Cost-Minimized Computation Offloading of Online Multi-Function Services in Collaborative Edge-Cloud Networks

Chuan Feng, Pengchao Han, Xu Zhang, Qihan Zhang, Yue Zong, Yejun Liu, and Lei Guo

***Abstract*—Cloud Computing (CC) is powerful for the computation offloading of services, promoting the implementation of various modern applications. Mobile Edge Computing (MEC) can provide low-latency services utilizing edge servers locating in proximity to users. The combination of MEC and CC can give play to the dual advantages of both. However, it is a challenging problem to offload service requests to the collaborative edge-cloud networks aiming at minimizing costs due to the resource limitation of edge servers and the online feature of services. To address this issue, we mathematically model the service requests with multiple inter-connected functions. Then, the problem of computation offloading of multi-function service requests in collaborative edge-cloud networks is formulated to be an Integer Linear Programming (ILP) and is proved to be NP-hard. Furthermore, a Cost-minimized Computation Offloading with Reconfiguration (CCOR) algorithm is proposed to minimize the total cost of online services. Finally, simulation results show that the proposed CCOR algorithm can effectively reduce the cost of computation offloading with higher resource utilization of edge cloud compared with baseline algorithms.**

***Index Terms*—Computation offloading, cost optimization, mobile edge computing, cloud computing.**

I. Introduction

Cloud Computing (CC) [1] has gained a lot of popularity for providing services to modern applications, e.g., AR/VR, online deep learning platform, and cloud games by leveraging resources such as CPU, RAM, and storage [2] in a shared resource pool. As a result, complicated and expensive devices are not necessary for users to get access to complex applications. For example, thank to cloud games [3], players no longer need to have high-end processors and graphics cards on mobile phones to play 3H (high quality, high sales, and high cost) games.

However, applications [4] in the 5G era have posed higher requirements in terms of mobility, security, energy [5], latency, and reliability [6]. The service provisioning supported by the central cloud is facing the challenge of the Quality of Service (QoS) guarantee for these applications, especially the delay. For instance, in a multiplayer online game, the network delay affects the time interval between two actions and thus degrade the user experience. As a result, turn-based strategy games have very low delay requirements, i.e., 50 ms. For games with static scenarios, such as strategy and role-playing games, players have larger tolerable delay but still no more than 100 ms. Therefore, Mobile Edge Computing (MEC) has emerged, where geographically distributed edge clouds are located in proximity to end-users [7]. Namely, each wireless base station (BS) is able to access an edge cloud containing multiple inter-connected servers.

Combining the MEC and CC is beneficial for service provisioning with not only low latency but also high computation capacity. Specifically, by making full use of the computing power of edge devices, MEC [8] is preferable for processing real-time applications with private data, relieving the communication overhead for the computation offloading to the central cloud [9]. On the other hand, the central cloud with infinite capacity is necessary as a compensation for the resource-limited edge cloud. Therefore, it is promising to offload service requests from users to the collaborative edge-cloud networks for high QoS service provisioning with full coverage. In order to improve the QoS of services, i.e., to reduce delay, any service request is inclined to be offloaded to the edge cloud. The central cloud is responsible for holding requests when the resources of edge cloud are not enough.

A. Motivation

Related works for computation offloading in collaborative edge-cloud networks either assign one service request to one edge/central cloud server [10] or offload a part of a request from the local device to a server, i.e., partial offloading [11]. However, offloading each service request as a whole is always unavailable due to the resource limitation of edge servers. Partial offloading is also impractical on account that any application cannot be split arbitrarily [12], [13]. In addition, offloading online service requests according to their arrival time always results in the unbalanced distribution of workloads on the whole cloud networks. Reconfiguration of already offloaded services is critical to achieve global optimization. Overall, in this paper, we highlight the computation offloading of online service requests with multiple functions in collaborative edge-cloud networks.

B. Contributions

We mainly make the following contributions.

Chuan Feng, Qihan Zhang, and Yue Zong are with the School of Computer Science and Engineering, Northeastern University, Shenyang 110169, China.

Pengchao Han (Corresponding author) is with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, 518172, China (e-mail: hanpengchao@cuhk.edu.cn).

Xu Zhang, Yejun Liu, and Lei Guo are with the School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China.

1) In collaborative edge-cloud networks, we model the service requests by characterizing multiple functions in each service request, based on which, servers can work collaboratively to support a service. Furthermore, we formulate the problem of the computation offloading of multi-function service requests aiming at minimizing the total cost as an ILP model and prove it to be NP-hard.
2) To decrease the total cost of the service provisioning, a Cost-minimized Computation Offloading with Reconfiguration (CCOR) algorithm is proposed to offload more service requests that arrive online to edge cloud networks. Note that, the three processes (function offloading, link allocation, and service reconfiguration) are designed in the CCOR algorithm.
3) We conduct numerous simulations to verify the effectiveness of the proposed algorithms under different scales of physical networks. For small-scale networks, the cost of the proposed CCOR algorithm only 3.51% higher than the ILP solution. In addition, we also set up different service request scenarios to verify the scalability of the overall framework in large-scale networks. The results show that the lower cost is achieved by CCOR compared with the Baseline algorithms.

The remainder of this paper is organized as follows. The related works are reviewed in Section II. Section III presents the system models, based on which the problem of the computation offloading of multi-function service requests in collaborative edge-cloud networks is formulated and analyzed. In Section IV, we propose a CCOR algorithm for offloading online service requests. Section V presents the simulation results and we conclude this paper in Section VI.

II. Related Work

A. Service Architecture

There are many existing works focusing on the field of MEC networks considering user characteristics. The characteristics of users can be summarized from different perspectives, including required service architecture, i.e., full offloading or partial offloading and different number of subtask/functions, environment uncertainty sensing, online service requests, user mobility, etc.

Classic computation offloading intends to offload user requests to the central cloud, where each request is evaluated by computation resource and cannot be split during offloading. They have emphasized on minimizing the resource cost and guaranteeing the QoS of users [14], [15], [16]. Leveraging the advantage of the low latency of using edge servers for computation offloading, works in the literature have emphasized on designing the implementation protocols of real-world applications [17], optimizing the deployment of edge servers [18], and mobility management of services [19].

However, taking the characteristic of limited resources of edge servers into account, placing a large-scale service as a whole is always not available in practice. The partial offloading [20], [21], [22], [23], [24], [25], [26], has been investigated to offload a part of the computation instructions of a service to an edge server to meet the computation capacity constraint of the server. For example, aiming to maximize users' perceived satisfaction, part of each service request with specific functions can be offloaded to edge servers over a 5G heterogeneous network in [20]. For each service request, not only which server should be chosen for the offloading but also how much computation instructions should be offloaded are required to be determined. However, it is impractical to split any services randomly.

