# A Computation Task Offloading Scheme based on Mobile-Cloud and Edge Computing for WBANs

Rongrong Zhang, Chen Zhou
College of Information Engineering, Capital Normal University, Beijing, China
Email: {zhangrr, 2201002079}@cnu.edu.cn

ICC 2022 - IEEE International Conference on Communications | 978-1-5386-8347-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICC45855.2022.9838921

***Abstract*—The rapid development of Wireless Body Area Networks (WBANs) has brought revolutionary changes to the healthcare system. However, due to the shortcomings of the sink node with limited energy resource and computing capability, it is difficult to handle all computation tasks effectively and timely. The emergence of Mobile Cloud Computing (MCC) and Mobile Edge Computing (MEC) may provide a potential and efficient solution. Therefore, we devote this paper to developing a computation task offloading scheme based on MCC and MEC for WBANs. Technically, we first propose a three-tier system model with one Remote Cloud Server (RCS), several Mobile Edge Servers (MESs) and multiple WBAN users. Then, an optimization problem with the objective to minimize the total cost in terms of the energy consumption and the delay is formulated. In order to solve the problem, we next investigate a Computation Task Offloading Scheme based on Differential Evolution algorithm, called CTOS-DE. The simulation results demonstrate that our proposed CTOS-DE scheme can provide a best computation task offloading decision in terms of the total cost and load balancing.**

***Index Terms*—Computation task offloading, Mobile cloud computing, Mobile edge computing, Wireless body area networks**

## I. Introduction

Wireless Body Area Networks (WBANs) with their promising medical applications in Internet of Things (IoT) have attracted more and more attention from both academic and industry recently. Especially in the post-epidemic era, long-term and non-contact monitoring the patients' physical data through WBANs can dramatically reduce the risk of infection and workload of medical staff. In a WBAN, a smart phone generally plays the role of the sink node to collect data from sensor nodes and transmit them to servers via external gateway [1]. Although the CPU of the smart phones continuously becomes more and more powerful nowadays, many applications running on them still intend to offload computation tasks due to their limited energy resource and computing or hardware capabilities [2]. Moreover, the strict delay requirements of medical applications have become an important obstacle of the evolutionary development of WBANs. As a result, the smart phones cannot handle large amounts of computation tasks in a short time. In addition, we cannot only rely on the endless code optimization with the overflow of the massive amounts of data. Consequently, computation task offloading is one of the most efficient solutions to relief the energy consumption and execution delay of the smart phone in a WBAN.

Fortunately, the Mobile Cloud Computing (MCC) has been considered as a potential approach and an effective technique to provide users with powerful computing resource at a remote centralized cloud sever [3]. However, the remote cloud server is generally logically and spatially far from users, which dramatically leads to huge communication and processing delay and more energy consumption. Hence, the MCC cannot meet the stringent requirement of latency-sensitive applications. As a remedy to these limitations, a new technology, called Mobile Edge Computing (MEC), has recently emerged to enable in data processing at the network edge, which can offer both communication and computational capacities close to users [4]. Due to the distributed and limited computing capabilities of edge servers, the increasing demand of mobile applications for high computation and storage resource makes MEC an insufficient approach for accomplishing solely all the computation tasks. Thus, how to conduct an effective task offloading scheme in order to reduce the energy consumption and satisfy the delay requirement is challenging.

In recent years, several research efforts have been dedicated for the development of computation task offloading. Most of them focused on offloading tasks on either MCC or MEC. Specifically, in order to minimize the sum energy consumption of users, a resource allocation for a MEC system was studied in [5], where the base station made a binary offloading decision based on the computing energy and channel gains. In [6], the authors attempted to integrate MEC system with machine learning, where a multi-task learning enabled offloading framework for a MEC network was presented to save the system cost. Subsequently, the authors explored a joint optimization of offloading decision and computation resource allocation to minimize the total cost in terms of the delay and charge in [7]. A game theoretic approach for the computation offloading decision was proposed in [8], where a Nash equilibrium could be achieved by designing a distributed computation offloading algorithm.

