IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 20, 2020, accepted January 3, 2021, date of publication January 13, 2021, date of current version January 22, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3051278

Joint Computational Offloading and Data-Content Caching in NOMA-MEC Networks

LUAN N. T. HUYNH[1], QUOC-VIET PHAM[2], (Member, IEEE), TRI D. T. NGUYEN[1], MD. DELOWAR HOSSAIN[1], YOUNG-ROK SHIN[1], AND EUI-NAM HUH[1], (Member, IEEE)

[1]Department of Computer Science and Engineering, Kyung Hee University, Yongin 17104, South Korea
[2]Research Institute of Computer, Information and Communication, Pusan National University, Busan 46241, South Korea

Corresponding author: Eui-Nam Huh (johnhuh@khu.ac.kr)

This work was supported by the Institute for Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korean Government (MSIT) (Service mobility support distributed cloud technology) under Grant 2017-0-00294.

ABSTRACT Multi-access edge computing (MEC) can improve the users' computational capacity and battery life by moving computing services to the network edge. In addition, data-content caching on a MEC server improves the user quality of experience and decreases the backhaul network congestion. Moreover, non-orthogonal multiple access (NOMA) has recently been implemented to increase network throughput and capacity. Combining these techniques can improve the user performance and benefit the network. This paper investigates a combined computational offloading and data-content caching problem for NOMA-MEC systems. The aim was to achieve the minimum total completion latency of all users by jointly optimizing the offloading decision, caching strategy, computational resource, and power allocation. This satisfies the constraints within the scope of the potential violation for energy consumption, offloading decision, and the computation and storage capacity of the MEC server. The formulated problem is a mixed-integer non-linear programming and a non-convex problem. To solve this challenging problem, a block successive upper-bound minimization method was implemented to obtain efficient solutions. Numerous simulation results were presented to demonstrate the convergence and efficacy of the proposed algorithm. Compared with other schemes of all-offloading, local-only, and equal resources, our proposed algorithm can approximately reduce the total completion latency by 17.68%, 26.02%, and 70.98%.

INDEX TERMS Multi-access edge computing, non-orthogonal multiple access, block successive upper-bound minimization, computational offloading, data-content caching.

I. INTRODUCTION

Data traffic is increasing astronomically due to the explosive growth of smart mobile equipment and Internet of Things (IoT) devices, which are driving the development of many emerging applications, such as virtual reality (VR)/augmented reality(AR), interactive gaming, remote healthcare systems, surveillance, and autonomous driving. These applications are typically computation-intensive, latency-critical, and energy-consuming. On the other hand, mobile devices often have limited computational power and limited battery capacity [1], [2]. Handling the computation-intensive demands on the consumer end is a challenging task. Multi-access edge computing (MEC) was proposed and developed by ETSI in 2014 to tackle these challenges [1], [2]. It has emerged as an innovative computation paradigm designed to support the development of new computer applications, which provides data, cloud computing capabilities, and computation technology services from centralized cloud computing to the network edge [1]. In MEC systems, the users can offload their computation-intensive and delay-sensitive tasks to nearby MEC servers that are attached at the base stations for remote task execution. This scheme improves the computational capability and reduces the delay execution of users [1]. On the other hand, a resource allocation strategy at the MEC server should also be considered because the computational resources of the MEC server are limited. The computational resource of a MEC server is not always sufficient to support all users. Hence, inefficient resource allocation techniques would increase energy consumption and delay experienced by mobile users.

On the other hand, the benefits of edge caching (also known as caching at the edge) in handling a significant increase in mobile data traffic have been studied [2], [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Songwen Pei.

The data-content caching involves storing frequently used content or a database/library related to services deployed in MEC servers. Caching the requested data-content at the network edge can reduce computational delay and improve the quality-of-service (QoS) and the quality-of-experience (QoE) of users. For example, there should be more associated data-content with some applications, such as VR, online gaming, remote healthcare, autonomous driving, in addition to the data or content needed for the computation-intensive task [4], [5]. In museums, tourists prefer to use AR to feel a better sense of reality. Consequently, more real-time services can be provided if more data can be stored on the MEC server in this area. Significant research on the advantages of caching and computing offloading in MEC systems have been performed [2]–[11]. Some requested data-content can be downloaded from the MEC server for local execution. On the other hand, the communication resources (i.e., bandwidth and power budget) are typically limited. Therefore, the downlink resource is also a key element in reducing the delay if the data-content of a computational task needs to be downloaded from the edge server.

Moreover, as the number of IoT devices is massive, the use of multiple-access techniques will be needed to improve the performance of MEC systems. Non-orthogonal multiple access (NOMA) has more benefits for next-generation wireless networks (i.e., 5G and 6G), which enhances the system throughput and capacity [1], [12]–[15]. NOMA technology allows multiple users to use orthogonal resources simultaneously compared to traditional orthogonal multiple access (OMA) technologies. NOMA can handle more users than the number of possible sub-carriers, resulting in numerous potentials, including huge connectivity, reduced delay, higher spectral performance, and relaxed channel feedback [14]–[17]. Using the same frequency resources, multiple users may offload their computational tasks to the MEC server simultaneously. Therefore, integrating NOMA into MEC systems can enhance the computation-offloading efficiency and performance, i.e., reduce the delay and energy consumed in computational offloading, increasing the number of offloading users [13]–[15], [17]. Several studies on NOMA-MEC systems have centered on the issue of computational offloading optimization. The majority of papers attempted to minimize the latency and consumed energy [12], [17]–[27]. On the other hand, most studies on NOMA-MEC systems do not consider the benefits of edge caching.

Several studies focused on either computational offloading with data-content caching in MEC systems or computational offloading in NOMA-MEC systems. Despite this, most of those studies overlooked the benefits of incorporating NOMA-MEC into computing and caching systems to increase the offloading efficiency. Different from existing works, we combined NOMA and MEC for computational offloading and data-content caching in this paper. To decrease the latency of offloading and improve the performance of the MEC system, we propose a joint computational offloading policy, data-content caching strategy, computational resource, and downlink and uplink resource allocation in NOMA-MEC systems. An effective algorithm was developed to minimize the total completion latency of the users. The main contributions and features offered by our work can be summarized as follow.

- First, this study considered a NOMA-MEC system and edge caching network to achieve a minimum total completion latency for all users on this system. Mathematically, a joint problem of the offloading decision, caching strategy, computational resource, and power allocation, which applies to both the downlink and uplink NOMA transmissions, was formulated. The algorithm met the constraints within the scope of the energy consumption efficiency, offloading decision, and computation and storage capacity of the edge server.
- Second, the formulated problem is a mixed-integer nonlinear programming (MINLP) and a non-convex problem. Therefore, achieving the optimal solution in polynomial time is a great challenge. To solve this challenging problem, the block successive upper-bound minimization (BSUM) was used to develop an efficient algorithm for a high-quality solution.
- Finally, based on the simulation results, the effectiveness of the proposed algorithm was confirmed. The proposed algorithms can reduce the total completion time compared with other schemes. Furthermore, the proposed algorithm can converge to the near-optimal solution at a sub-linear convergence rate.

The remainder of this paper is structured as follows. Section II summarizes the work involved, and Section III presents the system model and problem formulation. Section IV introduces the proposed algorithm to solve the problem of optimization, and Section V discusses the outcomes of the simulation. This paper is finally concluded in Section VI.

II. RELATED WORKS

More recently, major research has been conducted to study the advantages of caching and computing offloading in MEC systems. Some studies examined the computational offloading decision, caching strategy, and resource allocation in MEC systems. Hao *et al.* [6] studied the joint design of the offloading decision and task caching strategy to minimize the total energy consumed by mobile users. This paper introduces a new concept of task caching, in which the computation-intensive tasks were cached on the edge network. Task caching has been mentioned elsewhere [9], where the authors considered jointly the task offload, task caching, and security service to minimize the energy consumed and security breach cost in MEC systems for IoT applications. Bi *et al.* [10] investigated a single MEC server to assist the user in processing a set of computational tasks. They optimized task offloading, service caching, and resource allocation to minimize the user's latency and energy consumption. Another study [3] designed an offloading policy by caching the popular computational results in MEC networks

to reduce the overall computing time of the users. Deep reinforcement learning (DRL) was studied to minimize the average latency of fog-enabled IoT services by optimizing the offloading decision-making and content-caching strategies [5]. Yang *et al.* [7] designed a collaborative offloading decision and data caching policy to minimize the total delay of mobile users in a hybrid mobile cloud/edge computing system. This study assumed an equal allocation of resources, such as computing resources and communication resources. Wang *et al.* [8] integrated a computational offloading policy, content caching strategy, computation resource allocation, and radio spectrum allocation to maximize the revenue of MEC systems. On the other hand, the proposals for optimizing the total latency or power consumption were inefficient. The optimal offloading decision, caching policy, and allocation of bandwidth resources were also designed to minimize the total latency of mobile users in the MEC system [4]. Nevertheless, they did not consider the requested content download to users. A previous study [2] investigated the combination of communication, computation, control, and caching (4C) in big data MEC to minimize the total users' latency and maximize the backhaul bandwidth. Wang *et al.* [11] minimized the total cost in terms of the completion latency and the charge based on optimizing computational offloading decision and computation and downlink resource allocation. In this article, the authors referred to the content cache service on the BS to download to the users to perform tasks, but they did not consider the content caching decisions on the BS.