Some works focus on Virtualized Network Functions (VNFs) in [27], [28], [29], [30]. For a specific network application/service, the operators will dynamically select a chain of VNFs. More generally, a service can be divided into multiple dependent functions. For example, a face recognition application is mainly composed of five functions including image acquisition, face detection, pre-processing, feature extraction, and classification [22]. There are three typical architectures of services with multiple functions, namely sequential, parallel, and general structures [31], [32]. In this paper, we consider the general service architecture where each service is modeled by a graph with each function acting as a vertex. Highlighting the services with multiple functions is beneficial for the real-world computation offloading in edge cloud networks.

There are many works in the field of computation offloading in MEC networks considering other user characteristics apart from required service architecture, including environment uncertainty sensing, online service requests, and mobility [20], [26], [33]. Considering the uncertainty introduced by the shared computing environment, the users are driven to exhibit a risk-aware behavior [20]. Aiming to maximize their payoff from the resources allocation, the authors have proposed the autonomous action model under the uncertainty. By charging users a price, the authors have regulated user behavior in offloading requests to the Access Point (AP) in [26]. The user decides the number of tasks to offload by comparing its utility function with the delay cost required by edge computing. Considering the mobile nature of users, the authors have built a mobility model based on the obtained trajectory of a pedestrian in [33] for studying the problem of task execution for latency minimization.

B. Computation Offloading in Edge Cloud Networks

There are many works stressing the computation offloading in edge cloud networks constraining that each user request should be fully offloaded to one edge server [34], [33], [35], [36], [37]. A method on the offloading of user requests to the only one edge server has been designed to minimize the total resource cost as well as the latency of users in [34]. Besides, more works focus on the computation offloading in edge cloud networks with multiple servers [33], [35], [36], [37], which is more challenging as the offloading decision includes not only whether or not a request should be offloaded but also where to offload [21], [22]. The mobility of users that feature multiple service requests has been taken into account during the computation offloading in [33]. A heuristic offloading algorithm has been developed for improving the utility of the system in [35]. The problem of computation offloading aiming at maximizing the offloading gains or minimizing costs has also been solved in [36] and [37] respectively.

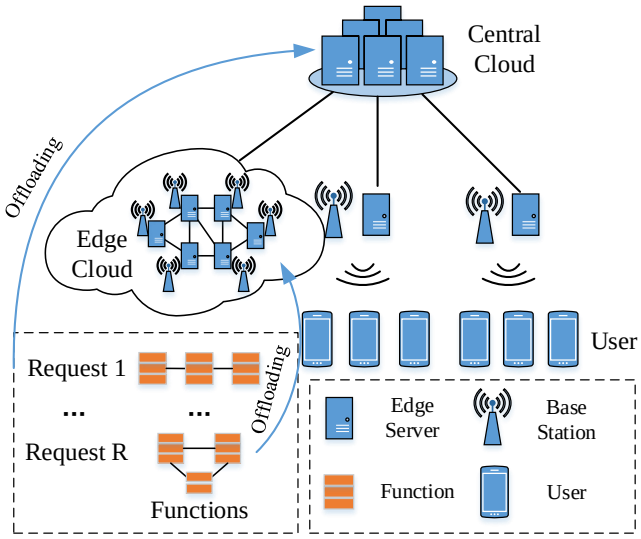


Fig. 1: A collaborative edge-cloud network architecture

The partial offloading of services in edge cloud networks has also been widely investigated [20], [21], [22], [23], [24], [25], [26], [38]. A price-based distributed method has been proposed to manage the offloaded services in [23] where each service request can be arbitrarily divided bitwise for partially local computing and partial offloading. In [24], the authors considered the heterogeneous energy harvesting MEC systems with multiple mobile devices and multiple edge servers. Any service request could be partially or fully offloaded to an edge server via wireless channels. The problem of multi-hop multi-task partial computation offloading in edge computing systems has been studied in [25] by considering the partial offloading among edge devices. Moreover, an incentive-based offloading control framework for the multi-access edge computing networks has been proposed in [26] for the partial offloading of service requests to access points. However, the practical characteristic of services, i.e., services arrive online, has been neglected in these works, which poses challenges for the global optimization of the offloading of all services.

Considering multiple functions offloading, some studies have focused on the computation offloading of VNF [29], [30] in edge cloud networks. The mentioned studies [27], [28] have considered VNFs placement instead of computation offloading of VNF. By sharing existing VNF instances or creating new VNF instances in cloudlet, the authors in [29] have considered the computation offloading approach of VNF in a mobile edge cloud network, aiming to maximize the number of admitted service requests within a limited time horizon while minimizing the operating cost of admitted service requests. In [30], the authors have applied the computation offloading method of VNF to the hardware processing node of the elephant stream. In terms of the computation offloading of services with multiple functions in edge cloud networks, the joint scheduling and offloading of mobile applications has been investigated in [32]. Aiming at minimizing the energy consumption while satisfying a strict delay constraint, the authors considered a practical offloading of applications consisting of a set of functions in [39]. However, the potential of utilizing the central cloud to make up for the resource-limited edge cloud networks for the computation offloading of multi-function services has also been ignored.

TABLE I: SUMMARY OF MAIN NOTATIONS

Notation	Description
R	Set of service requests.
$G\,(N, L)$	The physical network.
$G_r\,(V_r, E_r)$	The rth service request.
λ	The cost of unit computation resource in edge cloud.
β	The cost of unit bandwidth resource in edge cloud.
δ	The cost of offloading unit computation resource to the central cloud.
c_i^r	The computation resource required by the ith function in the rth service request.
C_j	The computation resource capacity of the jth edge node.
C'_j	The remaining computation resource capacity of the jth edge node.
b_e^r	The bandwidth requested by the eth link of the rth service request.
$B_{(j,j')}$	The total bandwidth capacity of the physical link (j, j').
e_s	The source node of the eth link in the rth service request.
e_d	The destination node of the eth link in the rth service request.
p	The path with the maximum remaining bandwidth from j to j'.
p_e^r	The path p of the eth link in the rth service request.
B_p	The bandwidth of path p, i.e., the minimum bandwidth of links on p.
$TR_{r,j}$	The transmission rate of the wireless channel between user r and BS j.
$BW_{r,j}$	The bandwidth of the wireless channel between user r and BS j.
p_j	The transmit power of the user r.
$h_{r,j}$	The channel gain of the wireless channel between user r and BS j.
$\sigma^2_{r,j}$	The noise power of the wireless channel between user r and BS j.
z_r	A binary decision variable, taking 1 if the service request r is uploaded to the central cloud and 0 if it is offloaded to the edge cloud.
$y_{i,j}^r$	A binary decision variable. It is equal to 1 if the ith function in the rth service request is allocated to the jth node in the edge cloud, otherwise it is 0.
$x^r_{e,(j,j')}$	A binary decision variable. It is equal to 1 if the eth link in the rth service request is allocated to the physical link from j to j' in the edge cloud, otherwise it is 0.
α	The coefficient of bandwidth resource with respect to computation resource for offloading services to the central cloud.
η	The average utilization rate of node resource.
ξ	The average utilization rate of link resource.
χ	The average path length in hops.