Nevertheless, few studies consider the cooperation between MCC and MEC, which can not only balance the load of servers but also guarantee the QoS requirements of different users. In [9], the authors both considered the single and multiple users offloading problem, where both the resource constraint and the interference among multiple users were taken into consideration. An online peer offloading framework by leveraging the Lyapunov technique was developed in [10] to keep the energy consumption of base stations below a threshold. As human mobility can significantly impact the offloading decision, the authors in [11] proposed a mobility-aware cooperative task

offloading scheme to reduce the service time, the percentage of failed tasks and the energy consumption of WBAN users. Then, a joint task assignment and power allocation problem was investigated in [12], where a dynamic offloading decision was proposed to minimize the total execution latency.

However, most of the existing work mainly considered the system model with a MCC or/and a MEC with multiple users. Indeed, there are generally multiple MECs around WBAN users. Moreover, the QoS requirements of different types of computation tasks are quite different, which is apparently overlooked. Motivated by the above observations, we devote this paper to developing a computation task offloading scheme to reduce the energy consumption of sink node and guarantee the delay requirements of medical tasks simultaneously. The main contributions of this paper are articulated as follows:

- We first develop a three-tier system model with a Remote Cloud Server (RCS), several Mobile Edge Severs (MESs) and multiple WBAN users, where all MESs can connect with each other to offload computation tasks among them.
- Second, an optimal problem with the objective to minimize the total cost in terms of the energy consumption and delay is formulated, where the queuing delay is taken into consideration.
- Third, we propose a Computation Task Offloading Scheme based on Differential Evolution algorithm (CTOS-DE) to provide an optimal offloading decision.
- Finally, the simulations are further conducted to demonstrate that our CTOS-DE scheme performs better in terms of the total cost and load balancing among all MESs.

## II. System Model

In this section, we first describe the system model. And the channel model is then introduced.

### A. System Model

In this paper, we consider a three-tier system model consisting of a Remote Cloud Server (RCS), $M$ Mobile Edge Servers (MESs) and $N$ WBAN users, as shown in Fig.1. Assume that the RCS can provide virtually unlimited available energy and computing resource. The MESs yet with sufficient energy resource and limited computational resource can communicate with the RCS directly and connect with each other through optical fiber link. As there is one and only one sink node in a WBAN, and we consider the full task offloading of a WBAN, it is interchangeable between a "sink node" and a "WBAN" in this paper. Without loss of generally, the sink node which is generally played by a smart phone with insufficient energy and computing resource can communicate with its local MES by wireless link. Notice that a MES can cover multiple WBANs, a WBAN belongs to only one local MES, and a WBAN cannot communicate with the RCS directly.

Let MS $= \{1, 2, ..., M\}$ index the set of all MESs and SS $= \{1, 2, ..., N\}$ denote the set of all sink nodes. For each sink node $S_i$ ($i \in$ SS), we define $M_i$ is the ID of its local MES, which means that $S_i$ can communicate with $M_i$ directly. Correspondingly, let $M_j$ express one of other MESs, that is

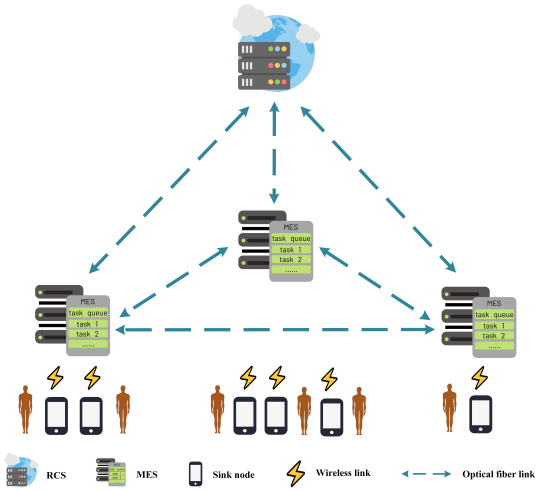


Fig. 1: The system model

$M_j \in$ MS and $M_j \neq M_i$. In this paper, we consider that the computation task of sink node can be executed locally on itself CPU or be fully offloaded to process on the RCS or one MES. Let $r_i \in \{0, 1, ..., M, M+1\}$ represent the computation task offloading decision of $S_i$, thus we can have

$$r_i = \begin{cases} 0 & \text{Local execution} \\ M_i & \text{Local MES execution} \\ M_j & \text{Other MES execution} \\ M+1 & \text{Remote RCS execution} \end{cases}. \quad (1)$$

When $r_i = 0$, it means that the task of $S_i$ will be executed on $S_i$ locally. As $r_i = M_i$, it expresses that the task of $S_i$ will be offloaded to the local MES $M_i$. Correspondingly, the task of $S_i$ will be processed on the MES $M_j$, if $r_i = M_j$. Otherwise, the task will be uploaded to the RCS when $r_i = M + 1$.