Numerous studies on NOMA-MEC systems integration have centered on optimizing computational offloading and resource allocation. A significant part of the current papers has been proposed to minimize the completion latency and consumed energy. Some researchers reduced the completion latency of MEC offloading in existing NOMA related studies. Ding *et al.* [17] proposed a hybrid-NOMA scheme with two mobile devices. The optimization problem was solved using Dinkelbach's method and Newton's method. An efficient workload, offloading, and downloading algorithm for optimizing the duration was proposed [18]. Some studies minimized the total power consumption of MEC offloading. Wang *et al.* [19] solved the optimization problem of local CPU-cycle frequency, transmit power, and MD rates using the Lagrange dual method, branch-and-bound, greedy method, and convex relaxation. On the other hand, the QoS specifications of all users may not be adequate when the number of users is increasing. A hybrid-NOMA strategy was proposed [20] in which geometric programming was applied to optimize the power and time allocation. Pan *et al.* [21] optimized power allocation, time allocation, and task assignment using the successive convex approximation algorithm. Yang *et al.* [22] considered multiple users in different groups and proposed an iterative algorithm with low complexity to balance latency and power consumption. A coalition game was developed to minimize the sum computation overhead by optimizing the computational offloading decision

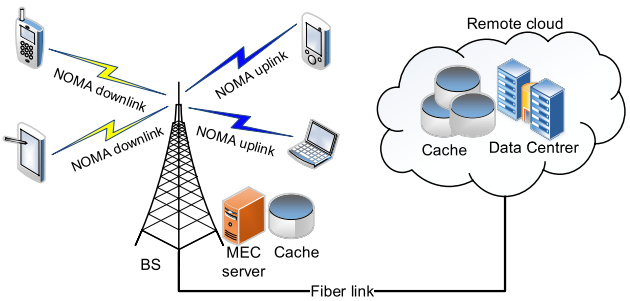

FIGURE 1. Computational offloading and data-content caching for NOMA-MEC system model.

and sub-carrier assignment on multi-carrier NOMA-MEC systems [23]. Diao *et al.* [24] examined the D2D-assisted and NOMA-based MEC network to reduce the total cost of users in terms of latency and energy consumed. Nduwayezu *et al.* [25] proposed an algorithm using deep reinforcement learning to maximize the total computational rate for multi-carrier NOMA-MEC systems by jointly optimizing the computation offloading decision-making and sub-carrier allocation. Fang *et al.* [26] minimized the overall task latency of mobile users for NOMA-MEC systems. They optimized the NOMA to support the decrease in latency in the MEC system. Pham *et al.* [27] maximized task offloading gains by jointly considering computational offloading, resource allocation, sub-carrier assignment, and power control in NOMA-MEC systems. Hao *et al.* [12] introduced a hybrid NOMA-MEC system to enhance the computation service for Sixth-Generation (6G) wireless networks. The multilevel programming method was applied to minimize the energy consumption by mutually optimizing the offloading strategy, time slot scheduling, and power control.

This paper shows that important requirements were often overlooked in previous studies and formulate a joint problem of computation-intensive task offloading, caching strategy, computational resource, and uplink and downlink resource allocation in NOMA-MEC systems. In addition, the BSUM algorithm was used to solve the proposed optimization problem.

III. NETWORK MODEL AND PROBLEM FORMULATION

This section presents the system model and formulates the process for optimizing offloading decision-making, caching decision, allocation of communication resources, and allocation of computing resource in NOMA-MEC system for minimizing completion latency.

A. NETWORK SCENARIO

Fig. 1 presents the scenario considered in this work. This NOMA-MEC network consists of a BS, a remote cloud, and users. BS is co-located with a MEC server, which can provide users with computational offloading. On the other hand, the computational resources of MEC servers are limited. The MEC server also has finite-capability storage that can be used to store a number of selected data-contents. Consequently, the caching decision of the BS affects the user

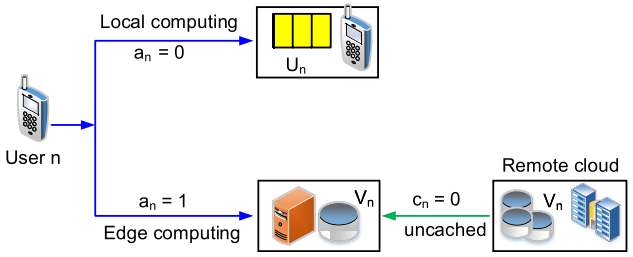

FIGURE 2. Illustration of a user execution.

performance significantly. The BS is connected to the remote cloud using a wired network. The remote cloud has large storage and computing capacities. Here, this study considered that each user has a computation-intensive task, which is inseparable. That is, the task can be performed using either local computing or edge server processing. This study did not consider performing the computational task on a remote cloud because of the increased latency. Optimization of this issue will be addressed in future work.

$\mathcal{N} = \{1, 2, \ldots, N\}$ is the set of users, with N being the number of users. A computation-intensive task is represented by the tuple $I_n = \{U_n, W_n\}$, where U_n is the input data size of the computation-intensive task and W_n is the number of CPU cycles needed to accomplish a computation-intensive task. As mentioned above, user n requests data-content when performing its computation-intensive task. The requested data-content is stored in the remote cloud or cached at the BS [4], [7]. Caching data-content at the network edge can avoid frequent data-downloading over the backhaul networks and reduce latency. On the other hand, only some data-content is cached at the BS owing to the limited storage space of the BS. The requested data-content for the computational task that is not cached at the BS must be obtained from the remote cloud. In particular, BS can distribute cached data to place when computational data is required. In contrast, the uncached requested data-content must be obtained to place from the remote cloud via the backhaul link. Fig. 2 presents an illustration of a user execution. V_n refers to the size of the data-content. If user n performs locally, the requested data-content V_n is downloaded to the user. If user n performs at the BS, the input data size of the task U_n is offloaded to the BS.

B. CACHING MODEL

In this model, $c_n \in \{0, 1\}, \forall n \in \mathcal{N}$ is denoted as the data-content caching decision, where $c_n = 1$ and 0 if the requested data-content by user n is cached at the BS and the remote cloud, respectively. Unlike a large and diverse remote cloud resource, limited storage and executing resource for the BS allow only some data-contents to be cached. Therefore, C_{cache} is maximum caching capability of the BS due to limited caching space. Consequently, caching decisions are constrained as follows:

$$\sum\nolimits_{n\in\mathcal{N}} c_n V_n \leq C_{\text{cache}}. \tag{1}$$

The requested data-content by the user has different popularity. Here, it was assumed that the data-content popularity is designed as the Zipf distribution [4], [28]. Therefore, the popularity of the i^{th} popular data-content demanded by user n can be expressed as

$$p_n(i) = \frac{1/i^{\kappa}}{\sum_{i=1}^{N_v}(1/i^{\kappa})}, \tag{2}$$

where N_v is the total types of data-content caching, and κ is the parameter of the Zipf popularity distribution.

C. COMMUNICATION MODEL

This study considered single-carrier NOMA for joint computation offloading and data caching at the network edge. Uplink and downlink utilize the time division duplex (TDD). In TDD mode, both the uplink and downlink transmissions use the same frequency spectrum [21]. Moreover, there is no interference between uplink and downlink transmission. Similar to previous studies such as [29], we assumed perfect channel state information (CSI) or the order of the instantaneous channel gain at the transmitter side, e.g., users at the uplink stage and BS at the downlink stage. Moreover, we also assume that the locations of users are fixed during the offloading period, but the locations can change across different periods.

Without loss of generality, users are ordered as $h_1 \leq h_2 \ldots \leq h_n$, where h_n is channel gain from user n to the BS. Power domain NOMA multiplexing was applied to superimpose multiple signals. Successive interference cancellation (SIC) and power constraints for efficient SIC were adapted to decode the superimposed signals at the receivers [21], [23], [26]. According to [21], [23], [26], the decoding order utilizes the increasing order of the channel gains in the downlink NOMA. In contrast, the decoding order in the uplink NOMA adopts the decreasing order of the channel gains [21], [23]. With TDD, channels gain of the uplink and downlink are the same. When user n offloads the computation-intensive task to the BS for remote execution, the user's uplink achievable data rate via the wireless connection can be defined as

$$R_n^{ul} = B\log_2\left(1 + \frac{h_n p_n^{ul}}{\sigma^2 + \sum\limits_{j\in\mathcal{N}:h_j<h_n} h_j p_j^{ul}}\right), \forall n \in \mathcal{N}, \tag{3}$$

where p_n^{ul} is the transmit power of user n for uplink transmission to the BS; B is the bandwidth, and σ^2 is the noise power spectral.

