Joint Offloading and Resource Allocation for Multi-User Multi-Edge Collaborative Computing System

Zihan Gao, Wanming Hao, *Member, IEEE*, and Shouyi Yang

***Abstract*—In this paper, we consider a computation offloading problem in a three-tier network consisting of multiple mobile users (MUs), multiple edge clouds, and a central cloud. On this basis, we formulate a system Energy Time Cost (ETC) minimization problem by jointly optimizing the resource allocation, mobile-edge matching decision, and offloading decision. Due to the difficulty of directly solving the formulated problem, we decompose it and propose two offloading algorithms, namely Gale-Shapley based Minimum/Sequential Offloading Algorithm (GS-MOA/GS-SOA), to optimize offloading decisions by minimizing the system ETC, in which the mobile-edge matching strategies based on the proposed Gale-Shapley based Mobile-Edge mAtching aLgorithm (GS-MEAL) and resource allocations are optimized iteratively. The simulations show the effectiveness of our proposed algorithm.**

***Index Terms*—Edge cloud computing, Gale-Shapley, three-tier network.**

I. Introduction

The rapid proliferation of Internet of Things (IoTs) users and the heterogeneous network architecture has spawned more emerging applications, such as smartphones, laptops, wearables, automotive devices, *etc* [1], [2]. Nevertheless, those applications are usually computation-intensive and energy-intensive, resulting in that they cannot be finished timely on mobile devices (MDs) due to the limited computation capability and battery power [3]. To deal with this, the mobile cloud computing (MCC) is first proposed to expand the MDs' computation capability by migrating part of or all computation applications to the cloud [4]. However, the cloud servers are usually located at the network center, which leads to significant transmission latency and extra transmission energy consumption. Thus, the mobile edge computing (MEC) is proposed as an alternative technology to better support delay-sensitive and computation-intensive mobile applications as well as overcome the drawbacks faced in the MCC system.

Recently, there has been several work considering resource optimization in the cloud computing system, while most of them only focus on MCC or MEC, such as [3]-[7]. Specifically, [3]-[4] considered the computation offloading problem in the MCC systems. Guo *et al.* [4] and Lyu *et al.* [5] both utilized the weighted sum of energy consumption and time consumption to jointly optimize the resource and offloading decisions. To minimize the latency, both M. Sheng *et al.* [6] and Cao *et al.* [7] optimized the offloading decisions by minimizing the system latency in the cloudlet-based network. To further enhance the cloud computing performance, J. Ren *et al.* [8] studied the fully offloading situation and proposed a collaborative computing system combining the MEC and MCC techniques. A. E. Alchalabi *et al.* [9] proposed a reinforcement learning models that aim to minimize the variance of latency in user-server matching for edge computing system.

Compared to the two-tier cloud network, a three-tier cloud network composed of the MEC, MCC, and MDs has higher flexibility, partially allocating workload to the edge cloud and central cloud to achieve better performance. Rahimi *et al.* [10] focused on optimizing offloading decisions without considering the resource allocations, and thus, the full benefit of offloading cannot be reached. H. Yu *et al.* [11] minimized the total cost by optimizing offloading decisions and resource allocation in MEC systems with the assistance of the central cloud. However, they only considered the single-user single-edge cloud scenario, which simplifies the problem analysis and system model, and the joint optimization of offloading decisions and resource allocation for a general three-tier multi-user multi-edge computation offloading system has not been investigated.

Unlike the existing work, we consider a computation offloading problem in a three-tier network consisting of multiple mobile users (MUs), multiple edge clouds, and a central cloud. The main contributions are summarized as follows.

- We characterize the ETC by energy consumption and computation time in local computing and collaborative cloud computing, respectively. Then, we formulat a ETC minimization problem by jointly optimizing the offloading decision, communication and computation resources and mobile-edge matching strategies under the hard constraints. Since the formulated problem is an MINLP and optimization variables are coupled with each other, we decompose the original one and propose the GS-MOA and the GS-SOA to solve it.
- Our proposed GS-MOA/GS-SOA runs in two stages. First, the transmission power and computation resource for each possible matching strategies under the current offloading decisions are optimized. Then, the mobile-edge matching strategies are optimized by proposed the GS-MEAL based on the current offloading decisions and optimized resource. Finally, the proposed GS-MOA/GS-SOA is used to find the offloading decisions by iteratively updates resource allocation and matching strategies to minimize system ETC.

II. System Model

We consider a system consisting of M MUs, denoted by $\mathcal{M} = \{1, 2, \ldots, M\}$, and N edge cloud servers denoted by $\mathcal{N} = \{1, 2, 3, \ldots, N\}$. Each MU can be served by one edge cloud, and each edge cloud can serve multiple MUs within its maximum computing capacity Cap_n, which can be obtained by pre-scheduler. We assume that an MU offloads a computation-intensive application to the edge cloud via the wireless channel. Different edge clouds transmit data to the central cloud through different backhaul links. Similar to the previous work [4], [5], we assume a quasi-static scenario, and the channel states during the offloading period are constant.

