A Computation Offloading Strategy for the Large-Scale Disaster Scenario

Bo Wang(✉), Xu Wang, and Dongyan Huang

Key Laboratory of Wideband Communication and Signal, Ministry of Education, Guilin University of Electronic Technology, Guilin 541004, China
glbluewind@126.com

Abstract. For the large-scale disaster scenario, this paper investigates a hybrid satellite-aerial-terrestrial network (HSATN) to provide computing services for ground user equipment (GUE). The optimization problem is modeled to maximize the quality of experience (QoE) for all GUEs. To solve this problem, a multi-user game computation offloading (MUGCO) strategy is proposed. Firstly, the Initialized offloading decision is given according to the priority of the tasks. Then, the cooperative game algorithm and the task priority-based resource competition (TPRC) algorithm are used to allocate subcarrier resources and computing resources for the offloaded tasks, respectively. Finally, the offloading decision problem is proved as a potential game process. According to the properties of the potential game, the offloading decision under the Nash equilibrium is found. The simulation results show that the proposed strategy can effectively improve the total QoE value of the system compared to the comparison strategies.

Keywords: Mobile edge computing · Offloading decisions · Resource allocation

1 Introduction

In disasters, such as floods and earthquakes, wars, communication infrastructure is destroyed and related services are disrupted. To meet the communication needs of rescue activities and the affected people, a reliable and resilient wireless communication system for disaster relief must be constructed.

However, small-scale and large-scale disasters base on the size of the area affected are classified [1]. Compared to the small-scale disasters, large-scale disasters are characterized by extensive damage, longer time to rebuild the ground network and a large number of populations affected. Therefore, a reliable and resilient communications system for large-scale disasters relief must meet three demands: large coverage, long endurance, and sufficient computing resources.

Mobile edge computing (MEC) is defined as the deployment of computing capabilities and services at the edge of the network [2]. This way of providing computing services at the edge can meet the technical and application requirements for fast connectivity, real-time analysis, and rapid response. Therefore, MEC is usually integrated into disasters relief networks to provide computing services.

J. Wang (Ed.): MobiMedia 2023, LNICST 576, pp. 210–224, 2025.
https://doi.org/10.1007/978-3-031-60347-1_18

High-altitude platform (HAP) can carry heavy 5G infrastructure, offering longer endurance and greater area coverage[3]. However, too many tasks offloaded to HAP with limited computing resources may result in a degraded QoE for GUE. Therefore, a collaborates low earth orbit (LEO) satellites with HAP to enhance the computing capabilities in large-scale disaster relief network is proposed[4]. LEO satellites not only can provide edge computing services, but also can obtain remote cloud services through backhaul links.

Currently, the emerging satellite-aerial-terrestrial integrated networks like this one have become a research hotspot. In [5], Zhang et al. proposed a SAIC architecture that provides computing services for users in disaster areas and maximizes the system's sum rate through a 3D hypergraph matching algorithm. In [6], a SAIECN architecture is proposed to provide computing services for GUE at the edge side of LEO satellites and HAP. It minimizes the energy consumption in the system by jointly optimizing the resource allocation and offloading decisions in the system. They mainly consider a two-tier structure with HAP and LEO satellite collaboration, but both ignore the auxiliary role of the cloud data center in the computation offloading process.

The remaining work in this paper is as follows. In Sect. 2, the optimization problem model is proposed based on the HSATN. In Sect. 3, a MUGCO strategy is proposed to find an offloading decision scheme based on Nash equilibrium. In Sect. 4, the simulation analysis of the proposed strategy is performed. In Sect. 5, the work of this paper is concluded.

2 System Model and Problem Formulation

The HSATN based on a large-scale disaster relief scenario is shown in Fig. 1.

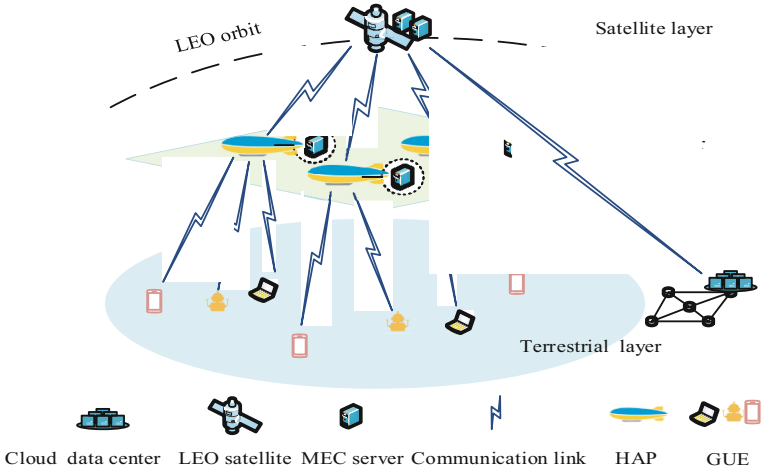

Fig. 1. HSATN system model.

In the aerial layer, I HAPs with MEC servers, denoted by set $\mathcal{I} = \{1, 2, ..., I\}$, are deployed to process the tasks running in GUE. Each HAP is deployed at altitude of L_H

above the disaster area and floats in a quasi-static manner over the stratosphere [7]. It can process tasks offloaded by GUE as well as act as a relay node to communicate with LEO satellite.

In the satellite layer, a satellite equipped with an enhanced MEC server is deployed at orbital altitude of L_s. It can either process offloaded tasks from GUE or act as a relay node to forward tasks to the cloud data center for processing.