In summary, we comprehensively consider the real-world online service requests that are structured by multiple interconnected functions and intend to design efficient cost-minimized computation offloading strategies in collaborative edge-cloud networks.

III. System Models and Problem Definitions

A. Collaborative Edge-Cloud Networks

The computation offloading framework of service requests in collaborative edge-cloud networks is shown in Fig. 1.

The physical network consists of a central cloud, multiple geographically distributed edge clouds, and various terminal devices, i.e., users. The computation capacity of the central cloud is assumed to be infinite. The edge cloud network is represented by $G(N, L)$ where N represents the set of edge nodes including BSs and edge servers and L represents the set of links among edge nodes. The computation capacity of each node $j \in N$ is C_j. Each physical link between $j \in N$ and $j' \in N$, i.e., (j, j'), has the bandwidth capacity of $B_{(j,j')}$. In the MEC networks, we suppose that users get access to the target edge servers deployed at the associated BSs through 5G wireless links [40]. Each base station is allocated with a specific wireless channel and the channel interference among BSs is ignored. All the users with the coverage of a BS share its wireless channel in the manner of Frequency Division Multiple Access (FDMA) and its extended technique, e.g, Orthogonal Frequency Division Multiple Access (OFDMA) and Single-carrier Frequency-Division Multiple Access (SC-FDMA). The transmission rate $TR_{r,j}$ of the link between the user r and the BS j can be calculated based on Eq. (1), where $BW_{r,j}$ denotes the bandwidth between the user r and the BS j, p_r is transmitted power of the user r, $\sigma^2_{u,q}$ denotes power of white noise and $h_{r,j}$ denotes the channel gain.

$$TR_{r,j} = BW_{r,j} \cdot \log_2(1 + \frac{p_r \cdot h_{r,j}}{\sigma^2_{r,j}}) \tag{1}$$

B. Service Requests

1) Request model: Denote the set of service requests by R. Each online arrived service request $r \in R$ is represented by $G_r(V_r, E_r)$ where V_r represents the set of functions and E_r indicates the set of edges among the functions in V_r. The resource demand of the service request $r \in R$ is represented by (c^r_i, b^r_e) where c^r_i denotes the computation resources required by the function i in V_r and b^r_e means the bandwidth requested by the eth link in E_r. Any function in V_r can be placed either to a edge server or the central cloud, corresponding to a virtual machine. We assume that each function is not splittable and thus placed to only one edge server (or the central cloud).

2) A real-world example of service request: In this section, we take cloud games [3] as an example to illustrate the architecture of service requests. Functions in a cloud game consist of image rendering, video en/decoding, account management, etc. Data transmitting among functions corresponds to the instructions of players, images, and voices and so on. For the cloud game with moving players, different resolutions for rendering images in the game are provided to users with different moving speeds. Specifically, as a user keeps moving, the real-time speed of the user can be obtained by the accessed BS. Generally, when the speed is less than or equal to 2m/s, the user either is passing through this BS slowly or will stay at the BS for a while. The longer stay time of the user at the BS, the higher quality of service, i.e., better resolution of the rendered picture, should be ensured. Therefore, 1080p resolution with 6Mbps bandwidth games should be provided in this case. Otherwise, when the speed is greater than 2m/s indicating that the user passes through the BS quickly, then the basic requirements on the quality of service should be satisfied. Thus the resolution of 480p resolution with 2Mbps bandwidth can be allocated to the user. Note that our model is also applicable to service requests other than cloud games.

C. Problem Formulation

In this section, the problem of computation offloading of multi-function service requests in collaborative edge-cloud networks is formulated as ILP with the main notations defined in Table I.

Objective:

$$\begin{aligned} Minimize: & \sum_{r=1}^{|R|} \sum_{j=1}^{|N|} \sum_{i=1}^{|V_r|} \left[(1 - z_r) \cdot c^r_i \cdot y^r_{i,j} \cdot \lambda\right] + \\ & \sum_{r=1}^{|R|} \sum_{e \in E_r} \left[(1 - z_r) \cdot x^r_{e,(j,j')} \cdot \beta\right] + \sum_{r=1}^{|R|} \left[z_r \cdot \sum_{i=1}^{|V_r|} c^r_i \cdot \delta\right]. \end{aligned} \tag{2}$$

Subject to:

$$\sum_{j=1}^{|N|} y^r_{i,\,j} \leq 1,\ \forall r \in R,\ \forall i \in V_r, \tag{3}$$

$$\sum_{r=1}^{|R|} \sum_{i=1}^{|V_r|} c^r_i \cdot y^r_{i,j} \leq C_j, \forall j \in N, \tag{4}$$

$$x^r_{e,\,(j,\,j')} + x^r_{e,\,(j',\,j)} \leq 1,\ \forall r \in R,\ \forall (j,\ j') \in L, \forall e \in E_r, \tag{5}$$

$$\begin{aligned} \sum_{j' \in N} [x^r_{e,(j,j')} - x^r_{e,(j',j)}] = y^r_{e_s,j} - y^r_{e_d,j}, \forall r \in R, \\ \forall e \in E_r,\ \forall j \in N, \end{aligned} \tag{6}$$

$$z_r = \begin{cases} 0, & if \sum_{j=1}^{|N|} \sum_{i=1}^{|V_r|} y^r_{i,j} \neq 0 \\ 1, & if \sum_{j=1}^{|N|} \sum_{i=1}^{|V_r|} y^r_{i,j} = 0 \end{cases}, \forall r \in R, \tag{7}$$

$$\sum_{r=1}^{|R|} \sum_{e \in E_r} b^r_e \cdot x^r_{e,(j,j')} \leq B_{(j,j')}, \forall (j,\ j') \in L. \tag{8}$$

The objective in Eq. (2) is to minimize the cost of all the service requests consisting of the computation cost for offloading functions to edge servers, the bandwidth cost for link allocation, and the cost of offloading functions to the central cloud. Equations (3) and (4) are the constraints for offloading functions to edge servers. Equation (3) indicates that each function in a service request should be allocated to one edge node in the edge cloud. The computation resource capacity constraint of each edge server is stated in Eq. (4), indicating that the occupied resource by all the functions that are offloaded to an edge server should not exceed the computation capacity of the edge server. The allocation of links among functions in a service request is constrained by Eqs. (5) and (6) where Eq. (5) indicates that each link in a service request should be allocated to a loop-free physical path between the edge servers that its end functions are offloaded and the flow conservation is formulated in Eq. (6). Any service request that can not be fully offloaded to the edge cloud is then offloaded to the central cloud with $z_r = 1$ as illustrated in Eq. (7). The bandwidth occupied by service requests on each physical link should not exceed the total bandwidth capacity of the link as shown in Eq. (8).