### B. Channel Model

For medical applications in IoT, WBANs can monitor and transmit the physiological parameters in complex environment such as a hospital building. In this paper, we consider in a hospital environment where patients are concentrated, and there are many kinds and numbers of obstacles which will greatly affect the wireless link quality. Thus, the pass loss between the sink node $S_i$ and its local MES $M_i$ can be calculated as

$$L(d_{S_iM_i}) = L_0(d_0) + 10n\lg(\frac{d_{S_iM_i}}{d_0}) + \sum_g K_g L_g,$$

where $d_{S_iM_i}$ is the distance between $S_i$ and MES $M_i$. $L_0(d_0)$ denotes the pass loss when the reference distance is $d_0$, i.e., $L_0(d_0) = 10\lg(4\pi d_0/\lambda)^2$, $\lambda$ is the wireless signal wavelength. $n$ represents the path loss factor. $K_g$ and $L_g$ define the number and path loss of the $g$-th obstacle, respectively.

Given the transmission power $P_i^T$ of sink node $S_i$, the pass loss $L(d_{S_iM_i})$ can also be given by

$$L(d_{S_iM_i}) = 10\lg\frac{P_i^T}{P_{M_i}^R},$$

where $P_{M_i}^R$ is the received power of the MES $M_i$.

According to the Shannon equation [13], the transmission rate $V_{S_iM_i}$ between $S_i$ and MES $M_i$ can be calculated as

$$V_{S_iM_i} = B_{S_iM_i}\log_2(1+\frac{P_{M_i}^R}{N_0}), \tag{2}$$

where $B_{S_iM_i}$ is the wireless channel bandwidth, and $N_0$ denotes the background noise power.

## III. Problem Formulation

In this section, we first introduce the processing time model and energy consumption model. Then, the problem formulation is described.

### A. Processing Time Model

We consider that sink node $S_i$ has a computation task $\tau_i = (P_i, C_i, T_i^{\text{Limit}})$ that can be executed either locally on the sink node or remotely on the RCS or one MES via computation offloading. Here $P_i$ denotes the size of computation task $\tau_i$, and $C_i$ defines the total number of CPU cycles required to accomplish the computation task $\tau_i$. As the computation tasks can be simply classified into urgent or non-urgent in WBANs, let $T_i^{\text{Limit}}$ denote the maximum tolerance execution time of computation task $\tau_i$. According to analysis in Eq.(1), we then deliberately discuss the processing time model of computation task $\tau_i$ in four cases.

*1) Local execution:* When the offloading decision $r_i = 0$, the computation task $\tau_i$ will be locally executed on the CPU of sink node $S_i$. Let $f_i^{\text{sink}}$ be the computation capability (i,e., CPU cycles per second) of sink node $S_i$. Note that different sink nodes may have different computation capabilities. Thus, the processing time of computation task $\tau_i$ by local execution is given as

$$T_i^{\text{sink}} = \frac{C_i}{f_i^{\text{sink}}}. \tag{3}$$

*2) Local MES execution:* When the offloading decision $r_i = M_i$, the computation task $\tau_i$ will be offloaded on the CPU of local MES $M_i$ who can communicate directly with sink node $S_i$. In this case, the computation task $\tau_i$ needs firstly to forward to the local MES $M_i$. Given the transmission rate $V_{S_iM_i}$ in Eq.(2), the transmission time of computation task $\tau_i$ can be calculated as

$$T_i^{S_iM_i} = \frac{P_i}{V_{S_iM_i}}. \tag{4}$$

Then, we derive the waiting time of computation task $\tau_i$ in the task queue of the MES $M_i$, which depends on the offloading decision of all sink nodes. Assume that computation task $\tau_i$ is the $b$-th task processed by MES $M_i$. Based on the Queuing Theory, the waiting time can be computed by

$$T_i^{WM_i} = \frac{\sum_{k=1}^{b-1} C_{q_k}}{f_{M_i}^{\text{MES}}}, \tag{5}$$

where $f_{M_i}^{\text{MES}}$ is the computation capability of the MES $M_i$. And $C_{q_k}$ ($q_k \in$ SS) is the total number of CPU cycles of the $k$-th computation task in the task queue of the MES $M_i$.