In the terrestrial layer, GUEs are randomly distributed. Assume that each HAP serves J denoted by set $\mathcal{J} \ominus = \{1, 2, ...J\}$. And each GUE has a data task to process. Denote by $task_{n,m}$ the task generated by the m^{th} GUE under the n^{th} HAP, where $n \in \mathcal{I}, m \in \mathcal{J}$. The attributes of $task_{n,m}$ are represented by the array $< d_{n,m}, c_{n,m}, T_{n,m}^{\max} >$, where $d_{n,m}$ denotes the size of the task data volume, $c_{n,m}$ denotes the number of CPU cycles required to process the task, and $T_{n,m}^{\max}$ denotes the maximum tolerated delay of the task.

2.1 Communication Model

The tasks that need to be offloaded are transmitted to the HAP via non-orthogonal multiple access (NOMA) technology. It is assumed that there are k subcarriers in the system and each HAP have the full channel state information of these k subcarriers. The tasks requiring transmission are divided into k groups, and the GUEs in each group can share the same subcarrier. Denote these k groups by the set $S = \{s_1, s_2, ...s_k\}$, where s_k denotes the group of tasks transmitted via subcarrier k. And the number of tasks in this group is Q, denoted by the set Γ= [1,.q..Q]. In addition, denote by $g_{i,j}^{kq}$ the channel gain of the task q transmitted via subcarrier k, where $i \in \mathcal{I}, j \in \mathcal{J}$. And the $g_{i,j}^{kq}$ is formulated as

$$g_{i,j}^{kq} = \eta_0 d_{i,j}^{-\tau} = \frac{\eta_0}{\left(L_H^2 + X_{i,j}^2\right)^{0.5\tau}} \tag{1}$$

where $X_{i,j}$ is the horizontal distance between HAP i and GUE j, η_0 is the channel gain at a reference distance of 1 m, and τ is the path loss index between GUE and HAP. Without loss of generality, it is assumed that the channel gain of the task transmitted via subcarrier k satisfies $g_{i,j}^{k1} \geq ... \geq g_{i,j}^{kQ}$. According to the definition of NOMA technology, the transmission rate of $task_{n,m}$ via subcarrier k is [8]

$$R_{n,m}^{h} = B_k \log\left(1 + \frac{p_{n,m}^{kq} g_{n,m}^{kq}}{\sum_{n=1}^{I}\sum_{m=1}^{J} v_{n,m}^{k} \sum_{a=q+1}^{Q} p_{n,m}^{ka} g_{n,m}^{ka} + \sigma^2}\right) \tag{2}$$

where $p_{n,m}^{kq}$ is the transmission power of task q, σ^2 is the background noise, B_k is the bandwidth of the subcarrier, $v_{n,m}^{k}$ is the subcarrier selection factor. If $task_{n,m}$ selects subcarrier k, then $v_{n,m}^{k} = 1$, and vice versa. For the convenience of analysis, the transmission rate from HAP to LEO satellite is set as $R_{n,m}^{s}$.

2.2 Computation Model

Under the HSATN, there are three offloading modes for tasks. Define $o_{n,m} \in \{0, 1, 2, 3\}$ as the offloading decision factor for $task_{n,m}$.

If $o_{n,m} = 0$, it means that the task is processed locally. The processing delay is $T^l_{n,m} = \frac{c_{n,m}}{f^l_{n,m}}$, where $f^l_{n,m}$ is the computing capacity of GUE. The processing energy consumption is $E^l_{n,m} = \kappa(f^l_{n,m})^2 c_{n,m}$,where κ is a constant related to the CPU architecture.

If $o_{n,m} = 1$, it means that the task is offloaded to the HAP for processing. The processing delay is $T^h_{n,m} = \frac{d_{n,m}}{R^h_{n,m}} + \frac{c_{n,m}}{f^h_{n,m}}$, where $f^h_{n,m}$ is the computing resource allocated to $task_{n,m}$ by the MEC server on HAP. The processing energy consumption is $E^h_{n,m} = p_{n,m}\frac{d_{n,m}}{R^h_{n,m}}$.

If $o_{n,m} = 2$, it means that the task is offloaded to the satellite for processing. The processing delay is $T^s_{n,m} = \frac{2L_{n,s}}{c} + \frac{d_{n,m}}{R^h_{n,m}} + \frac{d_{n,m}}{R^s_{n,m}} + \frac{c_{n,m}}{f^s_{n,m}}$, where $f^s_{n,m}$ is the computing resource allocated to $task_{n,m}$ by the MEC server on LEO satellite,$L_{n,s}$ is the distance from HAP to the LEO satellite, and c is the speed of light. The processing energy consumption is $E^s_{n,m} = E^h_{n,m}$.

If $o_{n,m} = 3$, it means that the task is offloaded to the cloud data center for processing. The processing delay is $T^c_{n,m} = \frac{2L_{n,s}}{\mathrm{c}} + \frac{d_{n,m}}{R^h_{n,m}} + \frac{d_{n,m}}{R^s_{n,m}} + T^c_{wait}$, where T^c_{wait} denotes the waiting delay of the task sent from the LEO satellite to the cloud data center for processing[9]. The processing energy consumption is $E^c_{n,m} = E^h_{n,m}$.