A. Computation Model

We adopt a two-filed notation $\mathcal{A}pplication_m = \{L_m, C_m\}$ to represent the computation-intensive application of MU m, where L_m (bit) is the input data size, including system settings, program codes,

Manuscript received July 5, 2021; revised November 6, 2021; accepted December 24, 2021. Date of publication December 31, 2021; date of current version March 15, 2022. This work was supported in part by the Natural Science Foundation of Henan Province, China under Grant 202300410482, in part by the National Natural Science Foundation of China under Grant 62101499, in part by the Young Elite Scientists Sponsorship Program of Henan Province under Grant 2022HYTP006, and in part by the Scientific and Technological Key Project of Henan Province under Grant 202102210119. The review of this article was coordinated by Prof. Mugen Peng. *(Corresponding author: Wanming Hao.)*

The authors are with the School of Information Engineering and Institute of Industrial Technology, Zhengzhou University, Zhengzhou 450001, China (e-mail: gaozihan zzu@foxmail.com; wmhao@hotmail.com; iesyyang@zzu.edu.cn).

Digital Object Identifier 10.1109/TVT.2021.3139843

and input parameters. C_m represents the required CPU cycles/bit to accomplish the application.

1) Local Computing: According to [4], [5], the time consumption and energy consumption of local computing of MU m can be represented as:

$$T_m^l = C_m L_m / f_m^l, \tag{1a}$$

$$E_m^l = \kappa^l f_m^l C_m L_m, \tag{1b}$$

where f_m^l is local computing capacity of MU m, and κ^l is the effective switched capacity [5].

2) Collaborative Cloud Computing: Collaborative cloud computing combines the MEC and MCC. For general analysis, we give two reasonable assumptions: (1) Each computation application can be arbitrarily split without considering the inherent content, which corresponds to the scenarios like video compression and speech recognition [12]–[14]. (2) The edge cloud and the central cloud work in parallel, and the central cloud allocates its complete computation resource to all MDs for parallel computing. According to [15], each edge cloud determines whether the offloaded applications should be executed by themselves or edge cloud and central cloud collaboratively. For latter, the edge cloud should determine the proportion of each application that needs to be executed on the central cloud server after it is completely received. We first consider how to design the optimal splitting ratio $\alpha_m^n \in [0,1]$ for achieving the lowest delay. Since edge cloud and central cloud process applications in parallel, the overall time consumption for executing an application of MU m by edge cloud n can be expressed as:

$$T_m^{n,exe} = \mathbf{min}\,\mathbf{max}\{T_m^{n,edge}, T_m^{n,ed-ce} + T_m^{cent}\}, \tag{2}$$

where $T_m^{n,edge} = \frac{\alpha_m^n L_m C_m}{f_m^{n,e}}$ is computation delay of edge cloud n executing application of MU m, $T_m^{n,ed-ce} = \frac{(1-\alpha_m^n)L_m}{W}$ is transmission delay through a backhaul link, where W is the backhaul communication capacity. $T_m^{cent} = \frac{(1-\alpha_m^n)L_m C_m}{F^{cent}}$ is the computation delay of central cloud, where F^{cent} is the ultra-high frequency of central cloud. By analyzing the monotonicity [15], the minimum value of $T_m^{n,exe}$ is achieved when $T_m^{n,edge} = T_m^{n,ed-ce} + T_m^{cent}$, therefore the optimal splitting ratio can be derived as:

$$\alpha_m^n = \frac{f_m^{n,e}(F^{cent} + C_m W)}{f_m^{n,e}(F^{cent} + C_m W) + F^{cent} C_m W}, \tag{3}$$

where $f_m^{n,e}$ is the computing frequency allocated by edge cloud n to MU m. The overall delay and energy consumption of the collaborative cloud computing at an optimal splitting ratio can be expressed as:

$$T_m^{n,exe} = \frac{\alpha_m^n L_m C_m}{f_m^{n,e}}, \tag{4a}$$

$$E_m^{n,exe} = \alpha_m^n \kappa^e f_m^{n,e} C_m L_m, \tag{4b}$$

where κ^e is the effective swatch capacity [15].

B. Communication Model

Computation-intensive applications are offloaded through the wireless channel. Here, $a_m \in \{0,1\}$ denotes the offloading decision of MU m, where $a_m = 0$ means local computing, and $a_m = 1$ means collaborative cloud computing. Hence, we can formulate an offloading decision profile as $\mathbf{a} = (a_1, a_2, \ldots, a_m)$, and the accumulation of offloading decisions $K(\mathbf{a}) = \sum_{m=1}^{M} a_m$ expresses the number of the offloaded applications. Based on the OFDMA technology, the total bandwidth B is equally divided among $K(\mathbf{a})$ offloaded devices [16]. Based on the Shannon principle, the wireless transmission rate can be expressed as:

$$R_m^n(p_m^{n,t}, \mathbf{a}) = \frac{B}{K(\mathbf{a})} \log_2\left(1 + \frac{p_m^{n,t} H_m^n}{B N_0 / K(\mathbf{a})}\right), \tag{5}$$

