Joint Optimization of Service Caching Task Offloading and Resource Allocation in Cloud-Edge Cooperative Network

Chaogang Tang*, Yao Ding*, Shuo Xiao*, Huaming Wu†, Ruidong Li‡
*School of Computer Science and Technology, China University of Mining and Technology, 221116, Xuzhou, China
†The Center for Applied Mathematics, Tianjin University, 300072, Tianjin, China
‡The Institute of Science and Engineering, Kanazawa University, Kanazawa 920-1192, Japan
cgtang@cumt.edu.cn, yding32@cumt.edu.cn, sxiao@cumt.edu.cn, whming@tju.edu.cn, liruidong@ieee.org

***Abstract*—The cloud-edge cooperative network presents both opportunities and challenges for latency-sensitive and computation-intensive tasks. Effectively harnessing the strengths of edge computing and cloud computing enables real-time task handling, thus reaching a win-win situation where not only the stated quality of service (QoS) is delivered from the angle of service providers, but also the quality of experience (QoE) is improved from the angle of service requestors. However, due to the unpredictable task generation and time-varying environments, it is challenging to achieve optimal task scheduling and effective resource management and allocation. To address this issue, we propose an innovative cloud-edge framework that incorporates task offloading, service caching, and resource allocation in this paper. In this framework, we can determine where to offload the task, e.g., locally, at the edge, or in the cloud center. In view of the importance of the superior user experience, we aim to maximize the user satisfaction regarding task offloading in this framework. The problem is actually a mixed-integer nonlinear programming (MINLP) problem that entails simultaneously addressing cache decisions, offloading decisions, and resources allocation in a dynamic cloud-edge computing system. Owing to the NP-hardness, our original problem is decomposed into two layers of alternating problems. Specifically, we adopt a genetic algorithm (GA) based approach to jointly make cache and offloading decisions, and then iteratively optimize the communication and computing resources allocation. Extensive experimentation has demonstrated the feasibility and effectiveness of the proposed approach.**

***Index Terms*—Service caching, task offloading, user satisfaction, cloud-edge network, QoS**

I. Introduction

Recently emerging Internet of Things (IoT) applications, including smart homes, remote healthcare, augmented reality, intelligent transportation, and online interactive games, have become an integral part of modern society. Driven by ultra-low latency and superior user experience, these applications exhibit significant demands for computation resources and are highly sensitive to latency requirements [1], [2]. However, mobile devices equipped with constrained computation resources are hardly to meet such demands on QoS and QoE. Cloud computing endowed with immense computational power, extensive storage resources, and flexible scalability, can elastically address the computationally intensive demands of diverse applications. Whereas, the substantially physical distance between cloud data centers and user endpoints can cause a highly long latency that may deteriorate the total performance regarding the response latency [3].

For some specific applications such as intelligent transportation and health monitoring, response latency is particularly significant. Currently, many works focus on addressing this issue. For instance, an edge computing empowered wireless network is put forward, which brings computational resources and storage function to the edge of the network [4]. This architecture enables task processing and data analysis in close proximity to users, which not only delivers lower response latency but also improves the overall performance with regards to reliability and robustness [5]. Despite the advantages of edge computing (EC), there still exist some shortcomings such as restricted computing capabilities compared to the cloud center and the front-haul link pressure incurred by the tremendous increase in data traffic. These shortcomings make it challenging to deliver high QoS for the tasks offloaded and executed in EC.

To overcome the aforementioned drawback, we propose a caching-assisted cloud-edge computing framework that combines the abundant cloud resources with the fast response capabilities of edge computing. We try to jointly optimize service caching, task offloading and resource scheduling in this framework. Here, service caching means to cache service related databases and libraries at the edge server (ES). As users can directly obtain the required applications from nearby edge nodes, the data transmission delay can be significantly reduced compared to uploading both application/service and input parameters. However, mobile services are often vary in terms of required computing and storage resources [6], and thus the resource-constrained edge server can only cache limited number of applications, which requires us to prudently design an efficient service caching strategy.

In addition, the task offloading decision is made according to current state of involved entities including response latency, energy consumption, as well as bandwidth utilization. It is crucial to effectively scheduling restricted computing resources at ES to meet the diverse requirements of dif-

ferent applications, considering the inherent constraints of edge resources, unpredicted dynamics and spatial demand coupling [3]. In EC systems, the resource allocation in current works is often conducted for response latency minimization or energy consumption saving. Various approaches, including game-theoretic methods [7], heuristic algorithms [8], or deep reinforcement learning techniques [9], have been employed to tackle these resource allocation challenges.

