offloading strategy **st**
Output: Energy consumpiton **E(st)**

for n= 1 to N **do**
  for i= 1 to $|TK_n|$ **do**
    $E^E(st_{n,i})$ is calculated by Eq. (8)
    $E^W(st_{n,i})$ is calculated by Eq. (9)
    $E^P(st_{n,i})$ is calculated by Eq. (10)
    $E^T(st_{n,i})$ is calculated by Eq. (11)
    $E(st_{n,i})$= $E^E(st_{n,i})$+ $E^T(st_{n,i})$+ $E^P(st_{n,i})$+ $E^W(st_{n,i})$
  end for
end for
return E(st)

applications. A second one is an experiment with different application scales, which is set from 10 to 50, and the number of MUs is fixed at 2. The last one is conducted to test the situation of large number of applications, the number of MUs is set from 100 to 300. Under this situation, the number of ESs has also increased to 20 accordingly. Each experiment is performed 10 times, and the results of the experiment are the average value of the 10 experiments. More detailed experimental parameters are shown in Table 1.

Two comparison methods named FCFS and Benchmark are introduced, the details of which are shown as follows.

- **Benchmark** All applications are executed locally, or are offloaded to ES cluster, or cloud randomly. Especially, the first task and last task of each MU are executed locally by MD.
- **First Come First Service (FCFS)** All applications are executed sequentially, namely, the first one is executed locally, the second one is executed in ES1, and so on, the $N+1$−th one is executed in cloud. Similarly to Benchmark, the last task of each MU are executed locally.

Algorithm 3 Average resource utilization calculation

Input: Offloading strategy **st**
Output: Resource utilization variance **ARU(st)**

for n= 1 to N **do**
  for tk= 1 to $|TK_n|$ **do**
    The resource utilization of all ESs is calculated by Eq. (13)
  end for
end for
The average resource utilization is calculated by Eq. (17)
return ARU(st)

Algorithm 4 Load balancing calculation

Input: Offloading strategy **st**
Output: Load balance variance **LB(st)**

The single load variance is calculated
for s= 1 to S **do**
  The load balancing of each ES is calculated by Eq. (18)
end for
return LB(st)

Algorithm 5 R2 Ranking

Input: population $\boldsymbol{pop_t}$ weight vectors $\boldsymbol{W}$ reference point $\boldsymbol{z_*}$
Output: Ranking of the population $\boldsymbol{pop_{t*}}$

while $w \in W$ **do**
  while $pop \in POP$ **do**
    The utility value $pop.a$ is calculated
    if $pop.a < pop.u^*$ **then**
$pop.u^* \leftarrow pop.a$
    end if
  end while
  Sort the population POP $rank \leftarrow 1$
  if Number of$(S_t) < N_{pop}$ **then** $pop.rank \leftarrow rank$
  end if
$rank \leftarrow rank+1$
end while

The experiments are processed in Eclipse based on JAVA language on a PC with 8 Intel Core i7-10850H 2.7GHz processors with 16GB RAM and the operating system is Win10 64bit.