Next, the execution time of computation task $\tau_i$ on the MES $M_i$ is given by

$$T_i^{EM_i} = \frac{C_i}{f_{M_i}^{\text{MES}}}. \tag{6}$$

Therefore, the processing time of computation task $\tau_i$ executed on the the MES $M_i$ can be formulated as

$$T_i^{M_i} = T_i^{S_iM_i} + T_i^{WM_i} + T_i^{EM_i}. \tag{7}$$

*3) Other MES execution:* When the offloading decision $r_i = M_j$, the computation task $\tau_i$ will be offloaded on the CPU of the MES $M_j$. In this situation, the processing time of computation task $\tau_i$ includes four parts: the transmission time from $S_i$ to MES $M_i$, the transmission time from MES $M_i$ to MES $M_j$, the waiting time of the task queue of the MES $M_j$ and the execution time on the MES $M_j$.

Inspired by the above analysis of cases 1) and 2), we can accordingly have

$$T_i^{M_j} = T_i^{S_iM_i} + \frac{P_i}{V_{M_iM_j}} + T_i^{WM_j} + T_i^{EM_j}, \tag{8}$$

where $V_{M_iM_j}$ represents the transmission rate between the MES $M_i$ and the MES $M_j$.

*4) Remote RCS execution:* As the offloading decision $r_i = M+1$, the computation task $\tau_i$ will be offloaded to the RCS. Correspondingly, the processing time of computation task $\tau_i$ is made up of the transmission time from $S_i$ to MES $M_i$ and from MES $M_i$ to the RCS and the execution time on the RCS, which can be elaborately expressed as

$$T_i^{\text{RCS}} = T_i^{S_iM_i} + \frac{P_i}{V_{M_i\text{RCS}}} + \frac{C_i}{f^{\text{RCS}}}. \tag{9}$$

$V_{M_i\text{RCS}}$ is the transmission rate between the MES $M_i$ and the RCS, and $f^{\text{RCS}}$ denotes the computation capability of the RCS.

In summary, by merging the four cases, the processing time of computation task $\tau_i$ can be formulated as

$$T_i = \alpha T_i^{\text{sink}} + \beta T_i^{M_i} + \mu T_i^{M_j} + \gamma T_i^{\text{RCS}}, \tag{10}$$

where $\alpha+\beta+\mu+\gamma = 1$ and $\alpha,\beta,\mu,\gamma \in \{0,1\}$. Notice that the processing time $T_i$ cannot be larger than the maximum tolerance latency $T_i^{\text{Limit}}$.

### B. Energy Consumption Model

As the energy of sink node is limited, we focus on analyzing the energy consumption model of sink node for each offloading decision. For case 1) that the computation task $\tau_i$ is locally executed on the sink node $S_i$, the energy consumption can be presented as

$$E_i^{\text{sink}} = T_i^{\text{sink}} \cdot P_i^E, \tag{11}$$

where $P_i^E$ defines the execution power of sink node $S_i$.

Correspondingly, as the computation task $\tau_i$ is offloaded on the local MES $M_i$ in case 2), given the waiting power $P_i^W$ of sink node $S_i$, we can derive the energy consumption as

$$E_i^{M_i} = T_i^{S_iM_i} \cdot P_i^T + (T_i^{WM_i} + T_i^{EM_i}) \cdot P_i^W. \tag{12}$$

Subsequently, for case 3) when the computation task $\tau_i$ is forwarded on the MES $M_j$, the energy consumption can be calculated as

$$E_i^{M_j} = T_i^{S_i M_i} \cdot P_i^T + (\frac{P_i}{V_{M_i M_j}} + T_i^{WM_i} + T_i^{EM_i}) \cdot P_i^W. \quad (13)$$