Therefore, the delay and energy consumption after $task_{n,m}$ scheduling can be expressed by

$$T_{n,m} = T^l_{n,m} Z_{\{o_{n,m}=0\}} + T^h_{n,m} Z_{\{o_{n,m}=1\}} + T^s_{n,m} Z_{\{o_{n,m}=2\}} + T^c_{n,m} Z_{\{o_{n,m}=3\}} \tag{3}$$

$$E_{n,m} = E^l_{n,m} Z_{\{o_{n,m}=0\}} + E^h_{n,m} Z_{\{o_{n,m}=1\}} + E^s_{n,m} Z_{\{o_{n,m}=2\}} + E^c_{n,m} Z_{\{o_{n,m}=3\}} \tag{4}$$

where $Z_{\{*\}}$ means that if equation $*$ holds, then $Z_{\{*\}} = 1$, otherwise $Z_{\{*\}} = 0$.

2.3 Problem Formulation

Based on the above, the QoE function after $task_{n,m}$ scheduling is defined to be

$$U_{n,m} = \beta^t_{n,m} \frac{T^l_{n,m} - T_{n,m}}{T^l_{n,m}} + \beta^e_{n,m} \frac{E^l_{n,m} - E_{n,m}}{E^l_{n,m}} \tag{5}$$

where $\beta^t_{n,m} \in [0, 1][0, 1]$ and $\beta^e_{n,m} = 1 - \beta^t_{n,m}$, denote the weights of delay and energy consumption, respectively. The objective of this paper is to maximize the value of the

QoE function for all GUEs, as shown in Eq. (6).

$$\begin{aligned}
&\max_{F,S,W} \sum_{n=1}^{I} \sum_{m=1}^{J} U_{n,m} \\
&s.t. C1: T_{n,m} \leq T_{n,m}^{\max}, \forall n \in \mathcal{I}, \forall m \in \mathcal{J} \\
&C2: o_{n,m} \in \{0, 1, 2, 3\}, n \in \mathcal{I}, \forall m \in \mathcal{J} \\
&C3: v_{n,m}^{k} \in \{0, 1\}, \forall n \in \mathcal{I}, \forall m \in \mathcal{J} \\
&C4: \sum_{m=1}^{J} Z_{\{o_{n,m}=1\}} f_{n,m}^{h} \leq f_{n}^{h}, \forall n \in \mathcal{I}, \forall m \in \mathcal{J} \\
&C5: \sum_{n=1}^{I} \sum_{m=1}^{J} Z_{\{o_{n,m}=2\}} f_{n,m}^{s} \leq f^{s}, \forall n \in \mathcal{I}, \forall m \in \mathcal{J}
\end{aligned} \tag{6}$$

where F is computing resource allocation scheme,S is subcarrier resource allocation scheme,W is offloading decision scheme. The constraints $C1$ is the maximum tolerated delay of the task, $C2$ is the offloading decision selection space of the task, $C3$ is the subcarrier decision factor of the task, $C4 \sim C5$ is that the sum of computing resources allocated to the offloading task cannot exceed the total computing resources of the corresponding MEC server.

3 The Proposed Strategy

The optimization variables in problem (6) have both integer and continuous variables, so the problem is a mixed-integer nonlinear programming problem. To solve this problem, a MUGCO strategy is proposed. To begin with, the offloading decision is initialized according to the priority of the task. Following that, the cooperative game and TPRC algorithms are proposed to allocate subcarrier resources and computing resources for the task. At last, the offloading decision is confirmed to be a potential game process by proving that the optimization function is a potential function. And the offloading decision under Nash equilibrium is found according to the finite improvement property of the potential game.

3.1 Flowchart of the MUGCO Strategy

The offloading decision scheme under the Nash equilibrium is obtained by the proposed MUGCO strategy. The strategy flowchart is shown in Fig. 2.

The specific steps are as follows:

Step one: Initialize offloading decision scheme based on task priority.

Step two: The cooperative game algorithm and TPRC algorithm are used to allocate subcarrier resources and computing resources for the task, respectively.

Step three: Entering the iterative process of potential game, the GUEs finds the best response position by Eq. (16).

Step four: The offloading decision solution under Nash equilibrium is obtained when each user reaches the best response position.

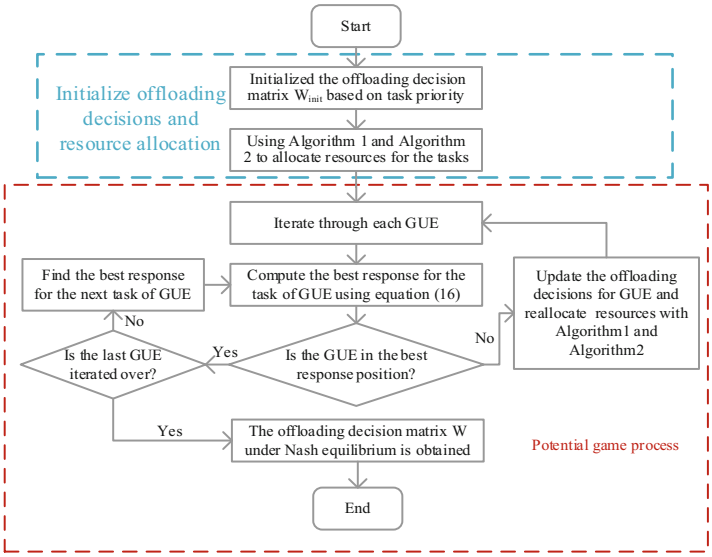

Fig. 2. Flowchart of the MUGCO strategy.