Algorithm 1: CCOR algorithm

Input: R, $G(N, L)$
Output: $y^r_{i,j}, x^r_{e,(j,j')}, z_r, \forall r \in R, i \in V_r, e \in E_r, j \in N, (j, j') \in L$

```
for r ∈ R do
    if Algorithm 2 with inputs V_r and N succeeds then
        if Algorithm 3 with inputs y^r_{i,j}, ∀i ∈ V_r, j ∈ N, E_r and L succeeds then
            Update the remaining resources of physical nodes and links according to y^r_{i,j}, x^r_{e,(j,j')}, ∀i ∈ V_r, e ∈ E_r, j ∈ N, (j, j') ∈ L;
        end
        else
            Execute Algorithm 4 with inputs r, R, and G(N, L);
        end
    end
    else
        Execute Algorithm 4 with inputs r, R, and G(N, L);
    end
end
Calculate the cost based on Eq. (1);
```

Algorithm 2: Function Offloading

Input: V_r, N
Output: $y^r_{i,j}, \forall i \in V_r, j \in N$

```
Rank functions in V_r in the decreasing order of their computation requirements as V'_r;
Rank nodes in N in the decreasing order of their remaining computation capacity as N';
k ← 1;
for i ∈ V'_r do
    Featch the kth node j in N';
    if c^r_i ≤ C'_j then
        y^r_{i,j} ← 1;
        k ← k + 1;
    end
    else
        y^r_{i,j} ← 0, ∀i ∈ V_r, j ∈ N;
        Return False;
    end
end
```

Algorithm 3: Link Allocation

Input: $y^r_{i,j}, \forall i \in V_r, j \in N$, E_r, L
Output: $x^r_{e,(j,j')}, \forall e \in E_r, (j, j') \in L$

```
for e ∈ E_r do
    Find out the physical nodes j and j' with y^r_{e_s,j} = 1 and y^r_{e_d,j'} = 1;
    Calculate the path p with the maximum remaining bandwidth from j to j' in graph G;
    if b^r_e ≤ B_p then
        x^r_{e,{j,j'}} ← 1, ∀(j, j') ∈ p;
    end
    else
        x^r_{e,{j,j'}} ← 0, ∀e ∈ E_r, (j, j') ∈ L;
        Return False;
    end
end
```

Algorithm 4: Service Reconfiguration

Input: r, R, $G(N, L)$
Output: $z_r, \forall r \in R$

```
Release the physical resources occupied by service requests, denoted by W, whose total computation resource costs of each service request less than that of r;
Rank the service request r and all the services in W in the decreasing order of their total computation resource costs as W';
for r' ∈ W' do
    if Algorithm 2 with inputs V_r' and N succeeds then
        if Algorithm 3 with inputs y^{r'}_{i,j}, ∀i ∈ V_r', j ∈ N, E_r' and L succeeds then
            z_r' ← 0;
        end
        else
            z_r' ← 1;
        end
    end
    else
        z_r' ← 1;
    end
end
```

D. Problem Analysis

Theorem 1: The problem of cost-minimized computation offloading of multi-function service requests is NP-hard.

Proof: It can be obtained that all the constraints in the above problem are linear with respect to variables $y^r_{i,j}$, $x^r_{e,\{j,j'\}}$, and z_r except for Eq. (7). We first prove that the Eq. (7) can be converted into linear constraints by introducing a large number M and a very small number ε with $\varepsilon > 0$. The Big-M method can be utilized to construct Eqs. (9) and (10) based on (7):

$$z_r \leq M \cdot \left(1 - \frac{\sum_{j=1}^{|N|} \sum_{i=1}^{|V_r|} y^r_{i,j}}{|V_r|}\right), \tag{9}$$

$$z_r \geq M \cdot \frac{\sum_{j=1}^{|N|} \sum_{i=1}^{|V_r|} y^r_{i,j}}{|V_r|} + \varepsilon. \tag{10}$$

It can be obtained that if $\sum_{j=1}^{|N|} \sum_{i=1}^{|V_r|} y^r_{i,j} \neq 0$ indicating that the service request r is offloaded to the edge cloud other than the central cloud, then $\sum_{j=1}^{|N|} \sum_{i=1}^{|V_r|} y^r_{i,j} = |V_r|$. As a result, Eqs. (9) and (10) are equivalent to $z_r \leq 0$ and $z_r \geq M + \varepsilon$. Since z_r is a binary variable for any service request r, this gives $z_r = 0$. On the contrary, if $\sum_{j=1}^{|N|} \sum_{i=1}^{|V_r|} y^r_{i,j} = 0$, then $z_r \leq M$ and $z_r \geq \varepsilon$. Thus, $z_r = 1$ on account that both M and ε are positive with $M >> 1$ and $\varepsilon << 1$. In summary, Eq. (7) can be converted into two linear constraints, i.e., Eqs. (9) and (10).

Based on the fact that all the decision variables are positive and all the constraints are in or can be converted to linear forms, this problem is an integer linear programming. Since the ILP is NP-hard [41], [42], the problem of cost-minimized multi-function service offloading is also NP-hard. Therefore, we have proved the theorem.

IV. Heuristic Algorithm

In this section, service requests that arrive in sequence are offloaded to the collaborative edge-cloud network using the proposed CCOR algorithm including function offloading, link allocation, and service reconfiguration.

A. Function and Link Allocation

In **Algorithm 1**, for any service request r in R, the function offloading (line 2) is applied first, followed by the link allocation (line 3). The cost is calculated based on the

results of function offloading and link allocation as shown in Eq. (2).

The function offloading is achieved in **Algorithm 2**. For any service request r, all the functions in V_r are ranked in the decreasing order of their computation requirements as V_r'. Similarly, all the physical nodes in N are also ranked in the decreasing order of their remaining computation capacity as N' (lines 1-2). The edge server with higher remaining computation resource is selected for offloading the function with higher computation requirement (line 7). Once the remaining capacity of the kth edge server in N' cannot meet the resource requirement of the kth function in V_r', the algorithm fails.

Based on the results of function offloading for a service request r, i.e., $y_{i,j}^r, \forall i \in V_r, j \in N$, the allocation of links among functions is described in **Algorithm 3**. Any link e between two functions e_s and e_d is placed to a loop-free path with the maximum remaining bandwidth capacity between the offloaded edge nodes j and j' of the source and the destination of the link, i.e., $y_{e_s,j}^r = 1$ and $y_{e_d,j'}^r = 1$ (lines 2-5). The algorithm fails if we fail to find a path with sufficient bandwidth resource for any link e.

B. Service Reconfiguration

The failure of either function offloading (line 7 in **Algorithm 1**) or link allocation (line 11 in **Algorithm 1**) will trigger the service reconfiguration as described in **Algorithm 4**.

The service reconfiguration related to service r is stated in **Algorithm 4**, we release the physical resources occupied by service requests, denoted by W, whose total computation resource costs are less than that of r (line 1). Then the released services are re-offloaded in order (lines 2-3). **Algorithms 2 and 3** are utilized again for the re-offloading. Any service that can not be re-offloaded to the edge cloud network are uploaded to the central cloud (lines 9 and 13).