If the computation task $\tau_i$ is executed on the RCS, we thus have

$$E_i^{\text{RCS}} = T_i^{S_i M_i} \cdot P_i^T + (\frac{P_i}{V_{M_i RCS}} + \frac{C_i}{f^{\text{RCS}}}) \cdot P_i^W. \quad (14)$$

As a consequence, the energy consumption model of the sink node $S_i$ can be finally given by

$$E_i = \alpha E_i^{\text{sink}} + \beta E_i^{M_i} + \mu E_i^{M_j} + \gamma E_i^{\text{RCS}}. \quad (15)$$

### C. Problem Formulation

In medical applications, different tasks have the different delay and energy consumption requirements. Generally, for the urgent tasks, we must preferentially ensure the processing time requirement. Nevertheless, the energy consumption is prior to the delay for the non-urgent tasks. In order to simultaneously satisfy the delay and energy consumption requirements, we define the weighted total cost of the computation task $\tau_i$ as

$$S_i = \omega_i \frac{T_i}{T_i^{\text{sink}}} + \varphi_i \frac{E_i}{E_i^{\text{sink}}}, \quad (16)$$

where $\omega_i, \varphi_i \in [0,1]$ and $\omega_i + \varphi_i = 1$ for $\forall i \in$ SS. $\omega_i$ and $\varphi_i$ are the weight factors of the delay and the energy consumption to sink node $S_i$.

Without loss of generally, the computation task offloading for multiple WBANs based on mobile cloud and edge computing can be formulated as an optimization problem. The objective is to minimize the sum weighted total cost of all sink nodes under the constraints of the processing latency and the computation capacities of the RCS, MES and sink node servers. Mathematically, the optimization problem can be expressed as

$$\begin{aligned} \min_{r_i} \quad & \sum_{i=1}^{N} S_i \\ s.t. \quad & r_i \in \{0, 1, \cdots, M, M+1\} \\ & T_i < T_i^{Limit}, i \in \text{SS} \\ & f_i^{\text{sink}} < f_m^{\text{MES}} < f^{\text{RCS}}, m \in \text{MS} \end{aligned} \quad . \quad (17)$$

## IV. Computation Task Offloading Scheme based on Differential Evolution Algorithm

In this section, we propose a computation task offloading scheme based on the Differential Evolution (DE) algorithm to solve the optimization problem (17).

As the offloading decisions of every sink node in our system model are interacted on each other, it turns out unfortunately that the optimal problem (17) is NP-hard [8] which is extremely challenging to solve mathematically. However, the DE algorithm [14] has been proved that it is an efficient and powerful stochastic search technique for solving optimization problems. Therefore, we develop a three-phases Computation Task Offloading Scheme based on the Differential Evolution algorithm (CTOS-DE) to solve the problem (17).

**Algorithm 1** Computation task offloading scheme (CTOS-DE)

**Input**: $L_{\max}$, $\tau_i$, $f_i^{\text{sink}}$, $f_m^{\text{MES}}$, $f^{\text{RCS}}$ for $i \in$ SS, $m \in$ MS
**Output**: $\Upsilon_{\text{best}}$

```
L = 1
Randomly generate T group offloading decisions
  R_T = [Υ_1, ..., Υ_t, ..., Υ_T] where Υ_t = [r_1, ..., r_N]
for all m ∈ MS do
  for each t → T do
    Record the task queue Q_mt
  end for
end for
for each t → T do
  for all i ∈ SS do
    Compute time T_it and energy consumption E_it
    if T_it > T_i^Limit then
      R'_T = R_T - Υ_t
    end if
  end for
end for
for each t → T' do
  Compute the total cost S_t
end for
Υ'_best = min_{r_t} S in R'_T
if L > L_max then
  Υ_best = Υ'_best and return Υ_best
else
  for each t → T do
    Select two offloading decisions Υ_{t_1} and Υ_{t_2} in R_T
    Υ'_t = Υ'_best + F(Υ_{t_1} - Υ_{t_2})
  end for
  for each t → T do
    for all i ∈ SS do
      r''_t = rand (r_t, r'_t)
    end for
  end for
  for each t → T do
    Υ_t = min_{r_t, r''_t} S
  end for
end if
L = L + 1
Repeat 3-36
```