The BS can deliver the required data-content to process a task to the corresponding user. The downlink achievable data rate between the user n and BS is calculated as

$$R_n^{dl} = B\log_2\left(1 + \frac{h_n p_n^{dl}}{\sigma^2 + \sum\limits_{j\in\mathcal{N}:h_j>h_n} h_n p_j^{dl}}\right), \forall n \in \mathcal{N}, \tag{4}$$

where p_n^{dl} denotes the transmit power to be allocated by the BS for downlink transmission to user n.

For backhaul communication, r^{bh} was denoted as the average data transmission rate of the backhaul link between the remote cloud and the BS. If the requested data-content by user n is cached in the BS, the system will be compensated for by reducing the backhaul latency or alleviating the backhaul bandwidth [4], [8]. According to (2), the reduced backhaul bandwidth for caching the i^{th} popular data-content demanded by user n can be achieved as

$$r_n^{bh} = r^{bh} p_n(i), n \in \mathcal{N}. \tag{5}$$

The transmission latency of the requested data-content transmitting from the remote cloud to the BS (backhaul latency) can be obtained as

$$T_n^{bh} = \frac{V_n}{r_n^{bh}}, n \in \mathcal{N}. \tag{6}$$

D. COMPUTATION MODEL

1) LOCAL EXECUTION

If user n executes its computational task locally, the total completion latency consists of a local processing time, downlink transmission time and backhaul latency. Let f_n^l denotes the local computing capability of user n. The local processing time of task I_n of user n can be given as

$$T_n^{el} = \frac{W_n}{f_n^l}, n \in \mathcal{N}. \tag{7}$$

The downlink transmission time between user n and the BS can be given as

$$T_n^{dl} = \frac{V_n}{R_n^{dl}}, n \in \mathcal{N}. \tag{8}$$

From equations (6), (7), and (8), the total completion latency experienced by the user can be written as

$$T_n^l = T_n^{dl} + (1 - c_n) T_n^{bh} + T_n^{el}, n \in \mathcal{N}. \tag{9}$$

The corresponding energy consumption of user n for local processing can be calculated as using equation (10):

$$E_n^l = \zeta W_n (f_n^l)^2, n \in \mathcal{N}, \tag{10}$$

where ζ is the switching capacitance depending on the CPU's chip architecture, which is set to $\zeta = 5 \times 10^{-28}$.

2) EDGE OFFLOADING

Most computation tasks should be offloaded to the BS for remote execution owing to the limited computing capability of the user. When user n executes its computational task in the BS, the total completion latency of the user consists primarily of the uplink transmission time, processing time of the MEC server, and backhaul delay. This study neglected the downloading time and the energy consumed by computation results from the BS to the user because its size is much smaller than the size of the input computation data [6], [11]. In addition, our current work focuses on latency and energy consumption from the user perspective, and the MEC server is normally powered by electricity supplied from the grid. Therefore, we ignore the energy computation at the MEC server-side [6], [11].

Let f_n denote the resource allocation of the MEC server to the execution of computational task I_n. The edge processing time for the computational task of user n can be obtained as follows:

$$T_n^{er} = \frac{W_n}{f_n}, n \in \mathcal{N}. \tag{11}$$

The uplink transmission time for transmitting input data U_n from user n to the BS is given as

$$T_n^{ul} = \frac{U_n}{R_n^{ul}}, n \in \mathcal{N}. \tag{12}$$

According to equations (6), (11), and (12), the total completion latency of user n when processing in the BS can be expressed as

$$T_n^r = T_n^{ul} + (1 - c_n) T_n^{bh} + T_n^{er}, n \in \mathcal{N}. \tag{13}$$

For task execution on the MEC server, the energy consumption of user n is only calculated from by the transmission energy consumed for offloading the task. Therefore, the energy consumption of user n can be computed as

$$E_n^r = p_n^{ul} T_n^{ul} = p_n^{ul} \frac{U_n}{R_n^{ul}}, n \in \mathcal{N}. \tag{14}$$

E. PROBLEM FORMULATION

$a_n \in \{0, 1\}, \forall n \in \mathcal{N}$, was defined as the offloading decision of user n, herein $a_n = 1$ if user n offloads its computational task to the BS and $a_n = 0$ otherwise. From Eqs. (9) and (13), the total completion latency experienced by user n can be represented as

$$T_n = a_n T_n^r + (1 - a_n) T_n^l, n \in \mathcal{N}. \tag{15}$$

This study aimed to achieve the minimum total completion latency of all users in NOMA-MEC systems by optimizing the binary offloading decision, caching decision, computational resource, and power allocation jointly. This problem can be formulated mathematically as follows:

$$P: \min_{\{\boldsymbol{a}, \boldsymbol{c}, \boldsymbol{f}, \boldsymbol{p}^{ul}, \boldsymbol{p}^{dl}\}} \sum_{n=1}^{N} T_n \tag{16a}$$

$$\text{subject to} \sum_{n \in \mathcal{N}} f_n \le f_0, \tag{16b}$$

$$f_n \ge 0, \ \forall n \in \mathcal{N}, \tag{16c}$$

$$0 \le p_n^{ul} \le p_n^{\max}, \ \forall n \in \mathcal{N}, \tag{16d}$$

$$\sum_{n \in \mathcal{N}} p_n^{dl} \le p_0, \tag{16e}$$

$$0 \le p_n^{dl} \le p_0, \ \forall n \in \mathcal{N}, \tag{16f}$$

$$\sum_{n \in \mathcal{N}} c_n V_n \le C_{\text{cache}}, \tag{16g}$$

$$(1 - a_n) E_n^l + a_n E_n^r \le E_n^{\max}, \ \forall n \in \mathcal{N}, \tag{16h}$$

$$a_n, c_n \in \{0, 1\}, \ \forall n \in \mathcal{N}. \tag{16i}$$

In this formulation, $\boldsymbol{a} = \{a_n\}$ is the offloading decision profile; $\boldsymbol{c} = \{c_n\}$ indicates the set of the data-content caching decision; $\boldsymbol{f} = \{f_n\}$ refers to the computational resource allocation policy; $\boldsymbol{p}^{ul} = \{p_n^{ul}\}$ defines the set of uplink transmit power of users, and $\boldsymbol{p}^{dl} = \{p_n^{dl}\}$ denotes the set of the downlink transmit power allocated by the BS. The set of users whose tasks are offloaded to the MEC server is denoted as $\mathcal{N}_{\text{off}} = \{n \in \mathcal{N} | a_n = 1\}$, and $\mathcal{N}_{\text{loc}} = \{n \in \mathcal{N} | a_n = 0\}$ denotes as the set of users performing their tasks locally, respectively. If $n \notin \mathcal{N}_{\text{off}}, f_n = 0$, i.e., user n executes the task locally. p_0 and $p_n^{\max}$ are the maximum downlink transmit power provided by the BS and maximum uplink transmit power of user n, respectively. $E_n^{\max}$ is the maximum allowable energy consumption of user n. f_0 is denoted as the maximum computational resource of the MEC server. Constraints (16b) and (16c) ensure that the computing resource for the computational offloading tasks is positive, and the total computing resource assigned to all offloading users does not exceed maximum computing capacity of the MEC server. Constraint (16d) means that the power assigned to the task of each user cannot exceed its maximum value. Constraints (16e) and (16f) suggest that the total downlink power resource for the users is limited by the overall transmission power budget of the BS. Constraint (16g) states that the total data cache on the BS cannot exceed its maximum caching capability. Constraint (16h) guarantees the the energy consumption of each user is limited. Constraint (16i) indicates the binary offloading decision and data-content caching strategy.

IV. PROPOSED ALGORITHM

A. PROBLEM DECOMPOSITION

Problem state in (16) is a non-convex problem caused by inter-cell interference and the variables (i.e., $\boldsymbol{a}, \boldsymbol{c}, \boldsymbol{f}, \boldsymbol{p}^{ul}$, and $\boldsymbol{p}^{dl}$) are linked together in the objective function. In addition, some constraints referred to (16b) − (16i) are non-linear and combine the continuous variables (i.e., $\boldsymbol{f}, \boldsymbol{p}^{ul}$, and $\boldsymbol{p}^{dl}$) and binary binary variables (i.e., $\boldsymbol{a}$, and $\boldsymbol{c}$). The aforementioned optimization problem is a non-convex, mixed-integer non-linear programming (MINLP) problem, which is typically an NP-hard problem. Problem (16) is quite difficult to solve optimally because of the complex combination of optimization variables and composite compositing features.

To tackle this problem, the original problem was decomposed into two sub-problems and solved alternately. Firstly, computing resource allocation $\boldsymbol{f}$ was solved using the Karush–Kuhn–Tucker (KKT) optimality conditions. The joint offloading decision, caching strategy, power uplink, and downlink resource allocation (i.e., $\boldsymbol{a}, \boldsymbol{c}, \boldsymbol{p}^{ul}$, and $\boldsymbol{p}^{dl}$) problem was addressed using BSUM method [30]. Owing to its advantages, BSUM has been used to solve many complex optimization problems [2], [28]. The results showed that BSUM is an effective algorithm for achieving a high-quality solution.