3.2 Initialize Offloading Decisions and Resource Allocation

Initialize Offloading Decisions In the disaster scenario, different tasks have different service requirements. Therefore, this paper constructs the priority score based on two metrics: the maximum task delay constraint and the number of CPU cycles required to process the task. And the mathematical model of the priority score of $task_{n,m}$ can be expressed as

$$\text{score}_{n,m}=\lambda_{n,m}\frac{\max\limits_{n\in I,m\in J}(T_{n,m}^{\max})-T_{n,m}^{\max}}{\max\limits_{n\in I,m\in J}(T_{n,m}^{\max})-\min\limits_{n\in I,m\in J}(T_{n,m}^{\max})}+(1-\lambda_{n,m})\frac{\max\limits_{n\in I,m\in J}(c_{n,m}^{\max})-c_{n,m}^{\max}}{\max\limits_{n\in I,m\in J}(c_{n,m}^{\max})-\min\limits_{n\in I,m\in J}(c_{n,m}^{\max})} \quad (7)$$

where $0 \leq \lambda_{n,m} \leq 1$ is the weight of the two indicators and can be obtained by the entropy method. After getting the rating of each task, the sorting algorithm is used to classify them into four levels from highest to lowest and store them in the matrix $W_{init} = [A_1, A_2, A_3, A_4]$. Then the initialized offloading scheme for $task_{n,m}$ can be obtained as: when $score_{n,m} \in A_1, o_{n,m} = 1$; when $score_{n,m} \in A_2, o_{n,m} = 2$; when $score_{n,m} \in A_3, o_{n,m} = 3$; when $score_{n,m} \in A_4, o_{n,m} = 0$.

Resource allocation The cooperative game algorithm is used to optimize the allocation of subcarrier resources. And the game model is modeled as . Among them, is the set of offloading tasks, is the grouping set of subcarriers, and is the Utility function of the subcarriers. The utility function in the subcarrier can be calculated as: ,Where is the Transmission rate of task through subcarrier . The process of cooperative gaming will be described in detail below.

Definition 1:(split)$\{s_\iota, s_k\} \rightarrow \{s_\iota \cup q, s_k \backslash \{q\}\}$ denotes the task q in grouping s_k leaves and joins grouping s_ι, where $s_\iota \in S$, and $s_\iota \neq s_k$.

Definition 2: $\succ_q$ denotes the preference of task q for grouping. For example, the preference of task q for group s_t in group s_k can be expressed as: $s_\iota \succ_q s_k \Leftrightarrow G_{s_k \backslash \{q\}} + G_{s_\iota \cup \{q\}} > G_{s_k} + G_{s_\iota}$.

Definition 3: (exchange)If task q in group s_k and task q' in group s_ι satisfy Definition 2, they generate an exchange behavior and update the group as: $\{s_\iota, s_k\} \rightarrow \{s_\iota \cup \{q\} \backslash \{q'\}, s_k \cup \{q'\} \backslash \{q\}\}$.

Algorithm 1: Cooperative Game Algorithm

Input: channel gain, offloading decision matrix, random grouping set S_{start}

1.repeat the iteration

2.randomly select task q , q' from the group $s_k \in S_{strat}$, $s_t \in S_{start}$

3. *if* swap tasks to form a new group S_{new}

4. *if* $S_{new} \succ_{q'} S_{start}$

5. update $S_{start} \leftarrow \{S_{start} \setminus \{s_t, s_k\}\} \cup \{s_t \cup \{q\} \setminus \{q'\}, s_k \cup \{q'\} \setminus \{q\}\}$

6. *else* suppose that task q is split and added to s_t to form a new group S_{new}

7. *if* $S_{new} \succ_{q'} S_{start}$

8. update $S_{start} \leftarrow \{S_{start} \setminus \{s_t, s_k\}\} \cup \{s_t \cup \{q\}, s_k \setminus \{q\}\}$

9. *Utile* convergence on the final grouping set S_{end}

Output : S_{end}

The initialization subcarrier is randomly assigned. After entering the iteration of the cooperative game, the subcarriers different from the previous subcarriers are randomly selected for the grouping operation in the above definition. If the grouping reaches stability, the NOMA system has obtained the best grouping. The specific steps of the cooperative game are shown in Algorithm 1.

Lemma 1: Algorithm 1 can converges to the grouping set S_{end} in a finite step.

Proof: To optimize the utility function value G, the tasks in the grouping are constantly split and swapped**.** In the kth to k + 1th iteration, if the grouping set is updated from S_k to S_{k+1},the utility function G in the game must strictly increase, i.e., $S_k \rightarrow S_{k+1} \Leftrightarrow G_{S_{k+1}} > G_{S_k}$.So the utility function G is always increasing in the game iteration, i.e.,$S_{start} \rightarrow S_1 \rightarrow ... \rightarrow S_{end}$. Since the number of GUE for transmission is finite, the grouping set is also finite and based on the Bell number [10]. Therefore, as the game iterates incrementally, the set of groupings will converge to S_{end}.