More specifically, each sink node computes the transmission rate between itself and its local MES according to the channel model in section II-B, and then forwards the task information including the computation task size, the total required CPU cycles number and the maximum tolerance latency to the RCS through its local MES, in the first phase. In the next phase, the RCS executes our CTOS-DE scheme which will be formally described in Algorithm 1 after receiving all the computation tasks, and broadcasts the final offloading decisions to all sink

nodes. Subsequently, each sink node offloads its computation task to the corresponding server according to the final offloading decisions during the third phase.

A formal description of the CTOS-DE is shown in the Algorithm 1, and we also offer an operation flowchart in Fig. 2 for the sake of clarity. Specifically, we initialize the system parameters and the task information for all sink nodes, all MESs and the RCS. Then, the RCS randomly generates $T$ group offloading decisions $R_T$ for all sink nodes, and it records the task queue information $Q_{mt}$ for all MESs. Under each offloading decision, the RCS respectively computes the processing time $T_{it}$ and the energy consumption $E_{it}$ based on Eq.(10) and Eq.(15) for all sink nodes as lines 8-15 in the Algorithm 1. If the processing time $T_{it}$ exceeds the maximum latency $T_i^{\text{Limit}}$, we need deleting the $t$-th offloading decision $\Upsilon_t$ from $R_T$. Next, we compare the total cost $S_t$ among all offloading decisions in $R_T^{'}$, and select a tentative best offloading decision $\Upsilon_{\text{best}}^{'}$ with the minimum total cost as in lines 16-19. Subsequently, let $\Upsilon_{\text{best}} = \Upsilon_{\text{best}}^{'}$ and return the final best offloading decision $\Upsilon_{\text{best}}$ if the iteration number satisfies the stopping criterion. Otherwise, we randomly choose two offloading decisions $\Upsilon_{t_1}$ and $\Upsilon_{t_2}$ in $R_T$ to accomplish the mutation operation, i.e.,$\Upsilon_t^{'} = \Upsilon_{\text{best}}^{'} + F(\Upsilon_{t_1} - \Upsilon_{t_2})$ where $F$ is a scaling factor, as in lines 23-26. As a result, we can obtain another $T$ group offloading decisions $R_T^{'}$. After that, it achieves the crossover operation to renew the offloading decision by randomly selecting a value between $r_t$ and $r_t^{'}$ for each sink node under each offloading decision in lines 27-31. Consequently, we can have the third $T$ group offloading decisions $R_T^{''}$. In order to make the set of offloading decisions even better, we finish the selection operation and update the $\Upsilon_t$ by choosing the one that can minimize the total cost $S$ in $r_t$ and $r_t^{''}$ as shown in lines 32-34. Thus, we gain a fresh $T$ group offloading decisions $R_T$ which absolutely perform better in terms of the total cost than the initialized offloading decisions. In the wake of the fresh $R_T$, we repeat the operations as lines 3-36 until it meets the stopping criterion, and then output the final offloading decision $\Upsilon_{\text{best}}$.

Here, we discuss the complexity of Algorithm 1. In each loop, for given $T$ group offloading decisions, we need to record the task queue of each MES, which results in the complexity as $O(TM)$. Then, the mutation, crossover and selection operations are executed among all sink nodes, thus the corresponding complexity can be given by $O(T) + 2 \cdot O(TN)$. Given the maximum loop number $L_{\max}$, the complexity of Algorithm 1 can be approximated by $O(L_{\max}TM) + O(L_{\max}TN)$.

## V. Performance Evaluation

In this section, we evaluate the performance of our proposed CTOS-DE scheme in terms of the total cost and load balancing among all MESs. In the following, we first introduce the simulation settings, and then present the simulation results.

### A. Simulation Settings

In our simulations, we consider a scenario where the WBAN users are randomly located in the hospitals and each hospital has its own MES with 20 GHz CPU frequency. There is only one RCS who is far from WBANs and whose computation capability is 50 GHz. Assume that there are many types of obstacles in a hospital, such as concrete wall, brick wall and glass wall with pass loss 10.8, 2.5∼6.0 and 2.31, respectively. In addition, the wireless signal frequency is 2.4 GHz, and the noise power is set to -100 dBm. Moreover, the other simulation parameters are given in Table I. Furthermore, every simulation result is the average of 100 independent experiments.