where $p_m^{n,t}$ indicates the transmission power of MU m to edge cloud n, which can be configured by the MU subject to a maximum transmission power constraint. H_m^n is the wireless channel gain between MU m and edge cloud n. B and N_0 denote the total channel bandwidth and noise power spectral density, respectively. The transmission energy consumption and time consumption are affected by $\mathbf{a}$ and can be represented by:

$$T_m^{n,trans}(p_m^{n,t}, K(\mathbf{a})) = \frac{L_m}{R_m^n(p_m^{n,t}, K(\mathbf{a}))}, \tag{6a}$$

$$E_m^{n,trans}(p_m^{n,t}, K(\mathbf{a})) = p_m^{n,t} \cdot T_m^{n,trans}(p_m^{n,t}, K(\mathbf{a})). \tag{6b}$$

Eqs. (6a) and (6b) show that transmission time consumption and energy consumption of transmission are affected by the offloading decision and transmission power.

III. Problem Formulation

To measure the system performance, we formulate an ETC minimization problem by optimizing the resource allocation, mobile-edge matching strategies, and offloading decisions jointly.

Definition 1: ETC is defined as the weighted sum of the energy consumption and time consumption for performing a particular action such as transmission, local computing, and collaborative cloud computing, which is expressed as $ETC_m = \gamma_m^e E + \gamma_m^t T$.

γ_m^e and $\gamma_m^t \in [0,1]$ are weight factors that measure the preference of the application of MU m. $\gamma_m^e + \gamma_m^t = 1$ balances the time consumption and energy consumption in case of excessive loss caused by massive weight. The ETCs of local computing, transmission, and edge cloud execution are represented by $ETC_m^l = \gamma_m^e E_m^l + \gamma_m^t T_m^l$, $ETC_m^{n,trans} = \gamma_m^e E_m^{n,trans} + \gamma_m^t T_m^{n,trans}$, and $ETC_m^{n,exe} = \gamma_m^e E_m^{n,exe} + \gamma_m^t T_m^{n,exe}$, respectively. Thus, the ETC of collaborative cloud computing for MU m can be expressed as:

$$ETC_m^{n,c} = \sum_{n=1}^{N} b_m^n (ETC_m^{n,exe} + ETC_m^{n,trans}), \tag{7}$$

where b_m^n denotes the mobile-edge matching strategy. If the MU m offloads its application to the edge cloud n, $b_m^n = 1$, otherwise $b_m^n = 0$. Considering the offloading decisions, the ETC of MU m can be represented as:

$$ETC_m = a_m ETC_m^{n,c} + (1 - a_m) ETC_m^l. \tag{8}$$

We aim to minimize the system $ETC = \sum_{m=1}^{M} ETC_m$ by optimizing the p_m^t, $f_m^{n,e}$, $\mathbf{b}$ and $\mathbf{a}$. Accordingly, the problem can be formulated as:

$$\mathbf{P}: \min_{a_m, b_m^n, p_m^{n,t}, f_m^{n,e}} \sum_{m=1}^{M} ETC_m \tag{9}$$

$$\begin{aligned}
\mathbf{s.t.}\quad & C1: a_m \in \{0,1\}, \forall m \in \mathcal{M} \\
& C2: b_m^n \in \{0,1\}, \forall m \in \mathcal{M}_{off}, \forall n \in \mathcal{N}, \\
& C3: \sum_{n=1}^{N} b_m^n \le 1, \forall m \in \mathcal{M}_{off}, \\
& C4: 0 \le p_m^{n,t} \le p_m^{max}, \forall m \in \mathcal{M}_{off}, \\
& C5: f_{min}^{n,e} \le f_m^{n,e} \le f_{max}^{n,e}, \forall m \in \mathcal{M}_{off}, \\
& C6: 0 \le \sum_{m=1}^{M} b_m^n \le Cap_n, \forall m \in \mathcal{M}_{off},
\end{aligned}$$

where $\mathcal{M}_{off}$ is the set of MUs offloading applications, and Cap_n is the maximum computing capacity of edge cloud n. $C1$ states that each application can be executed locally or offloaded onto the collaborative cloud. $C2 \sim C6$ are the constraints for the offloaded users in $\mathcal{M}_{off}$. $C2 \sim C3$ ensure that each application can only access one edge cloud. $C4$ and $C5$ indicate the upper bounds of transmission power and edge clouds computing frequency, respectively. $C6$ states that the number of applications served by a particular edge cloud should be within its capacity.

IV. Problem Decomposition

Since the formulated problem $\mathbf{P}$ is a mixed-integer nonlinear programming (MINLP) problem, which is difficult to solve directly, we decompose $\mathbf{P}$ into three subproblems: resource allocation, mobile-edge matching problem, and offloading decision. Since the three issues are coupled, we propose GS-MEAL and GS-MOA/GS-SOA, which iteratively update $\mathbf{b}$, $\mathbf{a}$, and resource allocation to minimize the system ETC.