Algorithm2 : TPRC algorithm

Input: offloading decision matrix, task priority, MEC servers computing resources
1.initialize the computing resource allocation matrix $F = 0$
2. sort the tasks that are pre-offloaded to the MEC server in order of priority from highest to lowest according to the offloading decision matrix
3. according to constraints C6 and C7, the minimum computing resources required for the task are allocated in turn.
4. if the task can be allocated the minimum resources by the MEC server, the offloading request is accepted, otherwise it is rejected
5. let $F_n \in \{f_1^h, ..., f_I^h, f_{I+1}^s\}$, where n denotes the number of the MEC server
6. *for* $n = 1 : I + 1$
7. *while* $F_n > 0$
8. splitting F_n into blocks of resources in units of f^l
9. *for* $m = 1 : J$
10. *if* $task_{n,m}$ are offloaded at server n
11. temporarily append computing resources f^l to the task and calculate $T_{n,m}^{'}$
12. calculate $M(1,m) = T_{n,m}^{'} - T_{n,m}$
13. *end if*
14. *end for*
15. $index = \max(M)$, update $f_{n,index} = f_{n,index} + f^l, F_n = F_n - f^l$
16. *end while*
17. *end for*
18. Output: allocation matrix of computing resources F

For the problem of allocating computing resources to MEC servers, the mathematical model can be derived from Eq. (6) as

$$\min_{F} \sum_{n=1}^{I} \sum_{m=1}^{J} Z_{\{o_{n,m}=1\}} \left(\frac{d_{n,m}}{R_{n,m}^h} + \frac{c_{n,m}}{f_{n,m}^h} \right) + Z_{\{o_{n,m}=2\}} \left(\frac{2L_{n,s}}{\mathrm{c}} + \frac{d_{n,m}}{R_{n,m}^h} + \frac{d_{n,m}}{R_{n,m}^s} + \frac{c_{n,m}}{f_{n,m}^s} \right)$$

$$s.tC4C5$$

$$C6 : f_{n,m}^h \geq \frac{c_{n,m} R_{n,m}^h}{T_{n,m}^{\max} R_{n,m}^h - d_{n,m}} = \left(f_{n,m}^h \right)_{\min}, \forall n \in \mathcal{I}, \forall m \in \mathcal{J}$$

$$C7 : f_{n,m}^s \geq \frac{c_{n,m} R_{n,m}^h R_{n,m}^s c}{R_{n,m}^h R_{n,m}^s \left(T_{n,m}^{\max} c - 2L_s \right) + c d_{n,m} \left(R_{n,m}^h + R_{n,m}^s \right)} = \left(f_{n,m}^s \right)_{\min}, \forall n \in \mathcal{I}, \forall m \in \mathcal{J} \tag{8}$$

The constraints $C6 \sim C7$ is the minimum computing resources required to offload the task to the corresponding MEC server.

Due to the different task service requirements in the disaster area, a TPRC algorithm is designed based on problem (6). First, the tasks that offloaded to the MEC servers are sorted from highest to lowest priority. Then, according to the constraints C6 and C7, the

minimum computing resources satisfying the maximum delay requirement are allocated to the tasks in turn. Finally, if the MEC server still has idle computing resources, these computing resources are divided into several resource blocks. And the offloaded tasks maximize the total QoE value of the system by competing for resource blocks. The specific steps are shown in Algorithm 2.

3.3 Potential Game Process

The offloading decision problem is first modeled as a game $\psi = \{Z,\ W,\ H_{n,m}\}$, Where denotes the set of GUEs $W = \{(o_{n,m}) : o_{n,m} \in \{0, 1, 2, 3\}, n \in I,\ m \in J\}$, denotes the choice strategy space of each GUE, $H_{n,m}$ and denotes the QoE function of $task_{n,m}$. Then the game is proved as a potential game. The details are presented as follows.

Definition 4: A game Ψ is said to be a potential game if there exists a function χ satisfying the following conditions[11].

$$\begin{aligned} &H_{n,m}\left(o_{n,m}, o_{-(n,m)}\right) - H_{n,m}\left(o_{n,m\prime}, o_{-(n,m)}\right) > 0 \Leftrightarrow \\ &\chi\left(o_{n,m}, o_{-(n,m)}\right) - \chi\left(o_{n,m\prime}, o_{-(n,m)}\right) > 0 \qquad n \in I, m \in J \end{aligned} \tag{9}$$

Lemma 2: The potential function in the game Ψ is.

$$\chi(W) = \sum_{n=1}^{I}\sum_{m=1}^{J} U_{n,m} \tag{10}$$

Proof: Case 1: If the offloading decision of $task_{n,m}$ is updated from $o_{n,m} = 0$ to $o_{n,m\prime} > 0$, some of the tasks will suffer from degradation of their transmission and computing performance because $task_{n,m}$ occupies subcarriers and computing resources. Denote by $task_{x,y}$ the affected tasks. If $H_{n,m}(o_{n,m}, o_{-(n,m)}) - H_{n,m}(o_{n,m\prime}, o_{-(n,m)}) < 0$, then it can be inferred that

$$\begin{aligned} &\frac{T^l_{n,m} - T_{n,m\prime}}{T^l_{n,m}} - \sum_x\sum_y \frac{T_{x,y\prime} - T_{x,y}}{T^l_{x,y}} > 0 \\ &\frac{E^l_{n,m} - E_{n,m\prime}}{E^l_{n,m}} - \sum_x\sum_y \frac{E_{x,y\prime} - E_{x,y}}{E^l_{x,y}} > 0 \end{aligned} \tag{11}$$

where $T_{x,y}$ and $E_{x,y}$ denote the delay and energy consumption of $task_{x,y}$ before $task_{n,m}$ updates the offloading decision;$T_{x,y\prime}$ and $E_{x,y\prime}$ denote the delay and energy consumption of $task_{x,y}$ after $task_{n,m}$ updates the offloading decision;$T_{n,m\prime}$ and $E_{n,m\prime}$ denote the delay and energy consumption after $task_{n,m}$ updates the offloading decision.