C. Complexity Analysis

The time complexity of **Algorithm 1** consists of three parts. First, performing the function offloading of $|R|$ service requests, i.e., **Algorithm 2**, costs $O\left(|V_r| \cdot \log |V_r|\right)$ for function ranking, $O\left(|N| \cdot \log |N|\right)$ for node ranking, and $O\left(|V_r|\right)$ for function allocation. Thus the time complexity of **Algorithm 2** is $O\left(|N| \cdot \log |N|\right)$ since generally we consider $|V_r| < |N|$. Second, finding the path with the maximum remaining bandwidth for each link E_r costs $O(|N|^3)$. Therefore, the time complexity of **Algorithm 3** can be expressed as $O(|E_r| \cdot |N|^3)$. For the service reconfiguration in **Algorithm 4**, node ranking is performed for $|W|$ service requests and the time complexity is $O\left(|W| \cdot |N| \cdot \log |N|\right)$. Then the path of the maximum remaining bandwidth is searched. The time complexity can be expressed as $O(|W| \cdot |E_r| \cdot |N|^3)$. Reconfiguring at most R services, i.e., $|W| < |R|$, consumes $O\left(|R| \cdot \log |R|\right)$ steps for service ranking and $O\left(|R| \cdot \left(|N| \cdot \log |N| + |E_r| \cdot |N|^3\right)\right)$ steps for re-offloading. Thus, the time complexity of **Algorithm 4** is $O\left(|R| \cdot \log |R| + |R| \cdot |E_r| \cdot |N|^3\right)$. Finally, the total time complexity of **Algorithm 1** is expressed by $O\left(|R| \cdot \log |R| + |R| \cdot |E_r| \cdot |N|^3\right)$ for $|R|$ service requests, which is polynomial time.

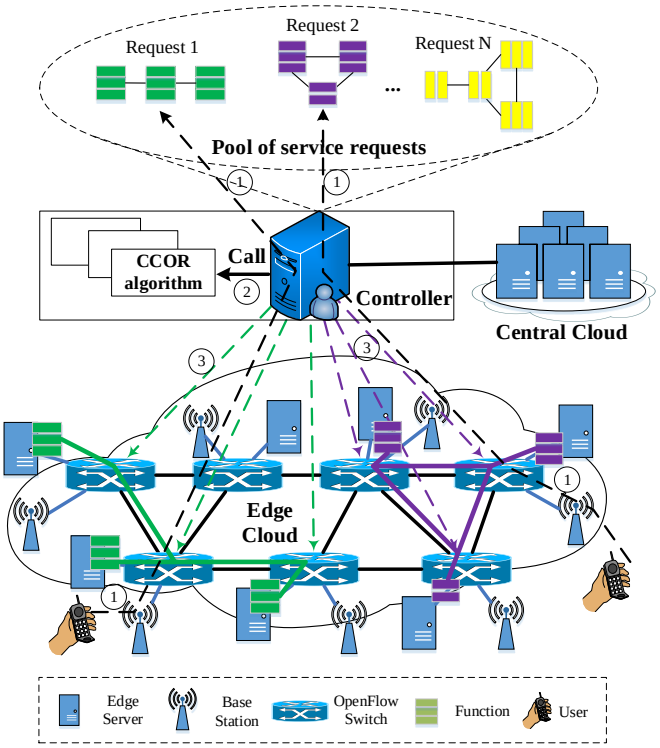


Fig. 2: SDN-based framework

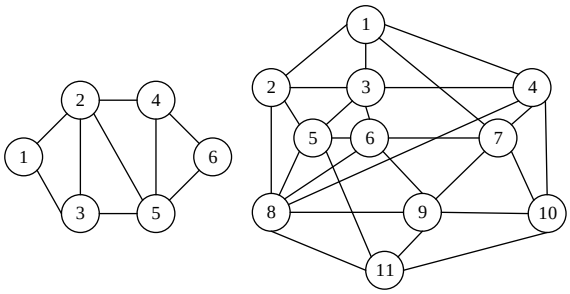
D. SDN-Based Framework

In the SDN-based framework as shown in Fig.2, the edge servers are connected by OpenFlow switches. The BS is usually co-located with a switch for the purpose of serving users and network management. The network management and control is realized in a centralized manner with a controller. The controller communicates with underlying network equipment via the OpenFlow protocol. The underlying equipment performs routing operations according to the flow entries distributed by the controller. In Fig. 2, when a service request arrives at the OpenFlow switch through a BS. The OpenFlow switch sends this request to the controller by the Packet_In message (Step 1). When multiple service requests arrive at the same time, the controller forms a centralized request pool. Then, the controller calls the CCOR algorithm to offload the calculation requests (Step 2). Finally, based on the result of the CCOR algorithm, the controller distributes the flow entries to the corresponding switch by the Flow_Mod message (Step 3), thereby offloading each function in the request to the corresponding edge server. The flow entry inserted into the edge node is quantitatively evaluated in Section V.

V. Simulation Results

A. Parameter Settings

We conduct two edge cloud network topologies for the simulations, including the small-scale network with $N = 6$, $L = 9$, i.e., n6s9 and the large-scale network with $N = 11$, $L = 26$, i.e., n11s26 as in Fig. 3. The total bandwidth of each link is 100 Mbps. The computation capacity of each edge node is 96 CPUs. It is also worth noting that our algorithm is



(a) 6-node, 9-link n6s9 network topology

(b) 11-node, 26-link n11s26 network topology

Fig. 3: Network topologies used in the simulation

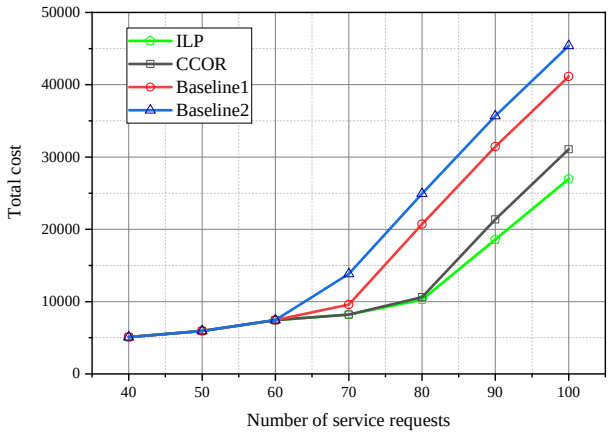


Fig. 4: Total cost of different methods over varying number of service requests in n6s9