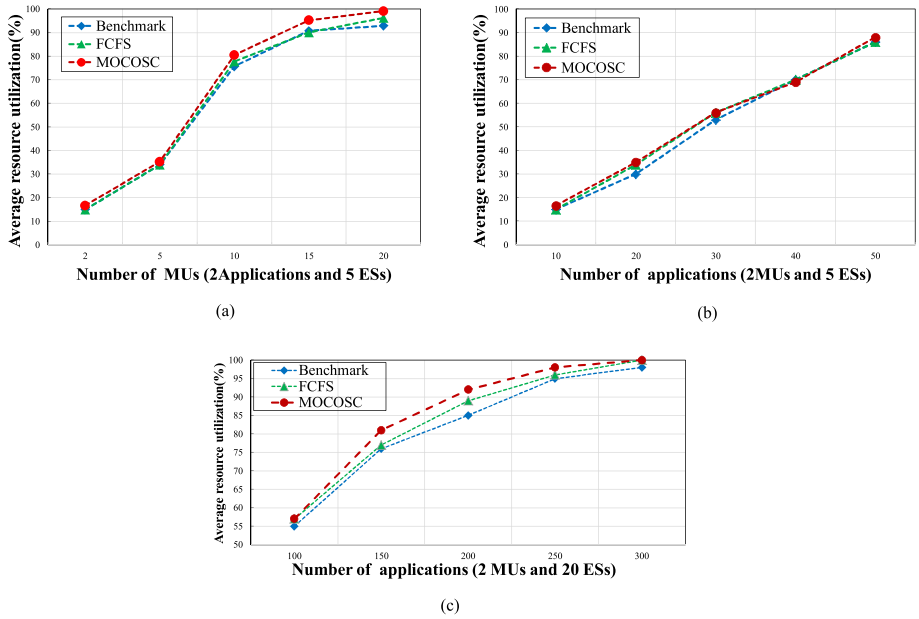

Fig. 6 Comparison of average resource utilization

Experimental evaluation and discussion

In this subsection, numerous and rigorous comparison experiments are conducted to analyze the performance of the total time consumption, the energy consumption, as well as the ESs' average resource utilization and load balancing of the above three methods. The relevant results are shown in Figs. 4, 5, 6, 7.

Comparison of time consumption

The total time consumption can be obtained by Eq. (7). Figure 4 illustrates the comparison results of the total time consumption of the above three methods under different number of MDs and different number of applications. Time consumption comparison results of these three methods under different numbers of applications are shown in Fig. 4a. The result of the impact of different applications performed by each MU on the total time consumption is shown in Fig. 4b and Fig. 4c. It can be seen that MOCOSC outperforms other two approaches from the perspective of different situations. More specifically, MOCOSC can widely used for different application scales. The main reason is that MOCOSC can comprehensively consider various factors to allocate application more reasonably to reduce the total time consumption in comparison to Benchmark and FCFS. In sum, we can conclude that MOCOSC is more stable and effective.

Comparison of energy consumption

The total energy consumption can be obtained by Eq. (12). Figure (5) illustrates the comparison results of the total energy consumption of the above three methods under different number of MDs and under different number of applications. Energy consumption comparison results of these three methods under different application scale are shown in Fig. 5a. The result of the impact of different tasks performed by each MU on the total energy consumption is shown in Fig. 5b and Fig. 5c. It can be seen that MOCOSC outperforms the other two approaches in terms of different situations. More specifically, MOCOSC can widely used for different application scales. The main reason is that MOCOSC can comprehensively consider various factors to allocate application more reasonably to reduce the total energy consumption in comparison to Benchmark and FCFS. In conclusion, MOCOSC is the most energy-efficient method among those methods.

Comparison of average resource utilization

The average resource utilization is an essential standard to measure the performance of the ESs. The resource utilization is measured by the quantity of active VM instances in the VM pool of each ES. When all the computing applications have been offloaded to ESs via the offloading method, the occupation of the VM instances is achieved. Higher resource utilization means fewer idle VM instances in the VM pool. The resource utilization can be obtained by Eq. (17).

Figure 6 illustrates the comparison results of the average resource utilization of the above three methods under different number of MDs and under different number of