In contrast, we focus on the issue of the fairness in this paper. Fairness among users is of paramount importance in certain scenarios, such as multiplayer online games, where disparities in user latency serve as a crucial metric for assessing user experience [10]. A significant delay between players, as evidenced by previous studies [11], can hardly guarantee a fair and equal opportunity to all participants. Accordingly, it is important to ensure a fair distribution of resources without favoring specific users. Generally, the major contributions of this paper can be outlined as below:

- We try to jointly optimize task offloading, service caching, and resource allocation in the cloud-edge cooperative network. The optimization problem is modeled as a MINLP problem, considering uncertain resource demands, latency constraints of heterogeneous computing tasks and energy constraints. Multiple limitations are also considered regarding communication, computing, and storage resources.
- Considering the NP-hardness of this problem, we put forward a GA-based two-level optimization algorithm, referred to as GACORA, with the outer layer utilizing GA method for determining task offloading and service caching and the inner layer for iteratively determining the optimal communication and computation resource allocation.
- Extensive experiments have demonstrated that our GACORA strategy for the joint optimization in the cloud-edge network outperforms several alternative algorithms in average user satisfaction, average task latency, task completion rate and fairness among users.

II. System Model

A. Caching-Assisted Cloud-Edge Framework

The considered scenario is constituted by one cloud server (CS), one ES, and multiple user devices (UD). The ES with caching and computation functions is deployed together with the base station (BS), enabling possible task offloading from UDs to the ES via wireless networks. The set of UDs is expressed as $\mathcal{N} = \{1, ..., N\}$, where N is the number of UDs. The optimization period is slotted into discrete intervals also known as time slots $\mathcal{T} = \{1, ..., T\}$, where T represents the number of time slots [12]. Each UD generates one computation task in each time slot in this paper. The task t_n^t generated by UD n in slot t can be described by $t_n^t = (d_n^t, s_n^t, l_n^{max,t})$, where d_n^t represents the input data size of the task, s_n^t stands for the required computation resources (CPU cycles), and $l_n^{max,t}$ is the maximum tolerable delay. We further assume that these tasks are inherently atomic and indivisible in nature. There is a collection of K distinct cacheable services indexed by $\mathcal{A} = \{a_1, a_2, ..., a_K\}$. We assume that one task can only request one service and one service can serve multiple tasks in this paper. Thus, the relationship between task t_n^t and service k can be expressed as $\alpha_{n,k} \in \{0, 1\}$. If task t_n^t requests service k, $\alpha_{n,k} = 1$, and 0, otherwise. Note that the execution of task t_n^t on the ES is contingent upon the presence of the required service cached in the ES. So as to facilitate the execution of computational tasks, the caching strategy should be strategically decided within each time slot. We define $\delta_k^t \in \{0, 1\}$ to represent whether service k is cached at the ES. If service k is cached in slot t, $\delta_k^t = 1$, and 0, otherwise. As such, we have the following constraint $\sum_{k=1}^{K} \delta_k^t h_k^t \leq C$, wherein h_k specifies the storage requirement of service k, and C is the maximal storage capacity of the ES. For clarity, $\beta_n^t = \sum_{k=1}^{K} \alpha_{n,k} \delta_k^t$ denotes whether the service required by task t_n^t is cached in the ES during time slot t.

Through the utilization of Dynamic Spectrum Access (DSA) technology, the ES is empowered to distribute bandwidth among distinct UDs in accordance with their specific requests [13]. The notation $\mathcal{B}(t) = \{b_1^t, b_2^t, ..., b_N^t\}$ is the allocation vector denoting the spectrum resources allocated to each task in slot t. Therefore, we have $\sum_{n=1}^{N} \beta_n^t \lambda_n^t (2 - \lambda_n^t) b_n^t \leq W$, where W represents the total bandwidth of the uplink channel, and $\lambda_n^t (\in \{0, 1, 2\})$ represents the offloading destination for the task, where $\lambda_n^t = 0$ specifies that the task is undertaken locally, $\lambda_n^t = 1$ indicates that the task is offloaded to the ES, and $\lambda_n^t = 2$ denotes that the task is offloaded to the remote cloud.