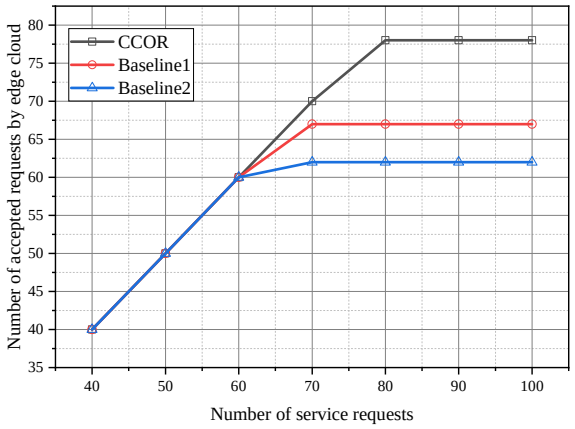


Fig. 5: Number of service requests offloaded to the edge cloud in different methods in n6s9

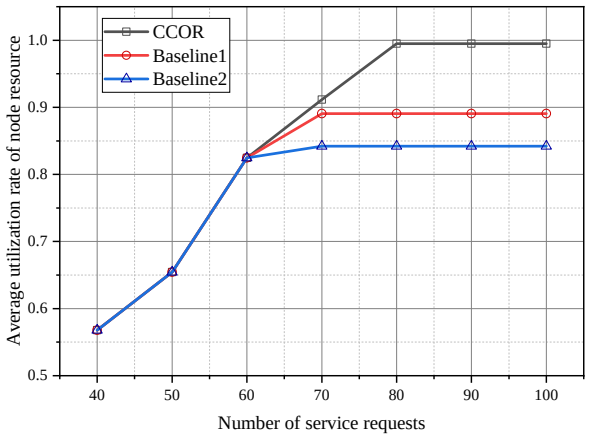


Fig. 6: Average utilization rate of node resource of different methods in n6s9

running in a virtual machine with a 2-core CPU and 4GB of memory.

The number of links in the rth service request is randomly generated in $\left[|V_r|-1, \frac{|V_r|(|V_r|-1)}{2}\right]$. The computation resource required by each function in a service request ranges within $[1,4]$ CPUs. The bandwidth requested by each link in a service request is in $[1,6]$ Mbps. For the n6s9 topology, the number of service requests increases from 40 to 100. The large-scale topology (n11s26) is evaluated under the number of service requests increases from 30 to 400. We set the number of functions in each service in [2-4] randomly when the number of functions in each service is not specifically described.

The proposed CCOR algorithm is compared with three approaches as follows.

- ILP: using the IBM CPLEX optimizer [42] to solve the ILP in Section III.C.
- Baseline1 [29]: allocating functions of each service request to edge servers using maximum bipartite matching and routing links to the shortest paths.
- Baseline2 [43], [44]: allocating functions of each service request using Algorithm2 and routing links to the shortest paths.

The performance metrics for evaluating the proposed algorithm are listed as follows.

- Total cost is defined in Eq. (2). Moreover, we set $\lambda = 10$, $\beta = 5$, and $\delta = 100\alpha$, where α is defined as

$$\alpha = \frac{\sum_{i=1}^{|V_r|} \lambda \cdot c_i^r + \sum_{e \in E_r} \beta \cdot b_e^r}{\sum_{i=1}^{|V_r|} \lambda \cdot c_i^r}. \tag{11}$$

- The number of service requests offloaded to the edge cloud in different networks.
- The average utilization rate of node resource η is defined as the ratio of the computation resource allocated to service requests over the total computation resource of all nodes in the edge cloud:

$$\eta = \frac{\sum_{r=1}^{|R|} \sum_{i=1}^{|V_r|} c_i^r \cdot (1 - z_r)}{\sum_{j=1}^{|N|} C_j}. \tag{12}$$

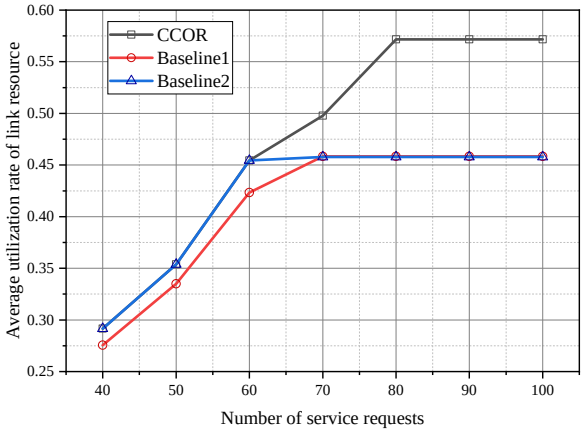


Fig. 7: Average utilization rate of link resource of different methods in n6s9

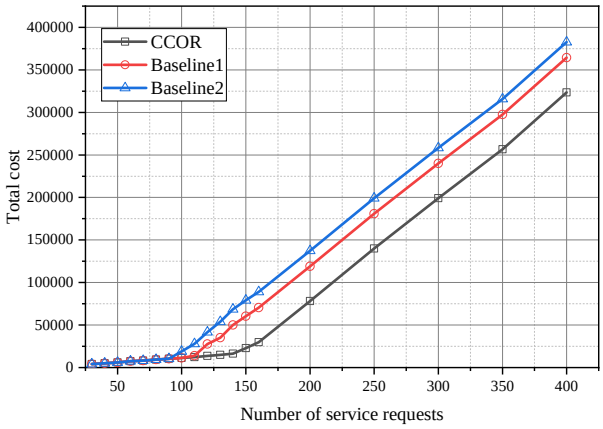


Fig. 9: Total cost for random number of functions in each service request in n11s26

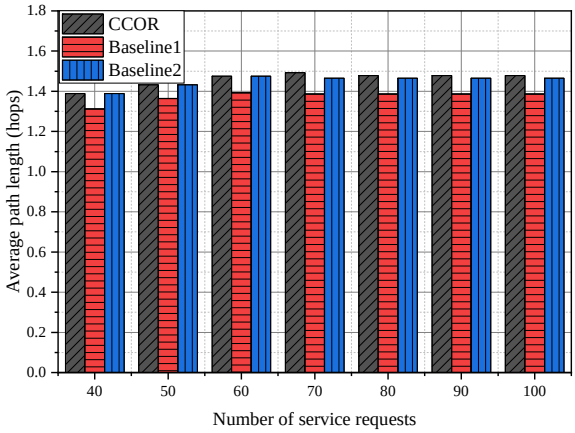


Fig. 8: Average path length (hops) for link allocation in different methods in n6s9

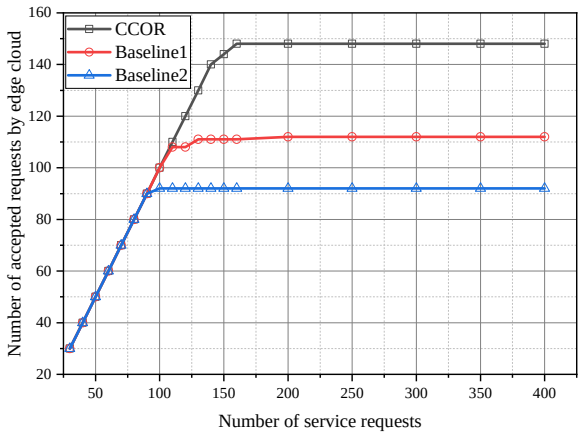


Fig. 10: Number of service requests offloaded to the edge cloud of different methods for random number of functions in each service request in n11s26

- The average utilization rate of link resource ξ is define as the ratio of the bandwidth allocated to service requests over all the bandwidth resource of the edge cloud:

$$\xi = \frac{\sum_{r=1}^{|R|} \sum_{e \in E_r} b_e^r \cdot (1 - z_r)}{\sum_{(j,j') \in L} B_{(j,j')}}. \tag{13}$$

- The average path length in hops χ is defined as the average path length evaluated by number of hops for allocating links in services:

$$\chi = \frac{\sum_{r=1}^{|R|} \sum_{e \in E_r} \frac{|p_e^r|}{|E_r|} \cdot (1 - z_r)}{\sum_{r=1}^{|R|} (1 - z_r)}. \tag{14}$$

- Running time is in milliseconds.
- The variance of link resource utilization.