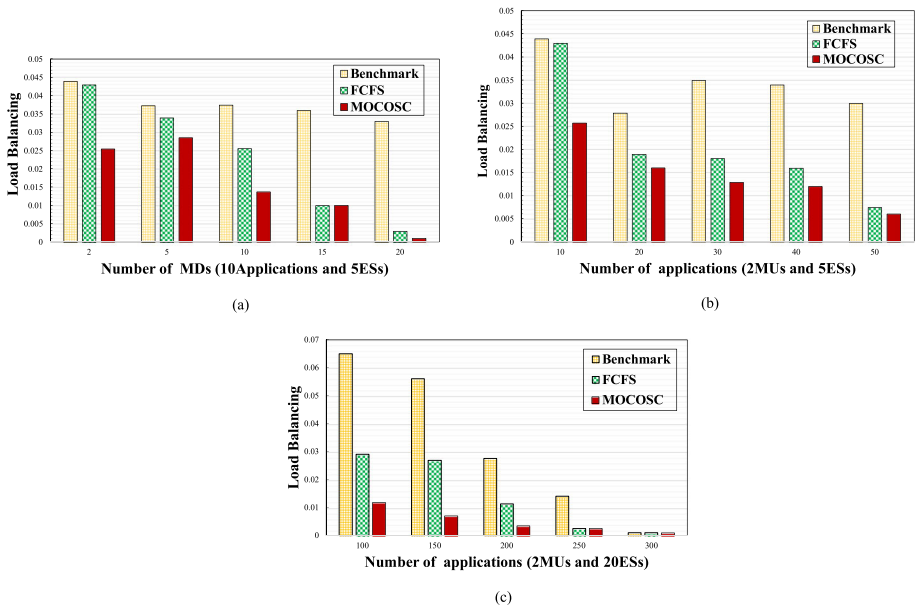

Fig. 7 Comparison of load balancing

applications. The average resource utilization comparison results of these three methods under different application scale are shown in Fig. 6a. The result of the impact of different applications performed by each MU on the average resource utilization is shown in Fig. 6b and Fig. 6c. It can be seen that MOCOSC outperforms other two approaches in terms of different situations. More specifically, MOCOSC can widely used for different application scales. The main reason is that MOCOSC can comprehensively consider various factors to allocate application more reasonably to improve the average resource utilization in comparison to Benchmark and FCFS. Above all, MOCOSC can be widely used for the different situations.

Comparison of load balancing

Similar to the average resource utilization, load balancing is also an essential standard to measure the performance of the ESs. Different from the former one, load balancing is a negative criteria. When its value is lower, the state of the ES cluster is much better. According to the previous mode, Load balancing can be obtained by Eq. (18).

Figure (7) illustrates the comparison results of load balancing of the above three methods under different number of MDs and under different number of applications. The load balancing comparison results of these three methods under different application scale are shown in Fig. 7a. The result of the impact of different applications performed by each MU on load balancing is shown in Fig. 7b and Fig. 7c. It can be seen that MOCOSC outperforms the other two approaches in terms of different situations. Moreover, as the number of user applications increases or the number of users increases, the final load balancing values of the three methods are basically the same. That is because the resources of all ESs are occupied, each ES is in a balanced state. Above all, MOCOSC can be widely used for the different situations.

Conclusion

The combination of smart city and MEC is a promising way to improve our daily life. In this study, we have studied computation offloading for MEC-enabled smart city. Aiming at improve the overall performance of the smart city system, both the MU and ES have been taken into consideration. More specifically, we not only augment the performance of MDs, but also make full use of the resources of ES cluster. For ES, we not only improve the average resource utilization of ES, but also keep the load of the ES cluster in a balanced state. Corresponding, a multi-objective optimization mode is established and a new optimization method is proposed to address this mode. Sufficient experiments and analysis have validated that our proposed method is effective and superior in comparison to the other methods in different situations. In future work, we will focus on privacy-aware computation offloading for scientific workflow application and transmission control in MEC-enabled vehicular networks [45–47].

Algorithm 6 A multi-objective collaborative optimization for smart city

(MOCOSC)