B. Communication model

If one UD with insufficient computational capabilities is unable to fulfill a task and the requested service is cached in the ES, the task can be executed at ES. If the requested service is not cached, the task can be offloaded to the cloud. Thus, the transmission latency of offloading task to the edge or cloud is respectively described as:

$$l_{n,trs}^{e,t} = d_n^t / r_n^{e,t} \tag{1}$$

$$l_{n,trs}^{c,t} = d_n^t / r_n^{c,t} + 2\eta \tag{2}$$

where $r_n^{e,t}$, $r_n^{c,t}$ are the achievable uplink data rates between user device and ES/CS, respectively. η represents the propagation delay across the backbone network. $r_n^{e,t}$ can be further expressed as:

$$r_n^{e,t} = b_n^t \log\left(1 + p_n^{e,t} g_n^t\right) \tag{3}$$

where $p_n^{e,t}$ is the transmission power of user device n in time slot t, g_n^t specifies the uplink wireless channel gain between UD n and the ES.

C. Computing model

Let f_n^l, $f_n^{c,t}$ respectively denote the average processing frequency of UD n and the CS. Let $f_n^{e,t}$ denote the amount of computational resources allocated to the task t_n^t by the ES.

The execution delay for processing the task locally or at the ES/CS can be respectively calculated as:

$$l_n^{l,t} = s_n^t / f_n^l \tag{4}$$

$$l_{n,exe}^{e,t} = s_n^t / f_n^{e,t} \tag{5}$$

$$l_{n,exe}^{c,t} = s_n^t / f_n^{c,t} \tag{6}$$

Denote by F_e the total computational resources available at the ES, and we thus have $\sum_{n=1}^{N} \beta_n^t \lambda_n^t (2-\lambda_n^t) f_n^{e,t} \leq F_e$. The corresponding energy consumption for processing task t_n^t on the ES is given by:

$$e_{n,exe}^{e,t} = k^e s_n (f_n^{e,t})^2 \tag{7}$$

where k^e is the effective switched capacitance coefficient related to the ES. We use $e_{str}^{e,t} = \sum_{k=1}^{K} \delta_k^t \gamma_k$ to represent the total storage consumption in the ES, where γ_k denotes the average static power consumption for caching service k. The total energy consumption in the edge for accomplishing all the offloaded tasks with necessary services is then formulated as:

$$E^{e,t} = e_{str}^{e,t} + \sum_{n=1}^{N} \beta_n^t \lambda_n^t (2-\lambda_n^t) e_{n,exe}^{e,t} \tag{8}$$

III. Problem Formulation

Combining task offloading and service caching, the response latency required to accomplish the task t_n^t is generalized as:

$$\begin{aligned} l_n^t =& 1/2(\lambda_n^t - 1)(\lambda_n^t - 2) l_n^{l,t} \\ &+ \beta_n^t \lambda_n^t (2-\lambda_n^t)(l_{n,trs}^{e,t} + l_{n,exe}^{e,t}) \\ &+ 1/2 \lambda_n^t (\lambda_n^t - 1)(l_{n,trs}^{c,t} + l_{n,exe}^{c,t}) \end{aligned} \tag{9}$$

In contrast to prior research on task offloading, which primarily concentrated on optimizing either task processing delay or energy consumption, our emphasis here is to enhance the maximization of user satisfaction which considers both system response latency and fairness among users. Accordingly, the user satisfaction function for processing task t_n^t can be expressed as:

$$S_n^t = \log\left(1 + \phi + l_n^{max,t} - l_n^t\right) \tag{10}$$

where ϕ serves the purpose of normalizing the satisfaction metric and ensures that it remains non-negative. Recognizing the limitations imposed by the finite computational, communication and caching capacities of the ES, we jointly optimize service caching, computing offloading, and the resource allocation, while accommodating the energy constraint associated with the ES. Define $\delta(t) = \{\delta_k^t\}$ as the vector of service caching decision, $\lambda(t) = \{\lambda_n^t\}$ as the vector of task offloading decision, $B(t) = \{b_n^t\}$ and $f^e(t) = \{f_n^{e,t}\}$ as the vector of communication and computing resource allocation schemes, respectively. The optimization problem can be formulated as:

$$P1: \max_{\delta(t),\lambda(t),B(t),f^e(t)} \sum_{t=1}^{T} \sum_{n=1}^{N} S_n^t \tag{11}$$

$$\text{s.t.} \quad l_n^t \leq l_n^{max,t} \quad \forall n \in \mathcal{N}, \forall t \in \mathcal{T} \tag{11a}$$

$$E^{e,t} \leq Q \quad \forall t \in \mathcal{T} \tag{11b}$$

$$\sum_{k=1}^{K} \delta_k^t h_k^t \leq C \quad \forall t \in \mathcal{T} \tag{11c}$$

$$\sum_{n=1}^{N} \beta_n^t \lambda_n^t (2-\lambda_n^t) b_n^t \leq W \quad \forall t \in \mathcal{T} \tag{11d}$$

$$\sum_{n=1}^{N} \beta_n^t \lambda_n^t (2-\lambda_n^t) f_n^{e,t} \leq F_e \quad \forall t \in \mathcal{T} \tag{11e}$$

$$b_n^t \geq 0 \quad \forall n \in \mathcal{N}, \forall t \in \mathcal{T} \tag{11f}$$

$$f_n^{e,t} \geq 0 \quad \forall n \in \mathcal{N}, \forall t \in \mathcal{T} \tag{11g}$$

$$\delta_k^t \in \{0,1\} \quad \forall n \in \mathcal{N}, \forall t \in \mathcal{T} \tag{11h}$$

$$\lambda_n^t \in \{0,1,2\} \quad \forall n \in \mathcal{N}, \forall t \in \mathcal{T} \tag{11i}$$

The constraint (11a) implies that the processing delay of a task should not surpass the predefined threshold $l_n^{max,t}$. The condition (11b) specifies the maximal energy constraint Q on the total energy consumption in the ES. The constraint (11c) means that the total cached services should not go beyond the maximal storage capacity of the ES. The conditions (11d), (11e) indicate that the assigned transmission and computing resources should not surpass the maximum resources of the ES, respectively.

Remark: $P1$ is obviously a MINLP problem. Solving Problem $P1$ directly to obtain the optimal solution poses significant difficulties, owing to its NP-hardness. Thus, in order to reduce the time complexity, we put forward a GA based two-level iterative algorithm in this paper.

IV. Algorithm Design

We decouple the original problem into two subproblems, i.e., the resource allocation (RA) problem and the service caching and task offloading (SC&TO) problem [14], [15]. First, for any given feasible solutions to the SC&TO decisions, we can obtain the optimal RA policy through a distributed optimization approach to maximize the user satisfaction. Then, GA based approach is used to obtain the sub-optimal SC&TO decision, given the transmission bandwidth and CPU frequency allocation schemes.

A. Multi-resource Allocation

Based on the formulation of problem $P1$, under given service caching and task offloading strategy, the RA problem $P1.1$ takes the following form:

$$\begin{aligned} P1.1: & \max_{B(t),f^e(t)} \sum_{t=1}^{T} \sum_{n=1}^{N} S_n^t \\ & \text{s.t.} \quad (11a), (11b), (11d), (11e), (11f), (11g) \end{aligned} \tag{12}$$

In order to mitigate system complexity, the time-combinatorial problem is disentangled into subproblems within discrete time slots, and $\max \sum_{t=1}^{T} \max \sum_{n=1}^{N} S_n^t$ is employed to provide an approximation of the original problem

$P1.1$ [16]. Thus the resource allocation problem in a given time slot t can be further expressed as:

$$\begin{aligned} P1.2: \max_{B(t),f^e(t)} & \sum_{n=1}^{N} \log\left(1+\phi+l_n^{max,t}-l_n^t\right) \\ \text{s.t.} \quad & (11a),(11b),(11d),(11e),(11f),(11g) \end{aligned} \tag{13}$$

We further partition RA problem $P1.2$ into two components and propose a distributed method to optimize different blocks of variables iteratively. We obtain computation frequency allocation given the communication resource allocation strategy, and then update the latter repeatedly until the convergence appears.