A. Resources Allocation

Clearly, resource allocation is valid in the case that the application needs to be offloaded onto a specific edge cloud, i.e., $a_m = 1$ and $b_m^n = 1$. The minimization of $\sum_{m=1}^{M} ETC_m$ can be transformed as minimization of each $ETC_m^{n,c}$ by optimizing the p_m^n and $f_m^{n,e}$. Substitute the $ETC_m^{n,trans}$ and $ETC_m^{n,exe}$ into $ETC_m^{n,c}$, minimizing $ETC_m^{n,c}$ can be written as $\mathbf{P1}$:

$$\mathbf{P1}: \min_{p_m^{n,t}, f_m^{n,e}} \gamma_m^t (T_m^{n,exe} + T_m^{n,trans}) + \gamma_m^e (E_m^{n,exe} + E_m^{n,trans}),$$
$$\mathbf{s.t.} \quad C4, C5. \tag{10}$$

Substitute the Eq.(4a), Eq.(4b), Eq.(6a), Eq.(6b) in Eq.(10), $\mathbf{P1}$ can be transformed as:

$$\mathbf{P1}: \min_{p_m^{n,t}, f_m^{n,e}} \frac{\gamma_m^t L_m K/B + \gamma_m^e L_m K p_m^{n,t}/B}{log_2(1 + \frac{p_m^{n,t} H_m^n K}{B N_0})} + \frac{\gamma_m^t L_m C_m (F^{cent} + C_m) f_m^{n,e} + \gamma_m^e \kappa^e L_m C_m (F^{cent} + C_m)(f_m^{n,e})^3}{(F^{cent} + C_m)(f_m^{n,e})^2 + F^{cent} \cdot C_m W f_m^{n,e}},$$
$$\mathbf{s.t.} \quad C4, C5. \tag{11}$$

Since $p_m^{n,t}$ and $f_m^{n,e}$ are independent, $p_m^{n,t}$ and $f_m^{n,e}$ can be optimized separately by bisection and polynomial analysis methods, respectively.

1) Communication Resource Allocation: The $p_m^{n,t}$ can be optimized by minimizing $\mathbf{P2}$:

$$\mathbf{P2}: \min_{p_m^{n,t}} \frac{\gamma_m^t L_m K/B + \gamma_m^e L_m K p_m^{n,t}/B}{log_2(1 + \frac{p_m^{n,t} H_m^n K}{B N_0})},$$
$$\mathbf{s.t.} \quad C4. \tag{12}$$

Since Eq.(12) is quasiconvex in the domain [14], bisection method is used to solve it [16].

2) Edge Cloud Computing Frequency Control: The $f_m^{n,e}$ can be optimized by minimizing $\mathbf{P3}$, expressed as:

$$\mathbf{P3}: \min_{f_m^{n,e}} \frac{\gamma_m^t L_m C_m (F^{cent} + C_m) f_m^{n,e} + \gamma_m^e \kappa^e L_m C_m (F^{cent} + C_m)(f_m^{n,e})^3}{(F^{cent} + C_m)(f_m^{n,e})^2 + F^{cent} \cdot C_m W f_m^{n,e}},$$
$$\mathbf{s.t.} \quad C5. \tag{13}$$

Algorithm 1: GS-MEAL.

Input: $\mathcal{M}, \mathcal{N}, Cap_n, n = 1, 2, \ldots, N$.
Output: Mobile-edge matching strategy b_m^n.
Initialize: $\mathcal{PL}_{MU}^m$, $\mathcal{PL}_{edge}^n$, $\mathcal{M}_{unmatch} = \mathcal{M}$.
while $\mathcal{M}_{unmatch} \neq \emptyset$ **do**
  ******** *MUs propose to the edge clouds* ********
  for all $m \in \mathcal{M}_{unmatch}$ **do**
    1. Propose to the most preferred edge cloud n based on $\mathcal{PL}_{MU}^m$ and remove it from $\mathcal{PL}_{MU}^m$, set $b_m^n = 1$;
  end for
  ********* *Edge clouds make decision* **********
  for $n \in N$ **do**
    if $\sum_{m=1}^{M} b_m^n \leq Cap_n$ **then**
      2. Edge cloud n keeps all of proposed MUs and remove accepted MUs from $\mathcal{M}_{unmatch}$;
    else
      3. Edge cloud n keeps the most preferred Cap_n MUs and reject the rest;
      4. Remove these Cap_n MUs from $\mathcal{M}_{unmatch}$;
      5. Add the rejected MUs into the $\mathcal{M}_{unmatch}$, and set $b_m^n = 0$.
    end if
  end for
end while

Combining the bisection method and quadratic polynomial analyzing [15], the minimized $\mathbf{P3}$ can be obtained by optimizing edge cloud computing frequency.