B. COMPUTING RESOURCE ALLOCATION

Given the computational offloading decision, caching decision, power uplink, and downlink resource allocation ($\boldsymbol{a}, \boldsymbol{c}, \boldsymbol{p}^{ul}$, and $\boldsymbol{p}^{dl}$), and after ignoring all parts of the objective function and the condition unrelated $\boldsymbol{f}$, the following optimization problem (16) was obtained as follows:

$$P1: \min_{\boldsymbol{f}} \sum_{n \in N_{\text{off}}} \frac{W_n}{f_n} \tag{17a}$$

$$\text{s.t.} \sum_{n \in \mathcal{N}_{\text{off}}} f_n \leq f_0, \tag{17b}$$

$$f_n > 0, \ \forall n \in \mathcal{N}_{\text{off}}. \tag{17c}$$

$h(\boldsymbol{f})$ is denoted as the objective function (17a). The Hessian matrix of $h(\boldsymbol{f})$ with respect to $\boldsymbol{f}$ consists of elements either $\frac{\partial^2 h}{\partial f_n^2} = (\frac{2W_n}{f_n^3}) > 0$ or $\frac{\partial^2 h}{\partial f_n \partial f_m} = 0$ $(n \neq m)$. Thus, the Hessian matrix is a semi-definite positive matrix. Moreover, the constraints (17b) is linear, and $\mathbf{f}$ is continuous variable. Hence, the problem in (17) is convex problem.

In order to obtain solution for (17), we first derive the Lagrangian function as follows:

$$L(\boldsymbol{f}, \boldsymbol{\lambda}) = \sum_{n \in N_{\text{off}}} \frac{W_n}{f_n} + \lambda \left(\sum_{n \in N_{\text{off}}} f_n - f_0 \right), \tag{18}$$

We then take the partial derivative of (18) with respected to $\mathbf{f}$ as follows:

$$\frac{\partial L(\boldsymbol{f}, \boldsymbol{\lambda})}{\partial f_n} = -\frac{W_n}{f_n^2} + \lambda. \tag{19}$$

Based on the KKT conditions [31], with $\boldsymbol{\lambda} > 0$, we can derive the close-form solution for (17) as follows:

$$f_n^* = \frac{f_0 \sqrt{W_n}}{\sum_{n \in \mathcal{N}_{\text{off}}} \sqrt{W_n}}. \tag{20}$$

C. JOINT OFFLOADING DECISION, CACHING STRATEGY, AND POWER RESOURCE ALLOCATION

Given $\boldsymbol{f}$, there is the following problem of $\boldsymbol{a}, \boldsymbol{c}, \boldsymbol{p}^{ul}$, and $\boldsymbol{p}^{dl}$. When $a_n = 0, f_n = 0$ and the objective function is not continuous. To solve this issue, when $a_n = 0, f_n = \varepsilon$, where ε is small enough and can approach 0 arbitrarily. Here, the BSUM approach was used to tackle the proposed issue. BSUM is a distributed algorithm that enables a parallel computation. The BSUM method allows a non-convex problem to decomposed into small sub-problems that can be addressed independently using convex optimization. The BSUM method guarantees convergence to the fixed points of the non-convex problem [30].

1) OVERVIEW OF BSUM APPROACH

The BSUM algorithm has advantages over centralized algorithms in both the solution speed and problem decomposition capability. An overview of the BSUM algorithm is first presented for more clarification. The following block-structured optimization problem [30] was examined:

$$\begin{aligned} &\min_{\boldsymbol{y}} \ h(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_m) \\ &\text{s.t.} \ \boldsymbol{y}_m \in \mathcal{W}_m, \\ &\quad \forall m \in \mathcal{M}, \ m = 1, 2, \ldots, M, \end{aligned} \tag{21}$$

where $h(.)$ is the continuous function; the feasible solution set $\mathcal{W} := \mathcal{W}_1 \times \mathcal{W}_2 \times \cdots \times \mathcal{W}_m$ are a close convex set, and $\boldsymbol{y}_m$ is a block variable. A common method to solve problem (21) is to apply the block coordinate descent (BCD) method; a single block of variables is optimized by solving the optimal problem at each iteration t:

$$\boldsymbol{y}_m^{(t)} = \underset{\boldsymbol{y}_m \in \mathcal{W}_m}{\operatorname{argmin}} \, h(\boldsymbol{y}_m, \, \boldsymbol{y}_{-m}^{(t-1)}), \tag{22}$$

where $\boldsymbol{y}_{-m}^{(t-1)} := (\boldsymbol{y}_1^{(t-1)}, \ldots, \boldsymbol{y}_{m-1}^{(t-1)}, \, \boldsymbol{y}_{m+1}^{(t-1)}, \ldots, \boldsymbol{y}_m^{(t-1)}), \boldsymbol{y}_j^{(t)} = \boldsymbol{y}_j^{(t-1)}$ for $m \neq j$.

Both problems (21) and (22) are difficult to solve, in particular (21)) is a non-convex function, and the convergence cannot always be guaranteed using the BCD method. The BSUM algorithm was introduced to overcome these issues [30]. In BSUM algorithm, $f_m(\boldsymbol{y}_m, z)$ is an upper-bound approximate function of the objective function $h(\boldsymbol{y}_m, z_{-m})$ for the block m at a given feasible point $z \in \mathcal{W}$. To guarantee the convergence, the upper-bound approximate function $f_m(\boldsymbol{y}_m, z)$ satisfies the following conditions:

Assumption 1: 1) $f_m(\boldsymbol{y}_m, \boldsymbol{y}) = h(\boldsymbol{y})$,
2) $f_m(\boldsymbol{y}_m, z) \geq h(\boldsymbol{y}_m, z_{-m})$,
3) $f_m'(\boldsymbol{y}_m, z; \boldsymbol{p}_m)|_{\boldsymbol{y}_m = z_m} = h'(z; \boldsymbol{p}), z_m + \boldsymbol{p}_m \in \mathcal{W}_m$,
4) $f_m(\boldsymbol{y}_m, z)$ is continuous in $(\boldsymbol{y}_m, z), \ \forall m$.

The first and second assumptions (i.e., 1(1) and 1(2)) ensure that the upper-bound approximate function is an upper-bound function of the objective function $h(x)$. Assumption 1(3) suggests that the first-order derivative in the direction $\boldsymbol{p}_m$ of the approximate function is the same as the objective function. Finally, Assumption 1 (4) then guarantees the approximate function should be continuous for all block variables. In BSUM, the most widely used technique for selecting the upper-bound approximate function is the quadratic upper bound, linear upper bound, and Jensen's upper bound [30]. For simplicity of introduction, the accompanying upper-bound approximate function, which is described by adding to the objective function a quadratic penalty, was used:

$$f_m(\boldsymbol{y}_m, z) = h(\boldsymbol{y}_m, z_{-m}) + \frac{\mu}{2} \|(\boldsymbol{y}_m - z_m)\|^2, \tag{23}$$

where $\mu > 0$ is a constant parameter. Let $\mathcal{M}^t$ denote a set of indexes at iteration t. The BSUM algorithm tackles the upper-bound approximate function in (23) with the following update at iteration t:

$$\begin{cases} \boldsymbol{y}_m^{(t)} = \underset{\boldsymbol{y}_m \in \mathcal{W}_m}{\operatorname{argmin}} \, f_m(\boldsymbol{y}_m, \boldsymbol{y}^{(t-1)}), & \forall m \in \mathcal{M}^t, \\ \boldsymbol{y}_j^{(t)} = \boldsymbol{y}_j^{(t-1)}, & \forall j \notin \mathcal{M}^t. \end{cases} \tag{24}$$

The block indexes selected at each iteration can be applied in various ways, such as random selection, cyclic rule, and Gauss-Southwell [30]. Compared to the BCD algorithm, the uniqueness of the solution in each iteration is not required by BSUM. Algorithm 1 presents the standard BSUM algorithm. Specifically, starting from a feasible point $\boldsymbol{y}^{(0)}$, BSUM algorithm generates a sequence of enhanced solutions and finally converges to a stationary point when the convergence criterion ϵ is met. Based on [30], [32], BSUM algorithm converges to the ϵ-optimal solution and takes at most $\mathcal{O}(\log(1/\epsilon))$ steps, i.e., a sub-linear convergence.

Algorithm 1 A Pseudocode of the BSUM Algorithm

1: **Initialization:** Set the iteration index t = 0, stopping criteria $\epsilon > 0$, and find initial feasible solutions $\boldsymbol{y}^{(0)}$.
2: **repeat**
3: $t = t + 1$;
4: Select index set $\mathcal{M}^t$.
5: Let $\boldsymbol{y}_m^{(t)} = \underset{\boldsymbol{y}_m \in \mathcal{W}_m}{\operatorname{argmin}} \, f_m(\boldsymbol{y}_m, \, \boldsymbol{y}^{(t-1)}), \ \forall m \in \mathcal{M}^t$
6: Set $\boldsymbol{y}_j^{(t)} = \boldsymbol{y}_j^{(t-1)}, \forall j \notin \mathcal{M}^t$.
7: **until** $\|\frac{f_m^{(t)} - f_m^{(t-1)}}{f_m^{(t-1)}}\| \leq \epsilon$.
8: Then, consider $\boldsymbol{y}^* = \boldsymbol{y}^{(t)}$ as a solution.