1) Computing resource Allocation: Given $\delta(t)$, $\lambda(t)$ and $B(t)$, the computation resource allocation problem can be expressed as:

$$\begin{aligned} P1.2.1: \max_{f^e(t)} & \sum_{n=1}^{N} \log\left(1+\phi+l_n^{max,t}-l_n^t\right) \\ \text{s.t.} \quad & (11a),(11b),(11e),(11g) \end{aligned} \tag{14}$$

Optimization problem $P1.2.1$ with the convex constraints (11a), (11b), (11e) and (11g) is a joint convex problem, since $\partial^2 S_n^t(f^e(t))/\partial f^e(t)^2 \leq 0$. To tackle this problem, the Lagrangian function $L(f^e(t),\boldsymbol{\theta},\mu,\omega)$ is then introduced by:

$$\begin{aligned} L(f^e(t),\boldsymbol{\theta},\mu,\omega) = & \sum_{n=1}^{N}[\log\left(1+\phi+l_n^{max,t}-l_n^t\right)- \\ & \theta_n(l_n^t-l_n^{max,t})] - \mu(E^{e,t}-Q) \\ & - \omega(\sum_{n=1}^{N}\beta_n^t\lambda_n^t(2-\lambda_n^t)f_n^{e,t}-F_e) \\ & \text{s.t.} \quad (11g) \end{aligned} \tag{15}$$

where the coefficients $\boldsymbol{\theta}$, μ, ω represent the Lagrange multiplier associated with the task latency threshold, maximum energy constraint and computation capacity constraint, respectively. The dual problem of $P1.2.1$ is:

$$\begin{aligned} & \min D(\boldsymbol{\theta},\mu,\omega) \\ & \text{s.t.} \quad \boldsymbol{\theta}\geq 0,\mu\geq 0,\omega\geq 0 \end{aligned} \tag{16}$$

where $D(\boldsymbol{\theta},\mu,\omega)=\max L(f^e(t),\boldsymbol{\theta},\mu,\omega)$. Indeed, the intrinsic convexity of $P1.2.1$ demonstrates that the optimization problem $P1.2.1$ possesses a zero dual gap and fulfills Slater's condition. Thus, we can effectively solve the dual problem $P1.2.1$ by employing the gradient method, which provides a practical approach to finding solutions. In particular, the Lagrange multipliers can be updated iteratively as follows:

$$\begin{aligned} &\theta_n(i+1)=[\theta_n(i)+c_1(l_n^t-l_n^{max,t})]^+ \\ &\mu(i+1)=[\mu(i)+c_2(E^{e,t}-Q)]^+ \\ &\omega(i+1)=[\omega(i)+c_3(\textstyle\sum_{n=1}^{N}\beta_n^t\lambda_n^t(2-\lambda_n^t)f_n^{e,t}-F_e)]^+ \end{aligned} \tag{17}$$

where $[\cdot]^+$ equals $max(0,\cdot)$, c_1,c_2,c_3 are positive gradient step sizes for each iteration, and i represents the current iteration in the optimization process. Upon acquiring the values for each Lagrange multiplier, we obtain $f_n^{e,t}$ by taking the first-order derivative of L with respect to $f_n^{e,t}$ and setting the result to zero.

2) Communication resource Allocation: Derived from $P1.2.1$, we can formulate the communication resource allocation problem, considering the fixed $f^e(t)$, service caching decision $\delta(t)$ and task offloading decision $\lambda(t)$, as follows:

$$P1.2.2: \max_{B(t)} \quad \sum_{n=1}^{N} \log\left(1+\phi+l_n^{max,t}-l_n^t\right) \tag{18}$$

$$\text{s.t.} \quad \sum_{n=1}^{N}\beta_n^t\lambda_n^t(2-\lambda_n^t)b_n^t = W \ \forall t\in\mathcal{T} \tag{18a}$$

$$(11a),(11f)$$

Given that the allocation of bandwidth to the UDs has no discernible impact on the energy consumption within the ES, we proceed to migrate constraint (11d) to become constraint (18a). It is evident that the optimization problem $P1.2.2$ is convex. To address this problem, we still employ the primal-dual Lagrangian method here. The Lagrangian function takes the following form:

$$\begin{aligned} L'(B(t),\boldsymbol{\vartheta},\varepsilon) = & \sum_{n=1}^{N}[\log\left(1+\phi+l_n^{max,t}-l_n^t\right) \\ & - \vartheta_n(l_n^t-l_n^{max,t})] \\ & - \varepsilon(\sum_{n=1}^{N}\beta_n^t\lambda_n^t(2-\lambda_n^t)b_n^t-W) \\ & \text{s.t.} \quad (11f) \end{aligned} \tag{19}$$

where $\boldsymbol{\vartheta},\varepsilon$ denotes the Lagrangian multipliers. In accordance with the Karush-Kuhn-Tucker (KKT) condition, it is imperative that the following complementary slackness conditions are met.