Input: The population size **NPOP**

Output: The optimal offloading strategy $\boldsymbol{ST_{n,i}}$

The optimal time consumption $\boldsymbol{T(st_(n,i))}$
The optimal energy consumption $\boldsymbol{E(st_(n,i))}$
The optimal resource utilization $\boldsymbol{ARU(st_(n,i))}$
The optimal load balance variance $\boldsymbol{LB(st_(n,i))}$
Initialize the first generation population
Time consumption is calculated by Algorithm 1
Energy consumption is calculated by Algorithm 2
Average resource utilization is calculated by Algorithm 3
load balancing is calculated by Algorithm 4
Reference points z_t^{max} and z_t^{min} are calculated
Execute R2 ranking algorithm by Algorithm 5
while $i \neq NPOP$ **do**
    Perform tournament selection
    Perform mutation and crossover operation
    Time consumption is calculated by Algorithm 1
    Energy consumption is calculated by Algorithm 2
    Average resource utilization is calculated by Algorithm 3
    Load balancing is calculated by Algorithm 4
    Update reference points z_t^{max}, z_t^{min}
    Execute R2 ranking by Algorithm 5
    Reduce population $POP_{i+1} \leftarrow \{ POP_i \bigcup POP_{i+1} \}$
    $i++$
end while
SAW and MCDM are used to select the maximum utility individual
return $\boldsymbol{st_(n,i)}, \boldsymbol{T(st_(n,i))}, \boldsymbol{E(st_(n,i))},$
$\boldsymbol{ARU(st_(n,i))}, \boldsymbol{LB(st_(n,i))}$

Acknowledgment
The authors would like to thank all peer reviewers for their good comments.

Authors' contributions
Kai Peng conceived and designed this study. Peichen Liu conducted simulation experiments. Kai Peng wrote this paper. All authors reviewed and edited the manuscript. All authors read and approve the final manuscript.

Funding
This work is supported by the Fundamental Research Funds for the Central Universities(ZQN-817), the National Science Foundation of China (Grant No.61902133), the Natural Science Foundation of Fujian Province (Grant No.2018J05106), Quanzhou Science and Technology Project(No.2020C050R).

Availability of data and materials
The details of experimental parameters are given in Table 1.

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1]College of Engineering, Huaqiao University, Quanzhou, People's Republic of China. [2]Sincetech (Fujian) Technology Co., Ltd, Quanzhou, China. [3]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China.