2) PROPOSED SOLUTION

Given $\boldsymbol{f}$, to solve problem (16) using BSUM, the binary variables consisting of offloading decision and caching strategy are first relaxed into continuous ones, i.e., $0 \leq a_n \leq 1$ and $0 \leq c_n \leq 1$. Then, we can apply the BSUM algorithm to solve (16), which is guaranteed to converge to the stationary point by BSUM [30]. The optimizing problem (16) can be reformulated as follows:

$$P2: \min_{\{\boldsymbol{a}, \boldsymbol{c}, \boldsymbol{p}^{ul}, \boldsymbol{p}^{dl}\}} \sum_{n=1}^{N} T_n \tag{25a}$$

$$\text{subject to (16d)} - \text{(16h)}, \tag{25b}$$

$$a_n, c_n \in [0, 1], \ \forall n \in \mathcal{N}, \tag{25c}$$

To simplify the notation, the optimization problem is written concisely as

$$\mathcal{D}(\boldsymbol{a}, \boldsymbol{c}, \boldsymbol{p}^{ul}, \boldsymbol{p}^{dl}) \triangleq \sum_{n=1}^{N} T_n. \tag{26}$$

The constraints of both (25) and (26) are the same. Moreover, $\mathcal{A} \triangleq \{\boldsymbol{a} : (1 - a_n)E_n^l + a_n E_n^r \leq E_n^{\max}, a_n \in [0, 1]\}$, $\mathcal{C} \triangleq \{\boldsymbol{c} : c_n \in [0, 1], \sum_{n \in \mathcal{N}} c_n V_n \leq C_{\text{cache}}\}$, $\mathcal{P} \triangleq \{\boldsymbol{p}^{ul} : 0 \leq p_n^{ul} \leq p_n^{\max}\}$, $\mathcal{Q} \triangleq \{\boldsymbol{p}^{dl} : 0 \leq p_n^{dl}, \sum_{n \in \mathcal{N}} p_n^{dl} \leq p_0\}$, are defined as the feasible sets of $\boldsymbol{a}, \boldsymbol{c}, \boldsymbol{p}^{ul}$, and $\boldsymbol{p}^{dl}$, respectively.

For each iteration t, $\forall m \in \mathcal{M}^t$, where $\mathcal{M}$ is the set of indices, the upper-bound proximate function $\mathcal{D}_m$ of the objective function in (26) is defined. The quadratic penalty term was added to the objective function of the convex guarantee. The objective function can be approximated as

$$\mathcal{D}_m(\boldsymbol{a}_m; \boldsymbol{a}^{(t)}, \boldsymbol{c}^{(t)}, \boldsymbol{p}^{ul,(t)}, \boldsymbol{p}^{dl,(t)}) = \mathcal{D}(\boldsymbol{a}_m; \tilde{\boldsymbol{a}}, \tilde{\boldsymbol{c}}, \tilde{\boldsymbol{p}}^{ul}, \tilde{\boldsymbol{p}}^{dl}) + \frac{\mu_m}{2} \|(\boldsymbol{a}_m - \tilde{\boldsymbol{a}})\|^2, \tag{27}$$

where $\mu_m > 0$ is the penalty coefficient. The approximate function can be deployed to other vectors of variables, $\boldsymbol{c}_m, \boldsymbol{p}_m^{ul}$, and $\boldsymbol{p}_m^{dl}$, respectively. In addition, for each iteration t, the approximate function has unique minimizer vectors

Algorithm 2 The Proposed Algorithm for the Joint Problem of the Offloading Decision, Caching Strategy, Computational Resource, and Power Allocation

1: **Initialization:** Set the iteration index $t = 0$, stopping criteria $\epsilon > 0$, and set initial feasible solutions $(\boldsymbol{a}^{(0)}, \boldsymbol{c}^{(0)}, \boldsymbol{f}^{(0)}, \boldsymbol{p}^{ul,(0)}, \boldsymbol{p}^{dl,(0)})$.
2: **repeat**
3: $t = t + 1$.
4: According to (20), calculate $f^{(t)}$.
5: Select index set $\mathcal{M}^t$.
6: Let $\boldsymbol{a}_m^{(t)} = \underset{\boldsymbol{a}_m \in \mathcal{M}}{\operatorname{argmin}} \ \mathcal{D}_m(\boldsymbol{a}_m; \boldsymbol{a}^{(t-1)}, \boldsymbol{c}^{(t-1)}, \boldsymbol{p}^{ul,(t-1)}, \boldsymbol{p}^{dl,(t-1)})$.
7: Set $\boldsymbol{a}_j^{(t)} = \boldsymbol{a}_j^{(t)}, \forall j \notin \mathcal{M}^t$.
8: Find $\boldsymbol{c}_m^{(t)}$, $\boldsymbol{p}_m^{ul,(t)}$, and $\boldsymbol{p}_m^{dl,(t)}$ by solving the sub-problem in (29), (30), and (31).
9: **until** $\| \frac{\mathcal{D}_m^{(t)} - \mathcal{D}_m^{(t-1)}}{\mathcal{D}_m^{(t-1)}} \| \leq \epsilon$.
10: Then, set $\boldsymbol{a}^* = \boldsymbol{a}^{(t)}, \boldsymbol{c}^* = \boldsymbol{c}^{(t)}, \boldsymbol{f}^* = \boldsymbol{f}^{(t)}, \boldsymbol{p}^{ul,*} = \boldsymbol{p}^{ul,(t)}, \boldsymbol{p}^{dl,*} = \boldsymbol{p}^{dl,(t)}$ as the final solution.

$\tilde{\boldsymbol{a}}, \tilde{\boldsymbol{c}}, \tilde{\boldsymbol{p}}^{ul}$, and $\tilde{\boldsymbol{p}}^{dl}$ with respect to $\boldsymbol{a}, \boldsymbol{c}, \boldsymbol{p}^{ul}$, and $\boldsymbol{p}^{dl}$, which is regarded to be the solution of the previous step of $(t-1)$. The solution for each iteration $(t+1)$ can then be modified by solving the following sub-problems:

$$\boldsymbol{a}_m^{(t+1)} = \min_{\boldsymbol{a}_m \in \mathcal{A}} \mathcal{D}_m(\boldsymbol{a}_m; \boldsymbol{a}^{(t)}, \boldsymbol{c}^{(t)}, \boldsymbol{p}^{ul,(t)}, \boldsymbol{p}^{dl,(t)}), \tag{28}$$

$$\boldsymbol{c}_m^{(t+1)} = \min_{\boldsymbol{c}_m \in \mathcal{C}} \mathcal{D}_m(\boldsymbol{c}_m; \boldsymbol{c}^{(t)}, \boldsymbol{a}^{(t+1)}, \boldsymbol{p}^{ul,(t)}, \boldsymbol{p}^{dl,(t)}), \tag{29}$$

$$\boldsymbol{p}_m^{ul,(t+1)} = \min_{\boldsymbol{p}_m^{ul} \in \mathcal{P}} \mathcal{D}_m(\boldsymbol{p}_m^{ul}; \boldsymbol{p}^{ul,(t)}, \boldsymbol{a}^{(t+1)}, \boldsymbol{c}^{(c+1)}, \boldsymbol{p}^{dl,(t)}), \tag{30}$$

$$\boldsymbol{p}_m^{dl,(t+1)} = \min_{\boldsymbol{p}_m^{dl} \in \mathcal{Q}} \mathcal{D}_m(\boldsymbol{p}_m^{dl}; \boldsymbol{p}^{dl,(t)}, \boldsymbol{a}^{(t+1)}, \boldsymbol{c}^{(t+1)}, \boldsymbol{p}^{ul,(t+1)}). \tag{31}$$

The sub-problems in (28)-(31) can be solved by using our proposed BSUM method. Combined with (20), the proposed algorithm can be obtained. Based on the above analysis, Algorithm 2 provides the details of the proposed algorithm. In the problem $P1$, we have shown the convexity based on the Hessian matrix. We then obtain the closed-form solution via KKT optimality conditions; thus, the sub-problem $P1$ is always guaranteed to converge at the optimal solution. The second sub-problem $P2$ is obtained solution by BSUM algorithm [30], [32]. Based on [30], [32], we can claim that our proposed algorithm can converge to an ϵ-optimal solution. We now analyze the computation complexity of the proposed algorithm. Since $P1$ is a convex problem and existed a closed-form solution, the complexity of $P1$ is $\mathcal{O}(1)$. For the problem $P2$, BSUM has the computation complexity $\mathcal{O}(\log(1/\epsilon))$ [30], [32]. Thus, the total complexity of the proposed algorithm is $\mathcal{O}(\log(1/\epsilon))$.