$$\begin{aligned} &\partial L'(B(t),\boldsymbol{\vartheta},\varepsilon)/\partial B(t)=0 \\ &\vartheta_n(l_n^t-l_n^{max,t})=0 \\ &l_n^t-l_n^{max,t}\leq 0 \\ &\textstyle\sum_{n=1}^{N}\beta_n^t\lambda_n^t(2-\lambda_n^t)b_n^t-W=0 \\ &\boldsymbol{\vartheta}\geq 0. \end{aligned} \tag{20}$$

By solving the aforementioned slackness conditions, we can derive the optimal solution. The specific solution algorithm for the whole resource allocation problem referred as CCRA is shown in Algorithm 1.

B. Service Caching and Task Offloading Based on GA Method

In this subsection, we present a genetic algorithm which is designed to provide efficient search directions. Firstly, we encode the caching and offloading decisions, and then, based on the optimized problem and constraints, we calculate the fitness function for each candidate caching decision and offloading decision through algorithm 1. Then individuals are operated for crossover and mutation. Finally, the elite strategy is implemented to select individuals based on their fitness values. Specifically, the fitness of each individual can be

Algorithm 1: CCRA

Input:
$\delta(t), \lambda(t)$: the service caching/task offloading decision;
tol_l, tol_b, tol_f: the tolerance error;
Output:
$B(t), f^e(t)$: the optimal communication/computation resource allocation strategy;

Initialization:
Set feasible solutions for $f^e(t)$ and $B(t)$ randomly;
$i, j \longleftarrow 0$;
repeat
  repeat
    Update Lagrange multipliers $\theta_n(i+1)$, $\mu(i+1)$, and $\omega(i+1)$ by (17) based on $B(t)(j)$, respectively;
    Calculate $f^e(t)(i+1)$;
    $i \longleftarrow i+1$;
  until $|\theta_n(i) - \theta_n(i-1) < tol_l|$, $|\mu(i) - \mu(i-1) < tol_l|$ & $|\omega(i) - \omega(i-1) < tol_l|$;
  Calculate $B(t)(j+1)$ based on (20);
  $j \longleftarrow j+1$;
until $|b_n^t(j) - b_n^t(j-1) < tol_b|$ & $|f_n^{e,t}(j) - f_n^{e,t}(j-1) < tol_f|$;
Return $B(t)$, $f^e(t)$

calculated as $\sum_{n=1}^{N} S_n^t$ after determining the allocation of computational and communication resources.

We try to seek a comprehensive solution that not only reduces the average task completion time but also ensures fairness among users in this paper. Accordingly, the joint optimization problem can be solved by GACORA outlined in Algorithm 2.

Algorithm 2: GACORA

Input:
$iter_{max}$: the maximun itreation number;
P: the original population;
Z: the population size;
Output:
$\delta(t), \lambda(t)$: the optimal service caching/task offloading decision;
$B(t), f^e(t)$: the optimal communication/computation resource allocation strategy;

$iter \longleftarrow 0$;
while $iter < iter_{max}$ **do**
  Obtain optimal resource allocation decision for each individual by calling Algorithm 1;
  Calculate the fitness for each individual in P;
  Obtain offspring candidate population P' by crossover and mutation operations on P;
  Calculate the fitness values for individuals in P';
  Construct new population P from previous P and P' by adopting the elite preservation strategy;
  $iter \longleftarrow iter + 1$;
end while
return $\delta(t)$, $\lambda(t)$, $B(t)$, $f^e(t)$

V. Simulation Results

In this section, the effectiveness of our proposed GACORA is systematically evaluated through extensive simulations in the dynamic scenarios with multiple user devices.

A. Comparison Algorithms

The comparative evaluation is conducted to assess the performance of the GACORA algorithm against the following four alternative approaches.