B. Optimization of Mobile-Edge Matching Strategies

GS-MEAL is proposed to optimize the $\mathbf{b}$ based on the specific resource allocation. Substituting $ETC_m^{n,c}$ into Eq.(9), the mobile-edge matching problem can be expressed as $\mathbf{P4}$:

$$\mathbf{P4}: \min_{b_m^n} \sum_{m=1}^{M} a_m b_m^n (ETC_m^{n,exe} + ETC_m^{n,trans}),$$
$$\mathbf{s.t.} \quad C2, C3, C6. \tag{14}$$

$\mathbf{P4}$ is transformed as a many-to-one matching problem through GS-MEAL *w.r.t* known $\mathbf{a}$ and known resource allocation, which matches the multiple MUs to their preferred edge clouds with two-sided preferences [18], [19].

1) Preference List: We design the preference lists of edge cloud and MU as $\mathcal{PL}_{edge}^n(m)$ and $\mathcal{PL}_{MU}^m(n)$, respectively. Each edge cloud $n \in N$ ranks all the offloaded MUs by sorting optimal $ETC_m^{n,exe}, m \in \mathcal{M}_{off}$ in a strictly ascending order. Likewise, each MU $m \in M$ ranks all edge clouds by sorting optimal $ETC_m^{n,trans}, n = 1, 2, \ldots, N$ for all edge clouds in a strictly ascending order. Therefore, edge cloud n prefers MU m_1 to MU m_2 if $\mathcal{PL}_{edge}^n(m_1) < \mathcal{PL}_{edge}^n(m_2)$.

2) Gs-Meal: The process of GS-MEAL is described in detail in this subsection. Each MU proposes its current most preferred edge cloud in each iteration, w.r.t. the preference list over edge clouds, and remove this edge cloud from its preference list after proposing to it. After all the MUs have proposed their current preferred edge cloud, each edge cloud checks its received proposals. Several edge clouds may not get any proposals, while others may get one or several proposals. The edge cloud that does not get any proposal remains unpaired. The edge cloud that gets several proposals, together with the MUs it has accepted in the former iterations, temporarily accepts the most preferred MUs up to its capacity and rejects the rest, w.r.t. its preference over MUs. The

proposal iteration goes on until all MUs are accepted. The GS-MEAL is described in Algorithm 1.

Definition 2: A $Pair(m_i, n_j)$ is a blocking pair for a Matching Set ($\mathcal{MS}$), if the following conditions are satisfied relative to $\mathcal{MS}$:

(1) m_i is prefers n_j to $Pair(m_i)$,

(2) n_j is prefers m_i to $Pair(n_j)$,

where $Pair(*)$ denotes the matched element corresponding to the element in the bracket. $\mathcal{MS}$ is said to be stable if it admits no blocking pair.

Remark 1: GS-MEAL is a stable matching algorithm.

Proof: Since each MU prioritizes selecting their most preferred edge cloud, the edge cloud will consider whether to replace the paired MUs according to their preference and capacity. Therefore, there exists no blocking pairing. For example, MU2 matches with edge cloud 2. If MU1 prefers edge cloud 2 and edge cloud 2 also prefers MU1 to MU2, edge cloud 2 will prioritize accepting MU1 and deferred accepting or rejecting MU2 according to its capacity. So there is no blocking pairing. Consequently, $\mathbf{P4}$ can be transformed into a two-sided preference minimization problem based on the Gale-Shapley algorithm.

C. Optimal Offloading Decision

The optimal offloading decision can be obtained by exhaustively enumerating all possible offloading decision, which causes a high computational complexity. To this end, we propose the GS-MOA and GS-SOA to determine sub-optimal offloading decisions. Substituting the Eq.(1b) into Eq.(9), the formulated problem can be expressed as $\mathbf{P5}$:

$$\mathbf{P5}: \quad \min_{a_m} \sum\nolimits_{m=1}^{M} a_m ETC_m^{n,c}(b_m^n) + (1-a_m) ETC_m^l,$$
$$\mathbf{s.t.} \quad C1, C6. \tag{15}$$

Definition 3: Offloading decision $\mathbf{a}$ is better than $\mathbf{a}^*$, which is denoted as $\mathbf{a} \rhd \mathbf{a}^*$, if and only if the system $ETC(\mathbf{a})$ is lower than that of $ETC(\mathbf{a}^*)$, i.e., $ETC(\mathbf{a}) \leq ETC(\mathbf{a}^*)$.

Definition 4: A local computing set is inextensible, if and only if $|\mathcal{M}_{local}| = \mathcal{M}$ or there exists no local computing set $\mathcal{M}_{local} \cup \{k\}$ satisfing $\mathbf{a}_{\mathcal{M}_{local} \cup \{k\}} \rhd \mathbf{a}_{\mathcal{M}_{local}}$.