$$\begin{aligned} &\chi(W) - \chi(W\prime) = H_{n,m}(W) - H_{n,m}(W\prime) + \sum_x\sum_y \left[H_{x,y}(W) - H_{x,y}(W\prime)\right] \\ &= \beta^t_{n,m}\left(\sum_x\sum_y \frac{T_{x,y\prime} - T_{x,y}}{T^l_{x,y}} + \frac{T_{n,m\prime} - T^l_{n,m}}{T^l_{n,m}}\right) + \beta^e_{n,m}\left(\sum_x\sum_y \frac{E_{n,m\prime} - E_{x,y}}{E^l_{x,y}} + \frac{E_{n,m\prime} - E^l_{n,m}}{E^l_{n,m}}\right) \end{aligned} \tag{12}$$

Let $\beta_{n,m}^{t} = \beta_{n,m}^{e} = \beta_{x,y}^{t} = \beta_{x,y}^{e}$, then the above equation leads to: $\chi(W) - \chi(W\prime) < 0$.

Case 2: If the offloading decision of $task_{n,m}$ is updated from $o_{n,m} > 0$ to $o_{n,m\prime} = 0$, some of the tasks will have their transmission and computing performance increased because $task_{n,m}$ releases the subcarriers and computing resources. Using $task_{x,y}$ to denote the affected task, it get $H_{x,y}(o_{x,y}, o_{-(x,y)}) < H_{x,y}(o_{x,y\prime}, o_{-(x,y)})$.If $H_{n,m}\left(o_{n,m}, o_{-(n,m)}\right) - H_{n,m}\left(o_{n,m\prime}, o_{-(n,m)}\right) < 0$, then it can be introduced that

$$\chi(W) - \chi(W\prime) = H_{n,m}(W) - H_{n,m}(W\prime) + \sum_{x}\sum_{y}\left[H_{x,y}(W) - H_{x,y}(W\prime)\right]0 \quad (13)$$

Case 3: If the offloading decision is updated from $o_{n,m} > 0$ to $o_{n,m\prime} > 0$, but $o_{n,m} \neq o_{n,m\prime}$. There are three additional situation that can be discussed:

1)The offloading decision of $task_{n,m}$ is updated from $0 < o_{n,m} < 3$ to $0 < o_{n,m\prime} < 3$. Suppose it is updated from $o_{n,m} = 1$ to $o_{n,m\prime} = 2$. This update releases computing resources on the HAP and competes for computing resources with tasks offloaded on LEO satellites. Here, $task_{x_1,y_1}$ and $task_{x_2,y_2}$ are used to denote the tasks that are affected by the release and occupation of computing resources by $task_{n,m}$, respectively. If $H_{n,m}(o_{n,m}, o_{-(n,m)}) - H_{n,m}(o_{n,m\prime}, o_{-(n,m)}) < 0$, then it can be obtained

$$\frac{2L_s}{c} + \frac{d_{n,m}}{R_{n,m}^{s}} + \frac{c_{n,m}}{f_{n,m}^{s}} - \frac{c_{n,m}}{f_{n,m}^{h}} + \sum_{x1}\sum_{y1} f^{l}\left(\frac{f_{x1,y1\prime} - f_{x1,y1}}{f_{x1,y1}f_{x1,y1\prime}}\right) + \sum_{x2}\sum_{y2} f^{l}\left(\frac{f_{x2,y2\prime} - f_{x2,y2}}{f_{x2,y2}f_{x2,y2\prime}}\right) > 0 \quad (14)$$

where f_{x_1,y_1} and f_{x_2,y_2} denote the computing resources allocated to $task_{x_1,y_1}$ and $task_{x_2,y_2}$ before affected; $f_{x_1,y_1\prime}$ and $f_{x_2,y_2\prime}$ denote the computing resources allocated to $task_{x_1,y_1}$ and $task_{x_2,y_2}$ after affected.

$$\begin{aligned}
&\chi(W) - \chi(W\prime) = H_{n,m}(W) - H_{n,m}(W\prime) + \sum_{x_1}\sum_{y_1}\left[H_{x_1,y_1}(W) - H_{x_1,y_1}(W\prime)\right] + \\
&\sum_{x_2}\sum_{y_2}\left[H_{x_2,y_2}(W) - H_{x_2,y_2}(W\prime)\right] < 0 \\
&= H_{n,m}(W) - H_{n,m}(W\prime) + \sum_{x_1}\sum_{y_1}\frac{T_{x_1,y_1\prime} - T_{x_1,y_1}}{T_{x_1,y_1}^{l}} + \sum_{x_2}\sum_{y_2}\frac{T_{x_2,y_2\prime} - T_{x_2,y_2}}{T_{x_2,y_2}^{l}} \\
&= \beta_{n,m}^{t}\left[\left(\frac{c_{n,m}}{f_{n,m}^{h}} - \frac{2L_s}{c} - \frac{d_{n,m}}{R_{n,m}^{s}} - \frac{c_{n,m}}{f_{n,m}^{s}}\right) + \sum_{x_1}\sum_{y_1} f^{l}\left(\frac{f^{x_1,y_1} - f_{x_1,y_1\prime}}{f_{x_1,y_1}f_{x_1,y_1\prime}}\right)\right. \\
&\left. + \sum_{x_2}\sum_{y_2} f^{l}\left(\frac{f_{x_2,y_2} - f_{x_2,y_2\prime}}{f_{x_2,y_2}f_{x_2,y_2\prime}}\right)\right] < 0
\end{aligned} \quad (15)$$

where T_{x_1,y_1} and $T_{x_1,y_1\prime}$ denote processing delays before and after the affected task releases computing resources, respectively;T_{x_2,y_2} and $T_{x_2,y_2\prime}$ denote processing delays before and after the affected task occupy computing resources, respectively.