V. NUMERICAL SIMULATION

This section provides the simulation results to evaluate the performance of the proposed algorithm and compare it with some schemes. The simulation settings are as follows unless specified otherwise. A system setting with a BS located in the center of a 200×200 m^2 area was considered. All users are then deployed at random within the BS coverage. The path-loss from user n to the BS can be calculated as $L(d_n) = 128.1 + 37.6\log_{10}(d_n)$, where d_n denotes is the distance between the n^{th} user and the BS. The spectrum bandwidth was $B = 1$ MHz. The noise density was set to $\sigma^2 = -174$ dBm/Hz. Here, the maximum uplink transmit power of each user for performing the offloaded task to the MEC server was $p_n^{\max} = 23$ dBm, $\forall n \in \mathcal{N}$ and the maximum downlink transmit power of the BS for delivering the data-contents to the users was set to $p_0 = 48$ dBm. For a computation-intensive task, the input data size of the computation-intensive tasks are distributed randomly with $U_n \in [1.2, 2]$ Mb and the corresponding number of CPU of computation-intensive task are distributed randomly with $W_n \in [700, 1200]$ megacycles. The local computing capacity of each user is $f_n^l = 0.7$ GHz, $\forall n \in \mathcal{N}$, and the maximum computing resource of the MEC server is $f^{\max} = 30$ GHz. The size of each requested data-content to a computation-intensive task execution is $V_n \in [0.5, 1]$ Mb, in which total type of requested data-contents from remote cloud is $N_v = 500$. The average transmission rate of backhaul between the MEC server and the remote cloud is $r^{\text{bh}} = 500$ Mbps. The parameter of the Zipf popularity distribution is $\kappa = 0.56$ [4], [8]. The maximum cache storage capability of the BS is 50% of the total data-content size. This ensures that some requested data-contents are not cached at the MEC server. All the simulation plots were achieved from random channel realizations on average.

Three benchmark schemes were introduced to illustrate the advantages of the proposed algorithm in reducing the total computing time:

- Local-only: In this scheme, all users execute their computation-intensive tasks locally (i.e., $a_n = 0, \forall n \in \mathcal{N}$). The computing resource, communication resources (i.e., power of uplink and downlink), and caching decision are obtained using the proposed algorithm.
- All-offloading: All users offload their computation-intensive tasks to the MEC server at the BS (i.e., $a_n = 1, \forall n \in \mathcal{N}$). Similar to the case of local computing only, other variables are optimized.
- Equal resource: The communication resources are equally allocated to all users. In this case, computing resource, offloading decision, and caching strategy are solved using the proposed algorithm.

Fig. 3 compares the total completion latency versus the number of users under various schemes. With increasing number of users, the total completion latency of all schemes also increases. The figure confirmed that the proposed algorithm generates the lowest completion latency compared to the other schemes. The total completion latency under the proposed algorithm was 17.68%, 26.02%, and 70.98% lower than that of all-offloading, local-only, and equal resources, respectively. Fig. 4 presents the total energy consumed of the

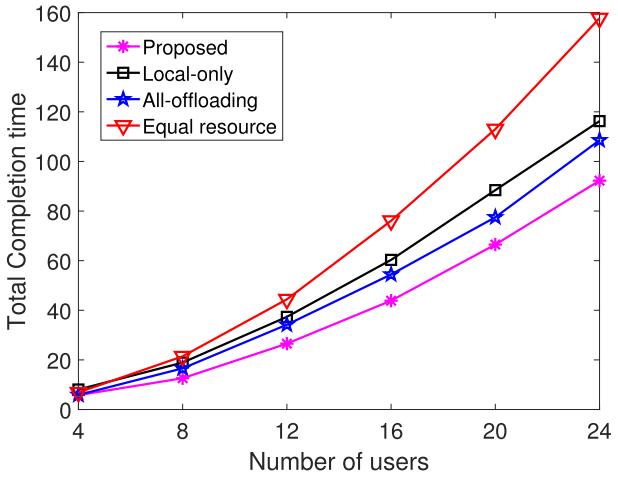

FIGURE 3. Comparison of the total completion latency as a function of the number of users.

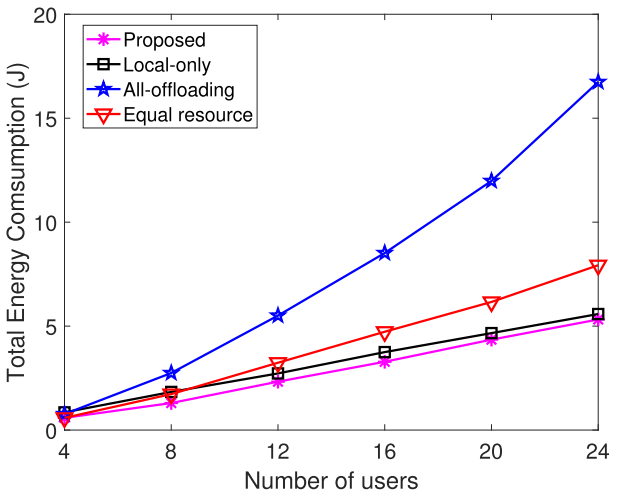

FIGURE 4. Comparison of the total energy consumed versus the number of users.

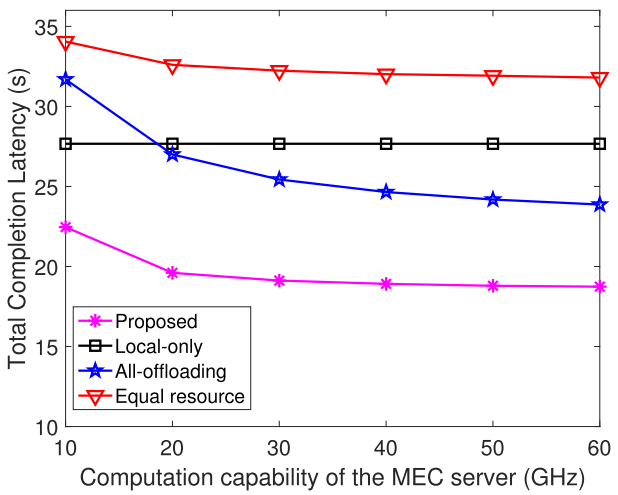

FIGURE 5. Comparison of the total completion latency different the computational capability of the MEC server.

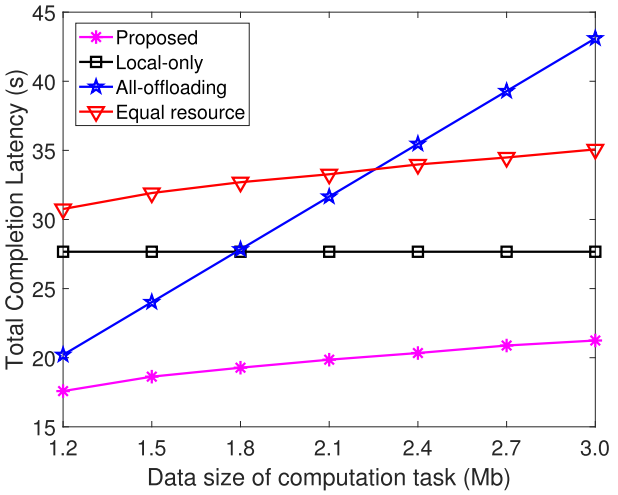

FIGURE 6. Comparison of the total completion latency different the offloaded data size of the computation-intensive task.

users against the number of users. The energy computation of all-offloading scheme is higher than in the other schemes. The offloading is beneficial in terms of the completion latency but requires a larger amount of energy consumed by the users. This figure shows that the proposed algorithm is beneficial in both the completion latency and energy consumption.

For more performance comparisons of total completion latency, the number of users was set to 10. The relationship between the total completion latency and the computational capability of the MEC server was then analyzed. Fig. 5 shows that the completion latency in the local-only case remains constant because all uses execute their computational tasks locally, irrespective of $f_n, \forall n \in \mathcal{N}$. The total completion latency of the three schemes, such as the proposed, all-offloading, and equal resource, decreases with increasing computational capability of the MEC server, ranging from 10 to 60 GHz. The all-offloading scheme performance curve decreased the most with increasing computing resources. This is reasonable because all tasks offloaded to the MEC server benefit, while the remaining two schemes may have some tasks offloaded to the server. On the other hand, the proposed algorithm still has better performance in terms of total completion latency. Always offloading over unfavorable wireless channels causes higher overhead (energy and latency). As considered in this problem, joint optimizing computational offloading and caching decisions with resource allocation can exploit the benefits of both local and remote computing to increase the system performance.