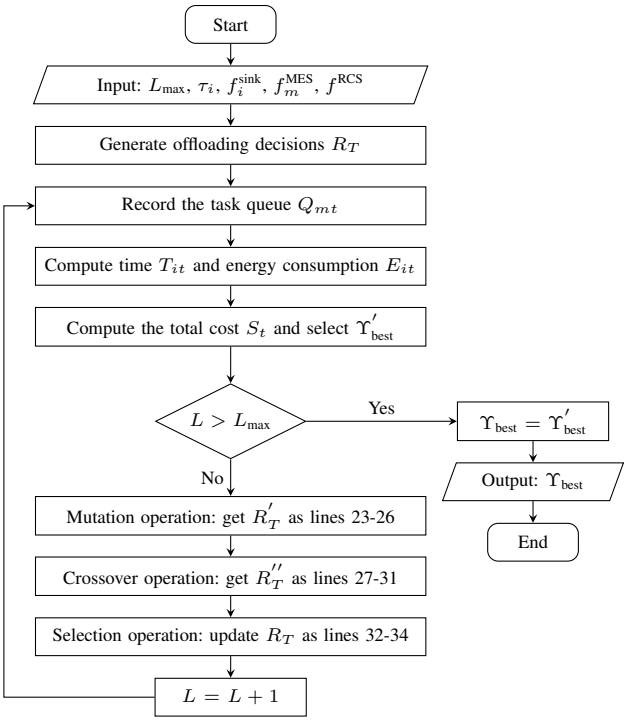


Fig. 2: The flowchart of the CTOS-DE scheme

TABLE I: Simulation parameters

| Parameter | Value |
|---|---|
| Computation task size | 300 ∼ 1500 KB |
| Required CPU cycles | (1 ∼ 3)×$10^8$ cycles |
| Maximum tolerance latency | 0.1 ∼ 3 s |
| CPU frequency of sink node $f_i^{\text{sink}}$ | 0.3 GHz |
| CPU frequency of MES $f_m^{\text{MES}}$ | 20 GHz |
| CPU frequency of RCS $f^{\text{RCS}}$ | 50 GHz |
| Waiting power of sink node $P_i^W$ | 100 mW |
| Transmission power of sink node $P_i^T$ | 200 mW |
| Execution power of sink node $P_i^E$ | 500 mW |

We compare the proposed CTOS-DE scheme with two reference algorithms: (1+1) Evolution Strategy ((1+1)ES) [15] and Genetic Algorithm (GA) [12] which are main computation offloading algorithms based on the cooperation between MCC and MEC. Specifically, (1+1)ES is an evolution strategy where the offspring individuals evolve from one parent. GA is a meta-heuristic algorithm which optimizes the population by iterations.