Received: 7 June 2021 Accepted: 17 August 2021
Published online: 28 August 2021

References

1. Al-Fuqaha A, Guizani M, Mohammadi M, Aledhari M, Ayyash M (2015) Internet of things: A survey on enabling technologies, protocols, and applications. IEEE Commu Surv Tutor 17(4):2347–2376
2. Li R, Song T, Mei B, Li H, Cheng X, Sun L (2018) Blockchain for large-scale internet of things data storage and protection. IEEE Trans Serv Comput 12(5):762–771
3. Gheisari M, Pham QV, Alazab M, Zhang X, Fernandez-Campusano C, Srivastava G (1557) ECA: an edge computing architecture for privacy-preserving in IoT-based smart city. IEEE Access 7:155779–86
4. Qian B, Su J, Wen Z, Jha DN, Li Y, Guan Y, et al. (2020) Orchestrating the development lifecycle of machine learning-based iot applications: A taxonomy and survey. ACM Comput Surv (CSUR) 53(4):1–47
5. Ahmed E, Yaqoob I, Gani A, Imran M, Guizani M (2016) Internet-of-things-based smart environments: state of the art, taxonomy, and open research challenges. IEEE Wirel Commun 23(5):10–16
6. Xie J, Tang H, Huang T, Yu FR, Xie R, Liu J, Liu Y (2019) A survey of blockchain technology applied to smart cities: Research issues and challenges. IEEE Commun Surv Tutorials 21(3):2794–2830
7. Eckhoff D, Wagner I (2018) Privacy in the smart city-applications, technologies, challenges, and solutions. IEEE Commu Surv Tutor 20(1):489–516
8. Rostirolla G, Righi R. d. R, Barbosa JLV, da Costa CA (2018) Elcity: An elastic multilevel energy saving model for smart cities. IEEE Trans Sustain Comput 3(1):30–43
9. Ramaprasad A, Sanchez-Ortiz A, Syn T (2017) A unified definition of a smart city. In: Janssen M, et al. (eds). Electronic Government. EGOV 2017. Lecture Notes in Computer Science, vol 10428. Springer, Cham. pp 13–24. https://doi.org/10.1007/978-3-319-64677-0_2
10. Qi L, Wang X, Xu X, Dou W, Li S (2020) Privacy-aware cross-platform service recommendation based on enhanced locality-sensitive hashing. IEEE Trans Netw Sci Eng. https://doi.org/10.1109/TNSE.2020.2969489
11. Liu Y, Pei A, Wang F, Yang Y, Zhang X, Wang H, Ma R (2021) An attention-based category-aware GRU model for the next POI recommendation. Int J Intell Syst. https://doi.org/10.1002/int.22412
12. Calero C, Mancebo J, García F, Moraga MÁ, Berná JAG, Fernández-Alemán JL, Toval A (2019) 5Ws of green and sustainable software. Tsinghua Sci Technol 25(3):401–414
13. Wang F, Zhu H, Srivastava G, Li S, Khosravi MR, Qi L (2021) Robust Collaborative Filtering Recommendation With User-Item-Trust Records. IEEE Trans Comput Soc Syst:064213. https://doi.org/10.1109/TCSS.2021.3
14. Sánchez MC, de Gea JMC, Fernández-Alemán JL, Garcerán J, Toval A (2019) Software vulnerabilities overview: A descriptive study. Tsinghua Sci Technol 25(2):270–280
15. Pedreira O, Garciía F, Piattini M, Cortiñas A, Cerdeira-Pena A (2020) An architecture for software engineering gamification. Tsinghua Sci Technol 25(6):776–797
16. Maimaiti M, Liu Y, Luan H, Sun M (2020) Enriching the Transfer Learning with Pre-Trained Lexicon Embedding for Low-Resource Neural Machine Translation. Tsinghua Sci Technol. https://doi.org/10.26599/TST.2020.9010029
17. Tekouabou SCK, Hartini S, Rustam Z, Silkan H, Agoujil S (2021) Improvement in automated diagnosis of soft tissues tumors using machine learning. Big Data Min Analytics 4(1):33–46
18. Guezzaz A, Asimi Y, Azrour M, Asimi A (2021) Mathematical validation of proposed machine learning classifier for heterogeneous traffic and anomaly detection. Big Data Min Analytics 4(1):18–24
19. Mahmud MS, Huang JZ, Salloum S, Emara TZ, Sadatdiynov K (2020) A survey of data partitioning and sampling methods to support big data analysis. Big Data Min Analytics 3(2):85–101
20. Zhang Y, Lan X, Li Y, Cai L, Pan J (2018) Efficient computation resource management in mobile edge-cloud computing. IEEE Int Things J 6(2):3455–3466