B. Performance of CCOR in Small-Scale Networks

For the n6s9 topology, the total cost of different approaches under the different number of service requests is demonstrated in Fig. 4. The total cost of the proposed CCOR is obviously less than that of the Baselines and is close to the optimal cost obtained by ILP. At most 48.69% and 57.40% of the cost can be saved by CCOR compared with Baseline1 and Baseline2. Only 3.51% higher cost is induced in CCOR compared with ILP solutions.

To improve the quality of services, service requests are preferred to be offloaded to the edge cloud to obtain a lower resource cost and delay compared with uploading service requests to the central cloud. The number of accepted service requests by edge cloud in different methods is shown in Fig. 5. It can be obtained that CCOR offloads more services to edge servers than Baselines, thus better QoS of users can be provided by CCOR.

Since more services are offloaded to the edge cloud, higher average resource utilization rates of nodes and links are achieved by CCOR as demonstrated in Figs. 6 and 7, indicating that CCOR can make full use of node and link resources. We can see from Fig. 8 that the average path length (hops)

TABLE II: Running time of different methods for random number of functions in each service request in n11s26

Number of service requests	40	60	80	100	120	140
CCOR (ms)	5.9895	11.0018	13.9658	15.9568	19.9775	22.9470
Baseline1 (ms)	983.6473	975.7850	993.0146	1004.4097	1008.7413	1025.0804
Baseline2 (ms)	2.9890	3.9885	5.9852	6.9804	10.9562	11.9397

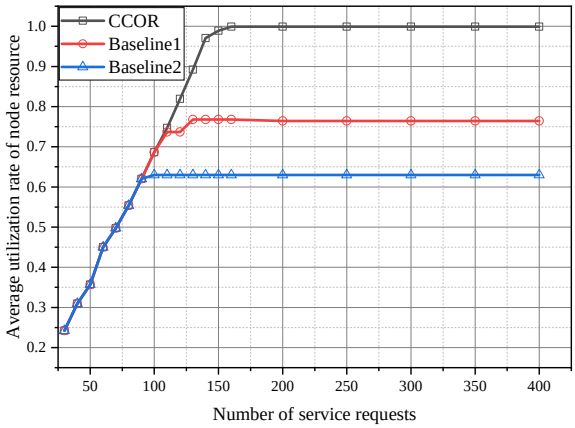


Fig. 11: Average utilization rate of resource in edge nodes of different methods for random number of functions in each service request in n11s26

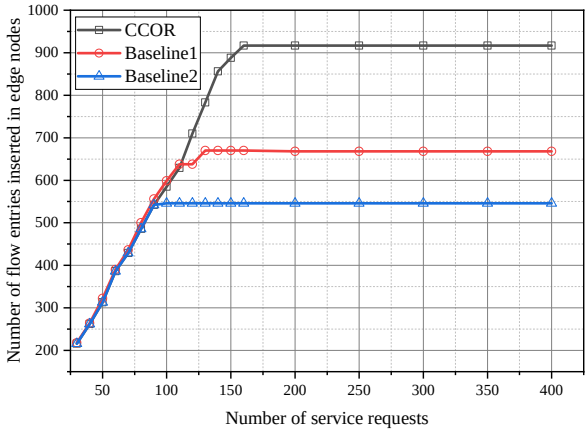


Fig. 13: Number of flow entries of different methods for random number of functions in each service request in n11s26

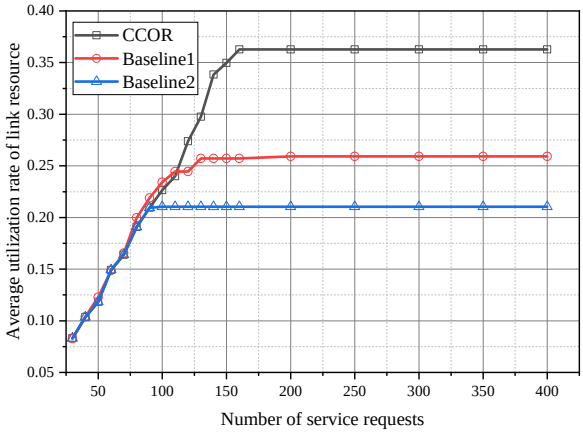


Fig. 12: Average utilization rate of link resource of different methods for random number of functions in each service request in n11s26

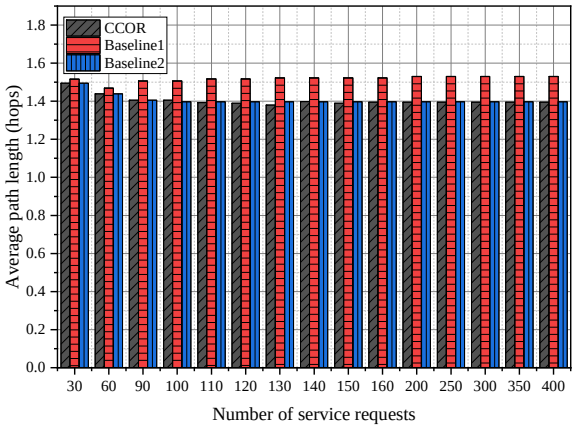


Fig. 14: Average path length (hops) for link allocation in different methods for random number of functions in each service request in n11s26

for link allocation in CCOR is slightly higher than Baselines.

C. Performance of CCOR in Large-Scale Networks

Figure 9 shows the total cost of the CCOR algorithm compared with the Baselines. When the number of services increases, Baseline1 cannot find enough resources in the edge network to accommodate more computing requests. More services are offloaded to the central cloud resulting in a rapidly increased cost. CCOR consumes less cost than the Baselines by offloading more services to the edge cloud as shown in Fig. 10, which results in higher average utilization of node and link resources in the edge cloud of CCOR, as shown in Figs. 11 and 12.