The offloading performance regarding the input size of the data of the computational tasks for offloading U_n was analyzed, as shown in Fig. 6. When the size of the input data U_n increases, the total completion latency of the local-only scheme is unaffected because there is no offloading. The completion latency of the all-offloading scheme increases significantly and becomes the worst compared to the other schemes when the input data size is large enough. Therefore, to achieve good performance, priority should be given to

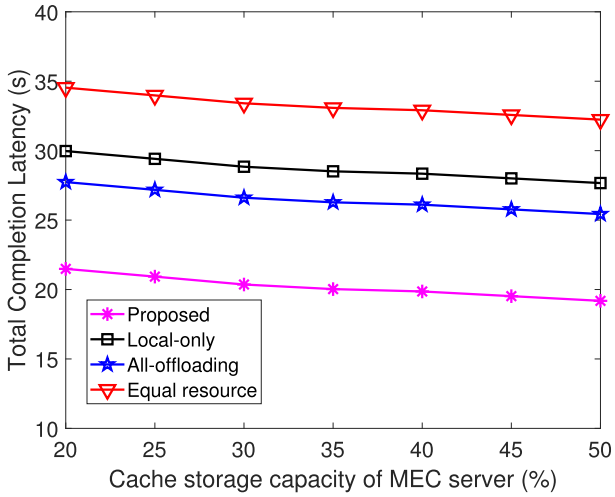

FIGURE 7. Effect of the caching storage capacity on the completion latency of tasks.

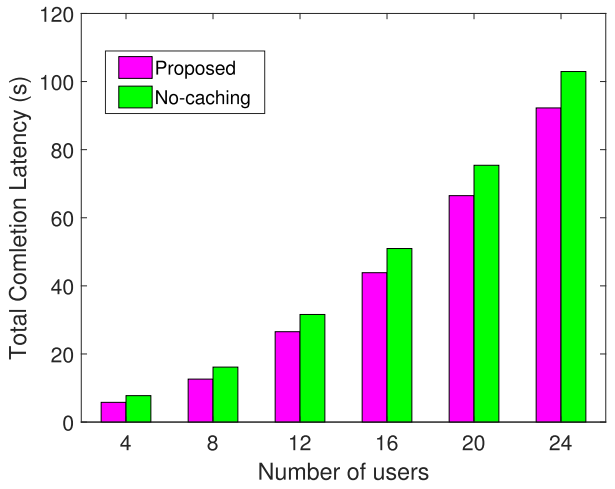

FIGURE 8. Comparison of our proposed algorithm versus No-caching.

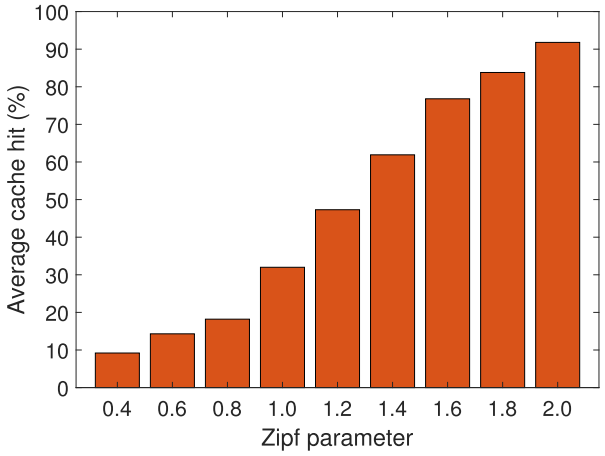

FIGURE 9. Cache hit under different Zipf parameters.

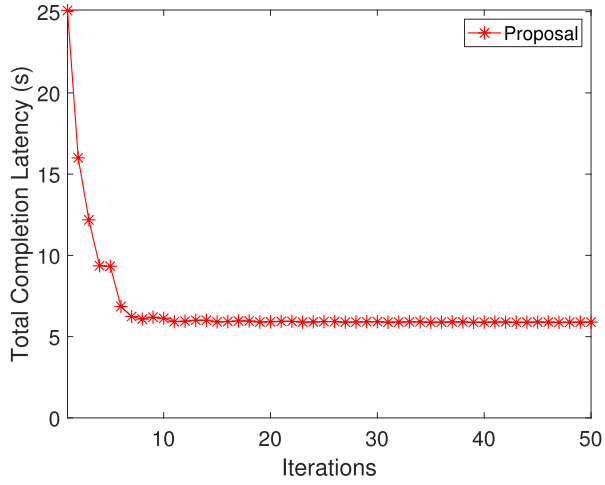

FIGURE 10. Convergence of the proposed algorithm.

offloading tasks with smaller data sizes rather than offloading tasks with larger data sizes. This is because by increasing the input data size, the completion latency for the offload computation-intensive tasks becomes higher. Therefore, the proposed scheme achieves better performance than the other schemes.

The next experiment, evaluated the effects of the MEC server's caching storage capacity on the completion latency. Fig. 7 show the total completion latency of users when adjusting the BS caching capacity (i.e., $C_{\text{cache}} \in [20\%; 50\%]$ of the total data-content size). The figure shows that the total completion latency of all schemes decreases with increasing cache storage capacity. The reason is that more data-content can be cached in the BS when there is a greater cache storage capacity. As a result, increasing the cache storage capacity can lower the latency required to download data-content from the remote cloud. Overall, the proposed algorithm has the lowest latency and performs better than the other schemes.

Fig. 8 compares data-content caching efficiency on the completion latency under various numbers of users. In the no-caching scheme, there is no requested data-content stored at the BS (i.e., $c_n = 0$). That is, all requested data-contents of computation-intensive tasks must be downloaded from the remote cloud. As a result, the backhaul latency is generated for each computational task when the requested data-content by the user is downloaded from the remote cloud. Furthermore, Fig. 8 shows the total completion latency when the number of users is increased. On the other hand, the larger the number of users, the larger the latency gap between the proposed algorithm and no-caching scheme. This can be explained by the large amount of data-content cached. There are many data-contents that the user requests are taken from the edge network. The delay is reduced by reducing the data-content that has to download data from the remote cloud. These analyses show that the completion latency of the users can be decreased by implementing the caching storage and storage of popularly requested BS data-content.

Fig. 9 presents the average cache hits on the MEC server. The algorithm was run 1000 times to calculate the cache hit. If the requested data-content is not cached on the MEC server (cache missed), it will be retrieved from

the remote cloud. Fig. 9 shows that the average cache hits increase with increasing Zipf parameter (e.g., Zipf parameter = 2.0, cache hit ratio = 91.80%). In other words, as the Zipf parameter increases, more content becomes more popular, resulting in a corresponding increase in cache hits and a decrease in backhaul bandwidth.

Finally, the convergence rate of the proposed algorithm was evaluated, as shown in Fig. 10. The figure shows that the proposed algorithm only requires fewer iterations to converge to the optimal solution, indicating that the proposed approach is efficient.

VI. CONCLUSION

This study examined the optimization of computational offloading, data-content caching, and resource management in NOMA-MEC systems considering both the uplink and downlink transmissions. An efficient algorithm was developed to minimize the total latency of users. The simulation results confirmed the convergence of the proposed scheme within only a few iterations and demonstrated the superiority of the proposed scheme in terms of delay reduction and energy saving. Future studies will consider joint computational offloading, caching strategy, and communication and computing resource allocation in NOMA-MEC heterogeneous networks.

REFERENCES

[1] Q.-V. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W.-J. Hwang, and Z. Ding, "A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116974–117017, 2020.

[2] A. Ndikumana, N. H. Tran, T. M. Ho, Z. Han, W. Saad, D. Niyato, and C. S. Hong, "Joint communication, computation, caching, and control in big data multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 6, pp. 1359–1374, Jun. 2020.

[3] S. Yu, R. Langar, X. Fu, L. Wang, and Z. Han, "Computation offloading with data caching enhancement for mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11098–11112, Nov. 2018.

[4] J. Zhang, X. Hu, Z. Ning, E. C.-H. Ngai, L. Zhou, J. Wei, J. Cheng, B. Hu, and V. C. M. Leung, "Joint resource allocation for latency-sensitive services over mobile edge computing networks with caching," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4283–4294, Jun. 2019.

[5] Y. Wei, F. R. Yu, M. Song, and Z. Han, "Joint optimization of caching, computing, and radio resources for fog-enabled IoT using natural actor–critic deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2061–2073, Apr. 2019.

[6] Y. Hao, M. Chen, L. Hu, M. S. Hossain, and A. Ghoneim, "Energy efficient task caching and offloading for mobile edge computing," *IEEE Access*, vol. 6, pp. 11365–11373, 2018.

[7] X. Yang, Z. Fei, J. Zheng, N. Zhang, and A. Anpalagan, "Joint multi-user computation offloading and data caching for hybrid mobile cloud/edge computing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11018–11030, Nov. 2019.

[8] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.

[9] M. I. A. Zahed, I. Ahmad, D. Habibi, and Q. V. Phung, "Green and secure computation offloading for cache-enabled IoT networks," *IEEE Access*, vol. 8, pp. 63840–63855, 2020.

[10] S. Bi, L. Huang, and Y.-J.-A. Zhang, "Joint optimization of service caching placement and computation offloading in mobile edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 7, pp. 4947–4963, Jul. 2020.

[11] K. Wang, Z. Hu, Q. Ai, Y. Zhong, J. Yu, P. Zhou, L. Chen, and H. Shin, "Joint offloading and charge cost minimization in mobile edge computing," *IEEE Open J. Commun. Soc.*, vol. 1, pp. 205–216, 2020.