Definition 5: The marginal value $\triangle_k$ denotes the reduced system ETC by adding application k into the $\mathcal{M}_{local}$. If moving MU k from $\mathcal{M}_{off}$ under the new updated $\mathbf{a}^*$, $\mathbf{b}^*$, and $ETC_m^{n,c}$, the $\triangle_k$ is less than zero, i.e., $\triangle_k \leq 0$, MU k should be executed locally, which $\triangle_k$ is represented as:

$$\begin{aligned} \triangle_k &= ETC(\mathbf{a}^*) - ETC(\mathbf{a}) \\ &= ETC_k^l - \left(\sum_{m=1}^{M} a_m^* ETC_m^{n,c}(\mathbf{b}^*) - \sum_{m=1}^{M} a_m ETC_m^{n,c}(\mathbf{b}) \right). \end{aligned} \tag{16}$$

Note that removing MU k has an impact on $ETC_m^{n,trans}$ due to the wireless channel sharing. Specifically, we initially assume that all MUs are offloaded, the initial offloading set is $\mathcal{M}_{off} = \mathcal{M}$ and local computing set is $\mathcal{M}_{local} = \emptyset$. Based on *Definition 3*, *Definition 4* and *Definition 5*, we can repeatedly extend the $\mathcal{M}_{local}$ by moving MU from $\mathcal{M}_{off}$ to $\mathcal{M}_{local}$ until the local computing set is inextensible.

Definition 6: $\mathbf{a}$ is locally optimal if and only if $\mathbf{a}$ is the best offloading decision under a fixed number of local computing MUs, i.e., $\mathbf{a}_{|\mathcal{M}_{local}|} \rhd \mathbf{a}_{|\mathcal{M}^*_{local}|}$, where $\mathcal{M}_{local}$ and $\mathcal{M}^*_{local}$ are different local computing sets with the same number of local computing MUs. Note that determining the optimality of $\mathbf{a}$ needs to comparing all the local computing set $\mathcal{M}_{local}$ with the same number of local computing MUs.

Algorithm 2: GS-MOA.

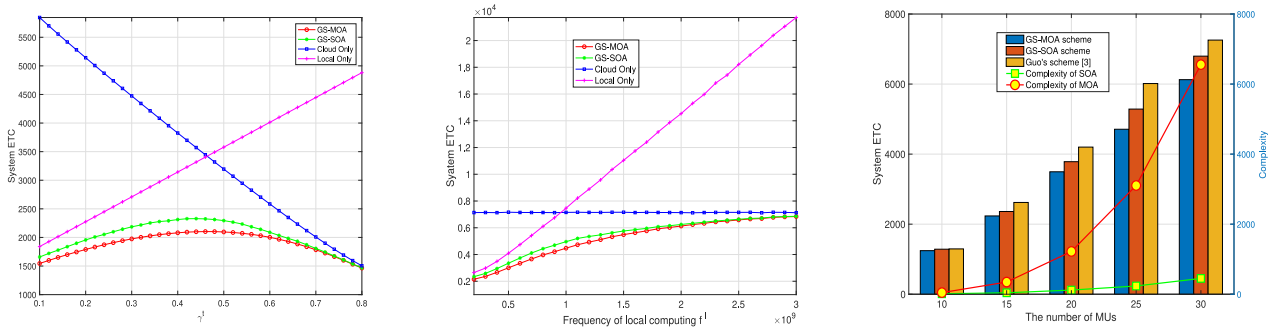
Input: b_m^n, $\mathbf{a} = \{1, \ldots, 1\}$, $ETC(\mathbf{a})$, $\mathcal{M}_{off} = \mathcal{M}$, $\mathcal{M}_{local} = \emptyset$.
Output: Offloading decision $\mathbf{a}$.
repeat
for $\forall k \in \mathcal{M}_{off}$ **do**
1. Calculate new $ETC_m^{n,trans}$ for $m \in \mathcal{M}_{off}/\{k\}, n \in \mathcal{N}$;
2. Update b_m^n for $m \in \mathcal{M}_{off}/\{k\}, n \in \mathcal{N}$;
3. Calculate the new $ETC(\mathbf{a}_{\mathcal{M}_{off}}/\{k\})$ based on the new b_m^n and $\mathbf{a}_{\mathcal{M}_{off}}/\{k\}$;
4. Caculate $\triangle_k$ according to the Eq. (16) for each $\mathbf{a}_{\mathcal{M}_{off}/\{k\}}, \forall k \in \mathcal{M}_{off}$;
if $\triangle_k \leq 0$ **then**
5. $\triangle_{\mathbf{readylocal}} = \{\triangle_k\}$;
6. $\triangle_{k,opt} = \min \triangle_{\mathbf{readylocal}}$;
7. Set $a_k = 0$, update offloading decisions $\mathbf{a}^*$;
8. MU k is determined to be executed locally;
end if
end for
9. Update $ETC(\mathbf{a}^*) = ETC(\mathbf{a}) + \triangle_{k,opt}$;
10. Update $\mathcal{M}_{local} = \mathcal{M}_{local} \cup \{k\}$;
11. Update $\mathcal{M}_{off} = \mathcal{M}_{off}/\{k\}$;
12. Set $\mathbf{a} = \mathbf{a}^*$;
until For each $\forall k \in \mathcal{M}_{off}$, $\triangle_{\mathbf{readylocal}} = \emptyset$.