1) PC+CCRA: The service caching scheme is obtained based on the popularity (i.e. the requested number), and the CCRA algorithm is used to allocate communication and computational resources.
2) WC+CCRA: The algorithm utilizes GA approach and CCRA method, only without the assistance of cloud platform.
3) PC only: In adherence to the popularity of services, caching is performed on the edge server that uniformly allocates its communication and computational resources to each task offloaded to it.
4) LP only: All tasks are handled through local processing.

The assessment of the performance of these methods encompasses several crucial aspects, which are evaluated based on the following performance metrics: 1) average user satisfaction calculated according to (21); 2) average task latency as shown (22); 3) task completion rate in (23), where $\mathbb{I}(\cdot)$ is an indicator function; 4) the fairness between generated tasks, calculated according to (25).

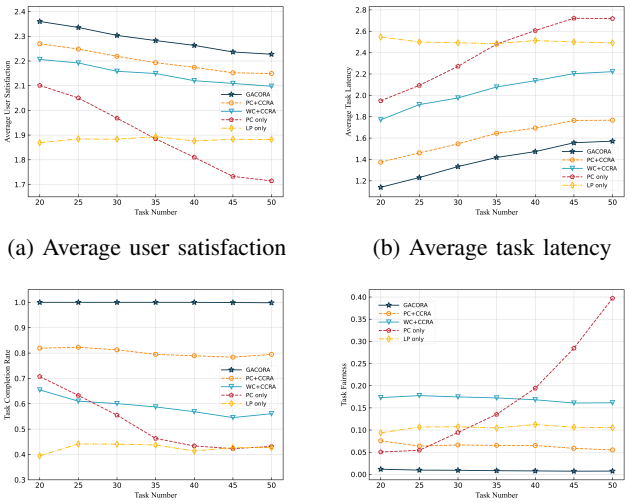
$$AS = \frac{\sum_{t=1}^{T}\sum_{n=1}^{N} S_n^t}{TN} \tag{21}$$

$$AL = \frac{\sum_{t=1}^{T}\sum_{n=1}^{N} l_n^t}{TN} \tag{22}$$

$$CR = \frac{\sum_{t=1}^{T}\sum_{n=1}^{N} \mathbb{I}(l_n^{max,t} - l_n^t \geq 0)}{TN} \tag{23}$$

$$fa_n = \max(\log\left(1 + \phi' + (l_n^{max,t} - L_n^t)/l_n^{max,t}\right), 0) \tag{24}$$

$$TF = \frac{1}{T}\sum_{t=1}^{T} \frac{(fa_n - \frac{\sum_{n=1}^{N} fa_n}{N})^2}{N-1} \tag{25}$$



(a) Average user satisfaction (b) Average task latency

(c) Task completion rate (d) Task fairness

Fig. 1. Performance evaluation of the GACORA

In this section, we will illustrate the impact of different user quantities on the aforementioned four metrics. From Fig. 1 (a), it becomes evident that as the number of UEs increases, the average user satisfaction begins to decline. This phenomenon can be attributed to the intensified competition for available computational and communication resources on the edge server as the user count rises. Consequently, the benefits users gain from offloading their computation tasks to the edge server decrease.

Fig. 1 (b) displays the average response time of various methods in our proposed cloud-edge system for different numbers of user devices. It is obvious that as the number of users increases, the average response time tends to rise for nearly all methods. However, our proposed method exhibits the most favorable average response time across different quantities of user devices.

In Fig. 1 (c), we can observe that as the number of tasks increases, all the methods except GACORA reveal fluctuation in task completion rate, and our proposed GACORA method obviously has the best performance, when the number of tasks is around 20-50, GACORA can nearly guarantee a hundred per cent completion rate of the tasks. This is because when computational resources are scarce, the CCRA algorithm can achieve a higher task completion rate by choosing appropriate scheduling decisions to achieve a balance between tasks. Comparing PC+CCRA with PC only, although both of them have same servicing caching and task offloading policy, without the help of CCRA method, which lacks the reasonable allocation of resources at the edge, the task completion rate of PC only method decreases from %11.5 to %36.3 compared with that of PC+CCRA method.

Fig. 1 (d) illustrates the fairness among users for different algorithms under varying task quantities, represented by variance. Therefore, smaller numerical values represent better method performance in terms of user fairness. From the graph, it is evident that the superiority of the GACORA algorithm remains pronounced. In the other algorithms, the differences in task completion times among users within the same group dramatically increase with the growth of user numbers, leading to inadequate fairness among multiple users.