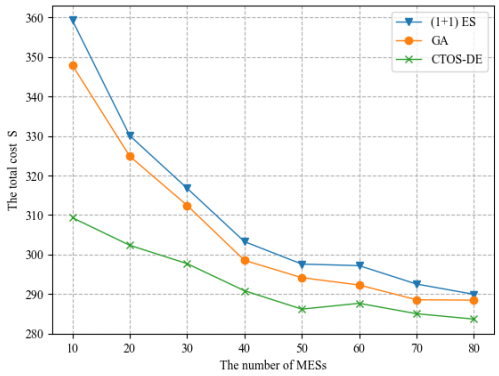


Fig. 3: The total cost $S$

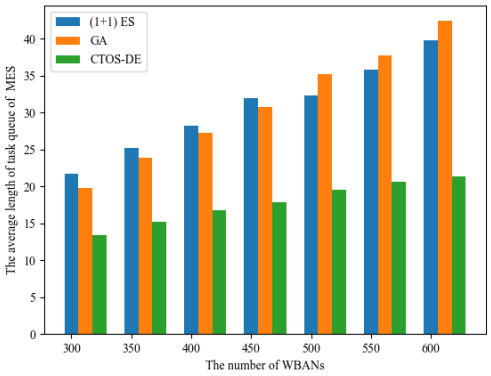


Fig. 4: The average length of task queue

### B. Simulations Results

We first assess the total cost with the increasing of the number of MESs when the urgent tasks occupy half of all tasks as shown in Fig.3. In detail, the weight factor of the delay $\omega$ is set to 0.8 and that of the energy consumption $\varphi$ is 0.2 for urgent tasks. The corresponding factors for non-urgent task are 0.2 and 0.8, respectively. Obviously, Fig.3 shows that our proposed CTOS-DE scheme performs best in terms of the total cost among the three algorithms. Moreover, the superiority of our scheme is outstanding when the number of MES is small. Specifically, the total cost of the CTOS-DE is reduced by 11.2% and 13.9% than that of GA and (1+1) ES when the number of MES is equal to 10, respectively.

Fig.4 illustrates the average length of task queue of MES against the increasing of the number of WBANs. It is shown that our CTOS-DE scheme can achieve the shortest average length of task queue among the three algorithms. A quantitative comparison can be made that the length of task queue is only increased by 58.8% when the number of WBANs increases from 300 to 600. However, the values of (1+1) ES and GA schemes enlarge to 83.7% and 122.7% respectively. It further demonstrates the superiority of our CTOS-DE scheme in terms of the load balancing among all MESs.

According to the above analysis, we can draw a conclusion that our CTOS-DE scheme can provide a better computation task offloading decision than that of (1+1) ES and GA algorithms, in terms of the total cost and load balancing among all MESs.

## VI. Conclusion

In this paper, we have studied a computation task offloading scheme for WBANs. First, we introduced a three-tier framework with one RCS, several MESs and multiple WBANs. Then, an optimization problem with the objective to minimize the total cost formulated. Subsequently, we proposed a computation task offloading scheme based on the differential evolution, denoted by CTOS-DE, to solve the problem. Finally, the simulation results demonstrated that our CTOS-DE scheme performed better than that of (1+1) ES and GA algorithms in terms of the total cost and load balancing.

## Acknowledgment

This work is supported by the National Natural Science Foundation of China (No.62171296) and the Science and Technology Project of Beijing Municipal Education Commission (No.KM202010028005).

## References

[1] S. Sodagari, B. Bozorgchami, and H. Aghvami, "Technologies and challenges for cognitive radio enabled medical wireless body area networks," *IEEE Access*, vol. 6, pp. 29567–29586, 2018.

[2] D. Sabella and A. Vaillant, "Mobile-edge computing architecture: The role of mec in the internet of things," *IEEE Consumer Electronics Magazine*, vol. 5, no. 4, pp. 84–91, 2016.

[3] Z. Zhang and S. Li, "A survey of computational offloading in mobile cloud computing," in *2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, pp. 81–82, 2016.

[4] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[5] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2017.

[6] B. Yang, X. Cao, J. Bassey, X. Li, T. Kroecker, and L. Qian, "Computation offloading in multi-access edge computing networks: A multi-task learning approach," in *IEEE International Conference on Communications (ICC)*, pp. 1–6, 2019.

[7] K. Wang, Z. Hu, and Q. Ai, "Joint offloading and charge cost minimization in mobile edge computing," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 205–216, 2020.

[8] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[9] Z. Ning, P. Dong, and X. Kong, "A cooperative partial computation offloading scheme for mobile edge computing enabled IoT," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4804–4814, 2019.

[10] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Transactions on Networking*, vol. 26, pp. 1619–1632, 2018.

[11] Y. Liao, Y. Han, Q. Yu, Q. Ai, Q. Liu, and M. S. Leeson, "Wireless body area network mobility-aware task offloading scheme," *IEEE Access*, vol. 6, pp. 61366–61376, 2018.

[12] U. Saleem, Y. Liu, and S. Jangsher, "Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 360–374, 2021.

[13] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[14] R. Storn, "Differential evolution-a simple and efficient heuristic for global optimization over continuous space," *Journal of Global Optimization*, vol. 11, 1997.

[15] T. Glasmachers, "Global convergence of the (1+1) evolution strategy to a critical point," *Evol. Comput.*, vol. 28, p. 27C53, Mar. 2020.