1) Gs-Moa: The objective of the proposed algorithm is to find the applications that can be executed locally by judging whether the ETC can be reduced. The algorithm converges until the system ETC cannot be further reduced. In each iteration, each application executed on the collaborative cloud is assumed computed locally corresponding to different $\mathcal{M}_{off}/\{k\}$. Next, $ETC_m^{n,trans}$, mobile-edge matching strategies $\mathbf{b}$, system ETC and $\triangle_k$ are updated for each $\mathcal{M}_{off}/\{k\}$, i.e., Steps 1-4 in the Algorithm 2. Then, all the negative $\triangle_k$s *w.r.t* each $\mathcal{M}_{off}/\{k\}$ are collected into $\triangle_{\mathbf{readylocal}}$, i.e., Step 5. The MU k with the smallest $\triangle_k$ in the $\triangle_{\mathbf{readylocal}}$ is selected to be executed locally, i.e., Steps 6-8. System ETC, $\mathcal{M}_{local}$, $\mathcal{M}_{off}$ and offloading decision $\mathbf{a}$ are updated in Steps 9-12. GS-MOA converges until $\triangle_{\mathbf{readylocal}}$ is an empty set.

2) Gs-Soa: Compared to the GS-MOA, the GS-SOA does not need to search for the locally optimal offloading decision. Therefore, the time complexity of the GS-SOA is the level of $O(m)$, which is lower than that of the GS-MOA. Instead, GS-SOA first randomly numbers all MUs in the network, then determines whether each MU should be moved from $\mathcal{M}_{off}$ to $\mathcal{M}_{local}$ in sequential, re-allocate the communication resources, and update $\mathbf{b}$. If system ETC can be reduced, the MU k should be executed locally. GS-SOA is shown as Algorithm 3.

V. Simulation Results

This section provides the numerical results to evaluate the proposed algorithms. We assume that there exist multiple edge clouds and multiple MUs, and bandwidth are divided equally. The input data size and the required CPU cycles/bit for each application both subject to uniform distribution with $L_m \in [1, 5]$ Mbits, and $C_m \in [500, 1500]$ CPU cycles/bit, respectively. The wireless channel gains are Rayleigh random variables with unit variance. Fig. 1(a) depicts the impact of the weight factors γ^t on ETC. As γ^t increases, ETC under the Cloud Only scheme and Local Only scheme show completely different trends. The ETC under the Local Only scheme rises approximately linearly, while the ETC of the Cloud Only is opposite. GS-MOA and GS-SOA execution schemes always show a better performance than the other two



(a) (b) (c)

Fig. 1. (a) The impact of the weight factor γ^t. (b) The impact of local computing frequency f^l. (c) Comparison of the complexity and ETC for different offloading scheme.

Algorithm 3: GS-SOA.

Input: b_m^n, $\mathbf{a} = \{1, \ldots, 1\}$, $ETC(\mathbf{a})$, $\mathcal{M}_{off} = \mathcal{M}$, $\mathcal{M}_{local} = \emptyset$, random numbered the M MUs.
Output: Offloading decision $\mathbf{a}$.
for $k = 1 : M$ **do**
1. Calculate new $ETC_m^{n,trans}$ for $m \in \mathcal{M}_{off}/\{k\}, n \in \mathcal{N}$;
2. Update b_m^n for $m \in \mathcal{M}_{off}/\{k\}, n \in \mathcal{N}$;
3. Calculate the new $ETC(\mathbf{a}_{\mathcal{M}_{off}}/\{k\})$ based on the new b_m^n and $\mathbf{a}_{\mathcal{M}_{off}}/\{k\}$;
4. Caculate $\triangle_k$ according to the Eq. (16) for each offloading decision $\mathbf{a}_{\mathcal{M}_{off}/\{k\}}$;
if $\triangle_k \leq 0$ **then**
5. Set $a_k = 0$, update offloading decisions $\mathbf{a}^*$;
6. MU k is determined to be executed locally;
end if
7. Update $ETC(\mathbf{a}^*) = ETC(\mathbf{a}) + \triangle_k$;
8. Update $\mathcal{M}_{local} = \mathcal{M}_{local} \cup \{k\}$;
9. Update $\mathcal{M}_{off} = \mathcal{M}_{off}/\{k\}$;
10. Set $\mathbf{a} = \mathbf{a}^*$.
end for