2) The offloading decision of $task_{n,m}$ is updated from $o_{n,m} = 3$ to $0 < o_{n,m\prime} < 3$. This update will cause the computing performance of some tasks to be degraded because it occupies the computing resources of the MEC server. The proof is similar to Case1.

3) The offloading decision of $task_{n,m}$ is updated from $0 < o_{n,m} < 3$ to $o_{n,m'} = 3$. This update released the computing resources of the MEC server, which makes computing performance of some tasks improve. The proof similar to Case 2.

In summary, it is proved that the function $\chi(W)$ is the potential function of the game Ψ. Therefore, the game Ψ is a potential game. According to the finite improvement property of potential games, the game Ψ can find the Nash equilibrium of the system after a finite number of iterations [12]. Based on this property, a MUGCO strategy is proposed to find the Nash equilibrium of the system. The core idea of the MUGCO strategy is that no more than one GUE updates the offloading decision at a time, and the update rule of GUE is it can find the best response. If the best response is found for all GUEs, the offloading decision scheme in Nash equilibrium is obtained. And the best response for $task_{n,m}$ is calculated as

$$\Theta_{n,m} \triangleq \left\{ o^*_{n,m} : o^*_{n,m} = \arg\max_{o_{n,m}\in\{0,1,2,3\}} U_{n,m}\left(o_{n,m}, o_{-(n,m)}\right) \right. \\ \left. and \;\; U_{n,m}\left(o_{n,m}, o_{-(n,m)}\right) < U_{n,m}\left(o^*_{n,m}, o_{-(n,m)}\right) \right\} \tag{16}$$

4 Simulation Results and Analysis

4.1 Simulation Parameters

In the HSATN, GUE is randomly distributed in a circular area with a diameter of 100 km. Ten HAPs and one LEO satellite are deployed at altitude $L_H = 20km$ and orbital altitude $L_s = 500km$ above the disaster area, respectively. The computing resources of the MEC servers deployed on LEO satellites and HAP are 20 GHz and 8 GHz, respectively. The rest of the simulation parameters are shown in Table 1 [6, 7, 13].

4.2 Analysis of Simulation Results

The average transmission rate of GUEs versus the number of GUEs for different algorithms is shown in Fig. 3.

The average transmission rate obtained by the offloading task decreases continuously as the number of GUEs increases. This is due to the increase in subcarrier multiplexing frequency, which leads to enhanced channel interference to the GUEs during transmission.

Compare with random (R-NOMA) allocation scheme and orthogonal multiple access (OMA) scheme, the cooperative game (CG-NOMA) scheme gets the highest average rate for the GUE. This is because the cooperative game algorithm can obtain the subcarrier assignment scheme with the least channel interference in the system by swapping or splitting tasks in the subcarriers.

Figure 4 depicts the QoE values obtained by the MEC servers for different computing resource allocation algorithms when $J = 17$. The number 1 to 10 of MEC servers belong to HAP and number 11 belongs to LEO satellite.

Table 1. Simulation parameters

Parameter	Value
Transmission power of GUE	300 mW
Computing power of GUE	0.3 GHz
The size of the data volume of the task	100 kb–1000 kb
Number of cycles required for the task	200–2000 Megacycles
Maximum tolerated delay of the task	1–3 s
Subcarrier bandwidth	6 MHz
Number of subcarriers	60
Background noise power	− 110 dbm
Effective capacitance constant	10^{-26}
Channel fading factor	2–3
Transmission rate from HAP to LEO satellite	10 Mbps

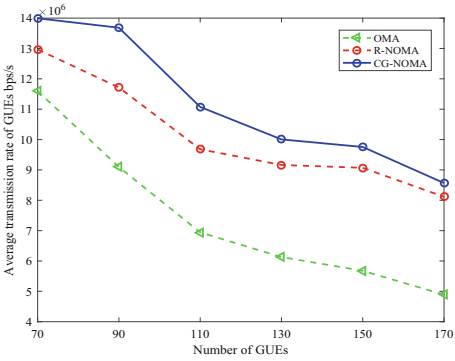

Fig. 3. Average transmission rate of GUEs vs.the number of GUEs.

Compared with the convex optimization-based computing resource allocation (COCRA) algorithm[14] and minimal computing resource allocation (MCRA) algorithm, the TPRC algorithm gets the highest QoE value. This is because the TPRC algorithm maximizes the use of the remaining computing resources in the MEC server.

Figure 5 depicts the total QoE value of the system versus the number of GUEs for different offloading strategies. When the number of GUEs is small, the offloaded tasks can get higher transmission rate and more computing resources, which makes the total QoE value of the system grow rapidly. However, as the number of GUEs increases, the growth of the total QoE value of the system slows down. The reason is the decrease in the transmission rate of the tasks and the intense competition for computing resources, which leads to a decrease in the QoE value obtained by the offloaded tasks.

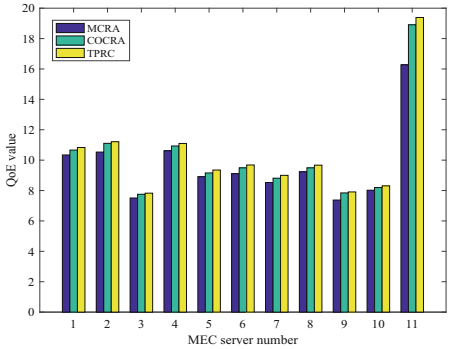

Fig. 4. QoE values of MEC servers for different algorithms.