VI. Conclusion

We proposed a two-level GA-based iterative approach to joint optimize cache decision, computation offload decision and resource allocation in dynamically changing cloud-edge scenarios. This method utilizes a Genetic Algorithm to select caching and offloading decisions. Based on the given caching and offloading decisions, we formulate the resource allocation problem as a convex optimization problem and iteratively find the optimal resource allocation solution. Compared to many baseline methods, this approach exhibits significant improvements in terms of average user satisfaction, average response time, task completion rate, and fairness among users. This study only considers a scenario with a single ES for offloading. In future work, we intend to explore service caching and computation offloading in scenarios with multiple ESs to address more complex situations, meeting the increasing demands of mobile data traffic and intelligent applications.

Acknowledgement

This work is supported by the National Natural Science Foundation of China under Grant Number 62071327, 62271486 and 62071470. Shuo Xiao and Huaming Wu are the corresponding authors.

References

[1] Y. Wu, H.-N. Dai, H. Wang, Z. Xiong, and S. Guo, "A survey of intelligent network slicing management for industrial iot: Integrated approaches for smart transportation, smart energy, and smart factory," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 2, pp. 1175–1211, 2022.

[2] C. Tang, H. Wu, R. Li, C. Zhu, and J. J. P. C. Rodrigues, "A systematic exploration of edge computing-enabled metaverse," *IEEE Network*, pp. 1–1, 2023.

[3] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, April 16-19, 2018*. IEEE, 2018, pp. 207–215.

[4] Y. Wu, H.-N. Dai, and H. Wang, "Convergence of blockchain and edge computing for secure and scalable iiot critical infrastructures in industry 4.0," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2300–2317, 2021.

[5] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surv. Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[6] C. Tang, W. Chen, C. Zhu, Q. Li, and H. Chen, "When cache meets vehicular edge computing: Architecture, key issues, and challenges," *IEEE Wirel. Commun.*, vol. 29, no. 4, pp. 56–62, 2022.

[7] J. Zhang, Y. Wu, G. Min, and K. Li, "Neural network-based game theory for scalable offloading in vehicular edge computing: A transfer learning approach," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–14, 2024.

[8] Y. Gong, K. Bian, F. Hao, Y. Sun, and Y. Wu, "Dependent tasks offloading in mobile edge computing: A multi-objective evolutionary optimization strategy," *Future Generation Computer Systems*, vol. 148, pp. 314–325, 2023.

[9] T. M. Ho and K. Nguyen, "Joint server selection, cooperative offloading and handover in multi-access edge computing wireless network: A deep reinforcement learning approach," *IEEE Trans. Mob. Comput.*, vol. 21, no. 7, pp. 2421–2435, 2022.

[10] X. Deng, J. Zhang, H. Zhang, and P. Jiang, "Deep-reinforcement-learning-based resource allocation for cloud gaming via edge computing," *IEEE Internet Things J.*, vol. 10, no. 6, March 15, pp. 5364–5377, 2023.

[11] P. Parastar, F. Pakdaman, and M. R. Hashemi, "FRAME-SDN: a fair resource allocation for multiplayer edge-based cloud gaming in SDN," in *Proceedings of the 25th ACM Workshop on Packet Video*. ACM, 2020, pp. 21–27.

[12] C. Tang, C. Zhu, H. Wu, Q. Li, and J. J. P. C. Rodrigues, "Toward response time minimization considering energy consumption in caching-assisted vehicular edge computing," *IEEE Internet Things J.*, vol. 9, no. 7, pp. 5051–5064, 2022.

[13] H. Zhou, K. Jiang, X. Liu, X. Li, and V. C. M. Leung, "Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1517–1530, 2022.

[14] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4377–4387, 2019.

[15] Z. Chen, W. Yi, A. S. Alam, and A. Nallanathan, "Dynamic task software caching-assisted computation offloading for multi-access edge computing," *IEEE Trans. Commun.*, vol. 70, no. 10, pp. 6950–6965, 2022.

[16] X. Li, Y. Qin, J. Huo, and W. Huangfu, "Heuristically assisted multiagent rl-based framework for computation offloading and resource allocation of mobile-edge computing," *IEEE Internet Things J.*, vol. 10, no. 17, pp. 15 477–15 487, 2023.