21. Khan WZ, Ahmed E, Hakak S, Yaqoob I, Ahmed A (2019) Edge computing: A survey. Futur Gener Comput Syst 97:219–235
22. Shi W, Cao J, Zhang Q, Li Y, Xu L (2016) Edge computing: Vision and challenges. IEEE Int Things J 3(5):637–646
23. Mao Y, You C, Zhang J, Huang K, Letaief KB (2017) A survey on mobile edge computing: The communication perspective. IEEE Commun Surv Tutor 19(4):2322–2358
24. Shakarami A, Shahidinejad A, Ghobaei-Arani M (2020) A review on the computation offloading approaches in mobile edge computing: A game-theoretic perspective. Softw: Pract Experience 50(9):1719–1759
25. Xu X, Li Y, Huang T, Xue Y, Peng K, Qi L, Dou W (2019) An energy-aware computation offloading method for smart edge computing in wireless metropolitan area networks. J Netw Comput Appl 133:75–85
26. Huang L, Bi S, Zhang YJA (2019) Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks. IEEE Trans Mob Comput 19(11):2581–2593
27. Cheng N, Lyu F, Quan W, Zhou C, He H, Shi W, Shen X (2019) Space/aerial-assisted computing offloading for IoT applications: A learning-based approach. IEEE J Sel Areas Commun 37(5):1117–1129
28. Xu X, Huang Q, Zhu H, Sharma S, Zhang X, Qi L, Bhuiyan MZA (2020) Secure service offloading for Internet of vehicles in SDN-enabled mobile edge computing. IEEE Trans Intell Transp Syst 22(6):3720–3729
29. Qi L, Hu C, Zhang X, Khosravi MR, Sharma S, Pang S, Wang T (2020) Privacy-aware data fusion and prediction with spatial-temporal context for smart city industrial environment. IEEE Trans Ind Inform. https://doi.org/10.1109/TII.2020.3012157
30. Peng K, Huang H, Wan S, et al. (2020) End-edge-cloud collaborative computation offloading for multiple mobile users in heterogeneous edge-server environment. Wirel Netw. https://doi.org/10.1007/s11276-020-02385-1
31. Peng K, Huang H, Pan W, Wang J (2020) Joint optimisation for time consumption and energy consumption of multi-application and load balancing of cloudlets in mobile edge computing. IET Cyber-Phys Syst: Theory Appl 5(2):196–206
32. Khan LU, Yaqoob I, Tran NH, Kazmi SA, Dang TN, Hong CS (1020) Edgecomputing-enabled smart cities: A comprehensive survey. IEEE Internet Things J 7(10):10200–32
33. Liu L, Chang Z, Guo X, Mao S, Ristaniemi T (2017) Multiobjective optimization for computation offloading in fog computing. IEEE Int Things J 5(1):283–294
34. Xu X, Liu Q, Luo Y, Peng K, Zhang X, Meng S, Qi L (2019) A computation offloading method over big data for IoT-enabled cloud-edge computing. Futur Gener Comput Syst 95:522–533
35. Chen W, Wang D, Li K (2018) Multi-user multi-task computation offloading in green mobile edge cloud computing. IEEE Trans Serv Comput 12(5):726–738
36. Wu H, Wolter K, Jiao P, Deng Y, Zhao Y, Xu M (2020) EEDTO: an energy-efficient dynamic task offloading algorithm for blockchain-enabled IoT-edge-cloud orchestrated computing. IEEE Internet Things J 8(4):2163–2176
37. Feng J, Yu FR, Pei Q, Chu X, Du J, Zhu L (2019) Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: A deep reinforcement learning approach. IEEE Int Things J 7(7):6214–6228
38. Xu X, Huang Q, Yin X, Abbasi M, Khosravi MR, Qi L (2020) Intelligent offloading for collaborative smart city services in edge computing. IEEE Int Things J 7(9):7919–7927
39. Wu H, Zhang Z, Guan C, Wolter K, Xu M (2020) Collaborate edge and cloud computing with distributed deep learning for smart city internet of things. IEEE Int Things J 7(9):8099–8110
40. Xu X, Liu X, Xu Z, Dai F, Zhang X, Qi L (2019) Trust-oriented IoT service placement for smart cities in edge computing. IEEE Int Things J 7(5):4084–4091
41. Hsieh H-C, Lee C-S, Chen J-L (2018) Mobile edge computing platform with container-based virtualization technology for IoT applications. Wirel Pers Commun 102(1):527–542
42. Hernandez Gomez R, Coello Coello C (2013) MOMBI:A New Metaheuristic for Many-objective Optimization based on the R2indicator. In: Proc. 2013 IEEE Congress on Evolutionary Computation. IEEE, Cancun. pp 2488–2495
43. Afshari A, Mojahed M (2010) Simple additive weighting approach to personnel selection problem. Int J Innov, Manag Technol 1(5):511
44. Aruldoss M, Lakshmi TM, Venkatesan VP (2013) A survey on multi criteria decision making methods and its applications. Am J Inf Syst 1(1):31–43
45. Quan W, Liu Y, Zhang H, Yu S (2017) Enhancing crowd collaborations for software defined vehicular networks. IEEE Commun Mag 55(8):80–86
46. Zhang Y, Pan J, Qi L, He Q (2021) Privacy-preserving quality prediction for edge-based IoT services. Futur Gener Comput Syst 114:336–348
47. Quan W, Cheng N, Qin M, Zhang H, Chan HA, Shen X (2018) Adaptive transmission control for software defined vehicular networks. IEEE Wirel Commun Lett 8(3):653–656

Publisher's Note