Compared to the greedy offloading (GO) strategy, random offloading (RO) strategy and task all offloading to HAP (All-to-HAP) strategy, the MUGCO strategy obtains the highest system QoE value because it finds the Nash equilibrium in the system.

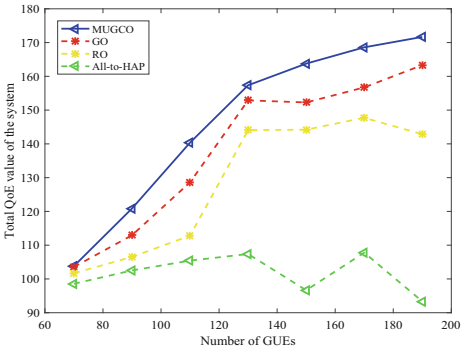

Fig. 5. Total QoE value of the system vs. the number of GUEs.

4.3 Complexity Analysis

The subcarrier resources use cooperative game algorithm with the time complexity is $O(K^2)$. The computing resource allocation uses the TPRC algorithm with the time complexity is $O(I \times J)$.Therefore, the time complexity of the MUGCO strategy is $O(K^2 \times I \times J)$. Compared with the greedy strategy, the MUGCO strategy has better performance. As shown in Table 2.

Table 2. Complexity Analysis

Strategies	MUGCO	GO
Time complexity	$O(K^2 \times I \times J)$	$O[4(K^2 \times I \times J)]$

5 Conclusion

This paper investigates the problem of computing offloading in the large-scale disaster scenario based on a HSATN. The optimization problem is modeled as maximizing the QoE value for all GUEs, which is solved by the proposed MUGCO strategy. The strategy first initializes the offloading decision by the priority of the task. Then, the subcarrier resources and computing resources are allocated for the task by the cooperative game and TPRC algorithm. Finally, it is proved that the offloading decision is a potential game process. Using the properties of the potential game, the offloading decision in Nash equilibrium is found by the MUGCO strategy. The results show that the proposed strategy can improve the QoE values of GUE.

Acknowledgments. This work was supported in part by the Director Fund of Key Laboratories of Cognitive Radio and Information Processing, Ministry of Education (Grant No.CRKL210109), in part by Innovation Project of GUET Graduate Education (Grant No.2022YCXS026).

References

1. Matracia, M., Saeed, N., Kishk, M.A.: Post-disaster communications: enabling technologies, architectures, and open challenges. IEEE Open J. Commun. Soc. **3**, 1177–1205 (2022)
2. Shah, Z. et al.: Mobile edge computing (MEC)-enabled UAV placement and computation efficiency maximization in disaster scenario. IEEE Trans. Veh. Technol.,1–12 (2023)
3. Kurt, G.K. et al.: A vision and framework for the high altitude platform station (HAPS) networks of the future. IEEE Commun. Surv. Tutor., 23(2), 729-779 (2021)
4. Huang, T., et al.: A survey on green 6g network: architecture and technologies. IEEE Access **7**, 175758–175768 (2019)
5. Zhang, L., Zhang, H., Guo, C., Xu, H., Song, L., Han, Z.: Satellite-aerial integrated computing in disasters: user association and offloading decision. In: ICC 2020 - 2020 IEEE International Conference on Communications (ICC).pp. 554–559. Dublin, Ireland (2020)
6. Ding, C., Wang, J.B., Zhang, H., Lin, M., Li, G.Y.: Joint optimization of transmission and computation resources for satellite and high altitude platform assisted edge computing, IEEE Trans. Wireless Commun., 21(2), 1362-1377 (2021)
7. D. S. Lakew, A.-T. Tran, N.-N. Dao and S. Cho.: Intelligent Offloading and Resource Allocation in HAP-Assisted MEC Networks.In: 2021 International Conference on Information and Communication Technology Convergence (ICTC). pp. 1582–1587.Jeju Island, Korea(2021)
8. Huynh, L. et al.: Joint computational offloading and data-content caching in NOMA-MEC networks. IEEE Access, **9**:12943–12954 (2021)
9. Rodrigues, T. K., Kato, N.: Deep Q networks with centralized learning over LEO satellite networks in a 6G cloud environment. In: GLOBECOM 2022-2022 IEEE Global Communications Conference (pp. 5905-5910). IEEE (2022).

10. Rota, G.C.: The number of partitions of a set. Am. Math. Mon. **71**(5), 498–504 (1964)
11. Wang, C., Dong, C., Qin, J., Yang, X., Wen, W.: In Energy-efficient Offloading Policy for Resource Allocation in Distributed Mobile Edge Computing. In: 2018 IEEE Symposium on Computers and Communications (ISCC). pp. 00366–00372. Natal, Brazil (2018)
12. Monderer, D., Shapley, L.S.: Potential games. Games Econom. Behav. **14**(1), 124–143 (1996)
13. Cheng, N., et al.: Space/aerial-assisted computing offloading for IoT applications: a learning-based approach. IEEE J. Sel. Areas Commun. **37**(5), 1117–1129 (2019)
14. Tran, T.X., Pompili, D.: Joint task offloading and resource allocation for multi server mobile-edge computing networks. IEEE Trans. Veh. Technol.Veh. Technol. **68**(1), 856–868 (2019)