schemes. When γ^t is small, the ETC under GS-MOA and GS-SOA are close to that under the Local Only scheme, which means that more delay-tolerant applications are executed locally. As γ^t increases, more delay-sensitive applications are offloaded onto the collaborative cloud to save time. Fig. 1(b) shows the ETC under different schemes versus f^l. Since the collaborative cloud computing is independent of f^l, the ETC under the Cloud Only scheme always remains constant. In comparison, the ETC under the Local Only scheme rises almost linearly with the increase of local computing frequency, which implies that large f^l leads to enormous energy consumption. One can observe that the performances of proposed GS-MOA and GS-SOA are always better than that of the other two schemes. We can find that ETC under the GS-MOA and GS-SOA firstly rise relatively shapely and close to that under the Local Only scheme, and then gradually slow down and approach that under the Cloud Only scheme. This indicates that more applications are executed on the MDs in the case of small f^l, and more applications are executed on the collaborative cloud for a large f^l. Therefore, it can be inferred that when the local computing frequency is significant, the ETC can be effectively reduced by properly offloading application onto the collaborative cloud. Fig. 1(c) shows the computational complexities of the proposed algorithms, and the system ETC with different numbers of MUs (i.e., applications). Meanwhile, we also plot the system ETC under the existing algorithm designed by Guo *et al.* [3] for comparison. One can observe that ETC increase with the number of applications under all algorithms, but ETC always lower under our proposed algorithms than that under the existing algorithm, which shows the effectiveness of the proposed algorithms. In addition, we also can find that the computational complexity of the GS-SOA is always lower than that of the GS-MOA at the expense of relative higher system ETC.

VI. Conclusion

This paper has investigated a computation offloading problem in a three-tier network consisting of multiple MUs, multiple edge clouds, and a central cloud. On this basis, we designed a system ETC minimization problem by jointly optimizing the resource allocation, mobile-edge matching strategies, and offloading decision. Due to the difficulty of directly solving the formulated problem, we decompose the original one and proposed two offloading algorithms, namely GS-MOA/GS-SOA, to find the optimal offloading decisions, which combined the GS-MEAL to update the resource allocations, mobile-edge matching strategies, and offloading decisions iteratively for minimizing the system ETC. Finally, numerical results have shown the effectiveness of our proposed algorithms.

References

[1] W. Hao *et al.*, "Robust design for intelligent reflecting surface-assisted MIMO-OFDMA terahertz IoT networks," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 13052–13064, Aug. 2021.
[2] Y. Dai, M. Sheng, J. Liu, N. Cheng, X. Shen, and Q. Yang, "Joint mode selection and resource allocation for D2D-enabled NOMA cellular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 7, pp. 6721–6733, Jul. 2019.
[3] W. Hao, M. Zeng, G. Sun, and P. Xiao, "Edge cache-assisted secure low-latency millimeter-wave transmission," *IEEE Internent Things J.*, vol. 7, no. 3, pp. 1815–1825, Mar. 2020.
[4] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, "Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing," *IEEE Trans. Mobile Comput.*, vol. 18, no. 2, pp. 319–333, Feb. 2019.
[5] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3435–3447, Apr. 2017.
[6] M. Sheng, Y. Dai, J. Liu, N. Cheng, X. Shen, and Q. Yang, "Delay-aware computation offloading in NOMA MEC under differentiated uploading delay," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2813–2826, Apr. 2020.

[7] H. Cao and J. Cai, "Distributed multiuser computation offloading for cloudlet-based mobile cloud computing: A game-theoretic machine learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 752–764, Jan. 2018.

[8] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5031–5044, May 2019.

[9] A. E. Alchalabi, S. Shirmohammadi, S. Mohammed, S. Stoian, and K. Vijayasuganthan, "Fair server selection in edge computing with Q-value-normalized action-suppressed quadruple Q-Learning," *IEEE Trans. Atif. Intell.*, vol. 2, no. 6, pp. 519–527, Dec. 2021.

[10] M. R. Rahimi, N. Venkatasubramanian, S. Mehrotra, and A. V. Vasilakos, "MAPCloud: Mobile applications on an elastic and scalable 2-tier cloud architecture," in *Proc. IEEE 5th Int. Conf. Utility Cloud Comput.*, 2012, pp. 83–90.

[11] H. Yu, Q. Wang, and S. Guo, "Energy-efficient task offloading and resource scheduling for mobile edge computing," in *Proc. IEEE Int. Conf. Netw., Architecture Storage*, 2018, pp. 1–4.

[12] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, Oct./Dec. 2017.

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

[14] X. Meng, W. Wang, and Z. Zhang, "Delay-constrained hybrid computation offloading with cloud and fog computing," *IEEE Access*, vol. 5, pp. 21355–21367, 2017.

[15] Z. Gao, W. Hao, Z. Han, and S. Yang, "Q-learning-based task offloading and resources optimization for a collaborative computing system," *IEEE Access*, vol. 8, pp. 149011–149024, 2020.

[16] Z. Kuang, Y. Shi, S. Guo, J. Dan, and B. Xiao, "Multi-user offloading game strategy in OFDMA mobile cloud computing system," *IEEE Trans. Veh. Technol.* vol. 68, no. 12, pp. 12190–12201, Dec. 2019.

[17] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2009.

[18] M. Abououf, S. Singh, H. Otrok, R. Mizouni, and A. Ouali, "Gale-Shapley matching game selection-A framework for user satisfaction," *IEEE Access*, vol. 7, pp. 3694–3703, 2019.

[19] Y. Zhang, Y. Mao, and S. Zhong, "Joint differentially private Gale-Shapley mechanisms for location privacy protection in mobile traffic offloading systems," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 10, pp. 2738–2749, Oct. 2016.