[12] H. Li, F. Fang, and Z. Ding, "Joint resource allocation for hybrid NOMA-assisted MEC in 6G networks," *Digit. Commun. Netw.*, vol. 6, no. 3, pp. 241–252, Aug. 2020.

[13] R. Ruby, S. Zhong, D. W. K. Ng, K. Wu, and V. C. M. Leung, "Enhanced energy-efficient downlink resource allocation in green non-orthogonal multiple access systems," *Comput. Commun.*, vol. 139, pp. 78–90, May 2019.

[14] X. Liang, Y. Wu, D. W. K. Ng, S. Jin, Y. Yao, and T. Hong, "Outage probability of cooperative NOMA networks under imperfect CSI with user selection," *IEEE Access*, vol. 8, pp. 117921–117931, 2020.

[15] Z. Wei, L. Yang, D. W. K. Ng, J. Yuan, and L. Hanzo, "On the performance gain of NOMA over OMA in uplink communication systems," *IEEE Trans. Commun.*, vol. 68, no. 1, pp. 536–568, Jan. 2020.

[16] Y. Sun, D. W. K. Ng, Z. Ding, and R. Schober, "Optimal joint power and subcarrier allocation for full-duplex multicarrier non-orthogonal multiple access systems," *IEEE Trans. Commun.*, vol. 65, no. 3, pp. 1077–1091, Mar. 2017.

[17] Z. Ding, D. W. K. Ng, R. Schober, and H. V. Poor, "Delay minimization for NOMA-MEC offloading," *IEEE Signal Process. Lett.*, vol. 25, no. 12, pp. 1875–1879, Dec. 2018.

[18] Y. Wu, L. P. Qian, K. Ni, C. Zhang, and X. Shen, "Delay-minimization nonorthogonal multiple access enabled multi-user mobile edge computation offloading," *IEEE J. Sel. Topics Signal Process.*, vol. 13, no. 3, pp. 392–407, Jun. 2019.

[19] F. Wang, J. Xu, and Z. Ding, "Multi-antenna NOMA for computation offloading in multiuser mobile edge computing systems," *IEEE Trans. Commun.*, vol. 67, no. 3, pp. 2450–2463, Mar. 2019.

[20] Z. Ding, J. Xu, O. A. Dobre, and H. V. Poor, "Joint power and time allocation for NOMA–MEC offloading," *IEEE Trans. Veh. Technol.*, vol. 68, no. 6, pp. 6207–6211, Jun. 2019.

[21] Y. Pan, M. Chen, Z. Yang, N. Huang, and M. Shikh-Bahaei, "Energy-efficient NOMA-based mobile edge computing offloading," *IEEE Commun. Lett.*, vol. 23, no. 2, pp. 310–313, Feb. 2019.

[22] Z. Yang, C. Pan, J. Hou, and M. Shikh-Bahaei, "Efficient resource allocation for mobile-edge computing networks with NOMA: Completion time and energy minimization," *IEEE Trans. Commun.*, vol. 67, no. 11, pp. 7771–7784, Nov. 2019.

[23] Q.-V. Pham, H. T. Nguyen, Z. Han, and W.-J. Hwang, "Coalitional games for computation offloading in NOMA-enabled multi-access edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 1982–1993, Feb. 2020.

[24] X. Diao, J. Zheng, Y. Wu, and Y. Cai, "Joint computing resource, power, and channel allocations for D2D-assisted and NOMA-based mobile edge computing," *IEEE Access*, vol. 7, pp. 9243–9257, 2019.

[25] M. Nduwayezu, Q.-V. Pham, and W.-J. Hwang, "Online computation offloading in NOMA-based multi-access edge computing: A deep reinforcement learning approach," *IEEE Access*, vol. 8, pp. 99098–99109, 2020.

[26] F. Fang, Y. Xu, Z. Ding, C. Shen, M. Peng, and G. K. Karagiannidis, "Optimal resource allocation for delay minimization in NOMA-MEC networks," *IEEE Trans. Commun.*, vol. 68, no. 12, pp. 7867–7881, Dec. 2020.

[27] H.-G.-T. Pham, Q.-V. Pham, A. T. Pham, and C. T. Nguyen, "Joint task offloading and resource management in NOMA-based MEC systems: A swarm intelligence approach," *IEEE Access*, vol. 8, pp. 190463–190474, 2020.

[28] Y. K. Tun, A. Ndikumana, S. R. Pandey, Z. Han, and C. S. Hong, "Joint radio resource allocation and content caching in heterogeneous virtualized wireless networks," *IEEE Access*, vol. 8, pp. 36764–36775, 2020.

[29] Z. Ding, Z. Yang, P. Fan, and H. V. Poor, "On the performance of non-orthogonal multiple access in 5G systems with randomly deployed users," *IEEE Signal Process. Lett.*, vol. 21, no. 12, pp. 1501–1505, Dec. 2014.

[30] M. Hong, M. Razaviyayn, Z.-Q. Luo, and J.-S. Pang, "A unified algorithmic framework for block-structured optimization involving big data: With applications in machine learning and signal processing," *IEEE Signal Process. Mag.*, vol. 33, no. 1, pp. 57–77, Jan. 2016.

[31] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[32] M. Razaviyayn, M. Hong, and Z.-Q. Luo, "A unified convergence analysis of block successive minimization methods for nonsmooth optimization," *SIAM J. Optim.*, vol. 23, no. 2, pp. 1126–1153, Jan. 2013.

LUAN N. T. HUYNH received the B.Sc. degree in information technology from the Ho Chi Minh City University of Transport, Vietnam, in 2009, and the M.Sc. degree in data communication and networking from the Posts and Telecommunications Institute of Technology, Vietnam, in 2012. He is currently pursuing the Ph.D. degree in computer science and engineering with Kyung Hee University, South Korea. He is also a Lecturer with the School of Engineering and Technology, Thu Dau Mot University, Vietnam. His research interests include applying analytic techniques of optimization and machine learning to optimize edge/cloud computing and the Internet of Things.

QUOC-VIET PHAM (Member, IEEE) received the B.S. degree in electronics and telecommunications engineering from the Hanoi University of Science and Technology, Vietnam, in 2013, and the Ph.D. degree in telecommunications engineering from Inje University, South Korea, in 2017.

From September 2017 to December 2019, he was with Kyung Hee University, Changwon National University, and Inje University, on various academic positions. He is currently a Research Professor with the Research Institute of Computer, Information and Communication, Pusan National University, South Korea. He has been granted the Korea NRF Funding for outstanding young researchers for the term 2019–2023. His research interests include convex optimization, game theory, and machine learning to analyze and optimize edge/cloud computing systems and beyond 5G networks. He received the Best Ph.D. Dissertation Award in Engineering (rank#1) from Inje University, in 2017. He is an Associate Editor of *Journal of Network and Computer Applications* (Elsevier).

TRI D. T. NGUYEN received the B.S. degree in computer science and engineering from the Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam, in 2012. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Kyung Hee University. His research interests include mobile edge computing, vehicular edge computing, cloud computing, the Internet of Things, and machine learning.

MD. DELOWAR HOSSAIN received the B.Sc. and M.Sc. degrees from the Department of Information and Communication Engineering (ICE), Islamic University, Bangladesh, in 2004 and 2005, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Kyung Hee University, South Korea. He was a Visiting Scholar with Infosys, Bengaluru, India. He is also serving as an Associate Professor with the Department of Computer Science and Engineering, Hajee Mohammed Danesh Science and Technology University, Dinajpur, Bangladesh, where he was the Chairman from 2011 to 2013. His current research interests include cloud/edge/fog computing, big data, machine learning, and the Internet of Things. He was a recipient of Best Paper Award in KSC 2018 and KSC 2019, South Korea.

YOUNG-ROK SHIN received the B.S. and master's degrees in computer engineering and the Ph.D. degree in computer science and engineering from Kyung Hee University, South Korea, in 2009, 2011, and 2017, respectively. He was a Research Professor with the Department of Computer Science and Engineering, Kyung Hee University, from November 2018 to October 2020. He is currently working as a Research Director with Happycom Company Ltd. His research interests include cloud/edge computing and service-level agreements.

EUI-NAM HUH (Member, IEEE) received the B.S. degree from Busan National University, South Korea, the master's degree in computer science from The University of Texas, USA, in 1995, and the Ph.D. degree from Ohio University, USA, in 2002. He is currently a Professor with the Department of Computer Science and Engineering, Kyung Hee University, South Korea. His research interests include cloud computing, the Internet of Things, future Internet, distributed real-time systems, mobile computing, big data, and security. He is a member of the Review Board of the National Research Foundation of Korea. He has also served many community services for ICCSA, WPDRTS/IPDPS, APAN Sensor Network Group, ICUIMC, ICONI, APIC-IST, ICUFN, and SoICT as various types of chairs. He is also the Vice-Chairman of the Cloud/Bigdata Special Technical Group of TTA and an Editor of ITU-T SG13 Q17.