As shown in Table II, the running time of CCOR is lower than Baseline1. The reason for this phenomenon is that the time complexity of the Baseline1 is higher than that of CCOR. For instance, when the number of service requests is 120, the execution time of the Baseline1 is about 1008.7413 milliseconds (ms), but the CCOR algorithm takes 19.9775ms. Note that, the cost of CCOR is 40.01% lower than Baseline 1 at this point. The running time in CCOR algorithm is slightly larger than that of Baseline2 for searching more efficient offloading results, but the cost of Baseline2 is obviously larger than that of CCOR.

In addition, based on the SDN deployment architecture in [45], the flow entry inserted into edge servers is also

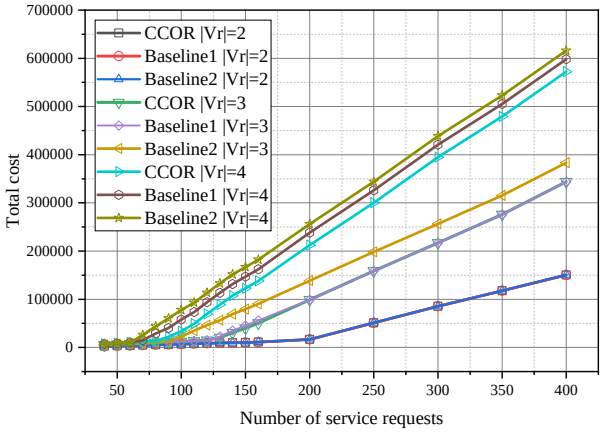


Fig. 15: Total cost of different methods with different fixed numbers of functions in each service request in n11s26

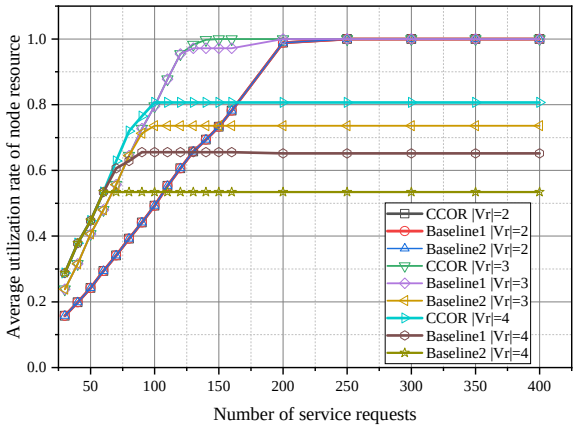


Fig. 17: Average utilization rate of resource in edge nodes with different fixed numbers of functions in each service request in n11s26

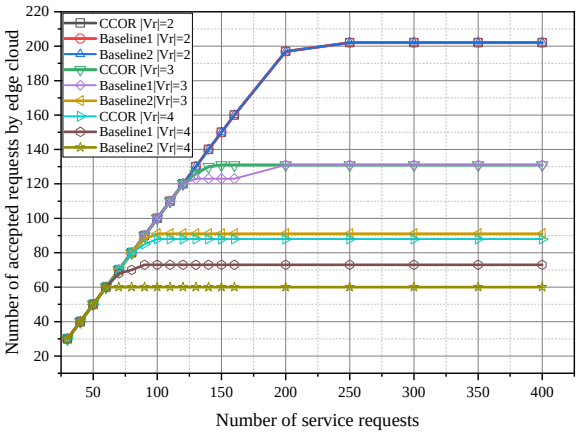


Fig. 16: Number of service requests offloaded to the edge cloud of different methods with different fixed numbers of functions in each service request in n11s26

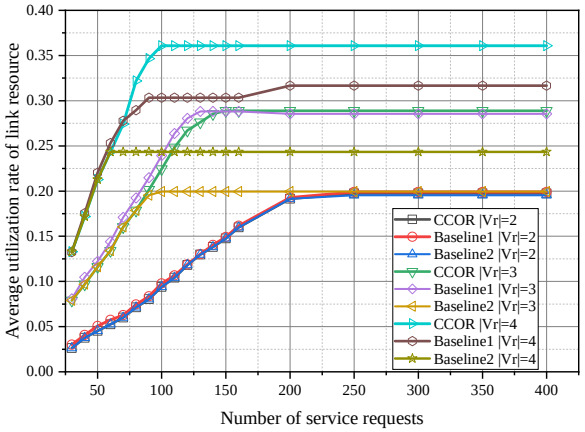


Fig. 18: Average utilization rate of link resource with different fixed numbers of functions in each service request in n11s26

quantitatively evaluated in Fig. 13 where the more service requests carried by the edge network, the more corresponding flow entries. The average path length (hops) for link allocation in Baselines is higher than that of CCOR as depicted in Fig. 14.

Furthermore, we conduct the performance comparison of different algorithms for service requests with fixed 2, 3, and 4 functions, respectively. Figure 15 shows the total cost. When the number of functions in each service request is 2, the total cost of the CCOR and Baselines is the same. The reason is that a simple service request is easier to be placed, and the edge network has sufficient resources. However, when the number of functions is 3 and 4, the total cost of the CCOR algorithm is significantly lower than Baselines.

Figs. 16-19 demonstrate the superiority of CCOR over Baselines in the number of service requests offloaded to the edge cloud, node resource utilization, link resource utilization, and the number of flow table entries, where CCOR also outperforms the Baselines when the number of functions in each service request is 3 and 4. Moreover, CCOR shows similar average path length (hops) to the Baselines as shown in Fig. 20.

To demonstrate the difference between CCOR and Baselines, we evaluate the variance of link resource utilization. As shown in Fig. 21, the link variance value of CCOR is lower than Baselines, indicating that the loop-free path selection method with the maximum remaining bandwidth capacity in CCOR promotes load balancing. As a result, more service requests can be offloaded to the edge cloud by CCOR.

Therefore, CCOR outperforms Baseline1 due to the following reasons:

- CCOR has lower time complexity. In the bipartite matching based function offloading process, the time complexity of building the bipartite graph is $O\left(|V_r| + |N| + |V_r| \cdot |N|\right)$, and the time complexity of maximum weight full matching is $O\left(|V_r| \cdot |N| \cdot \log |N|\right)$. Therefore, the time complexity of the maximum bipar-

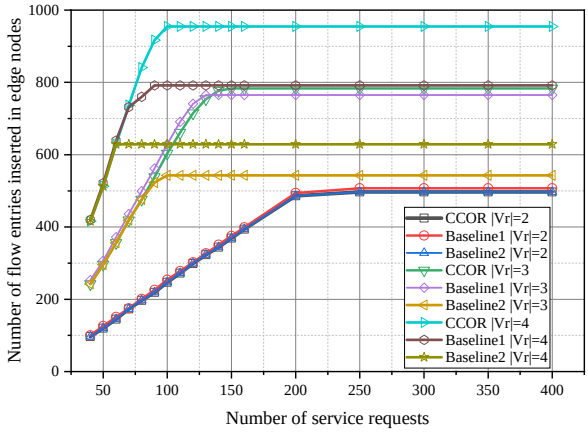


Fig. 19: Number of flow entries of different methods with different fixed numbers of functions in each service request in n11s26

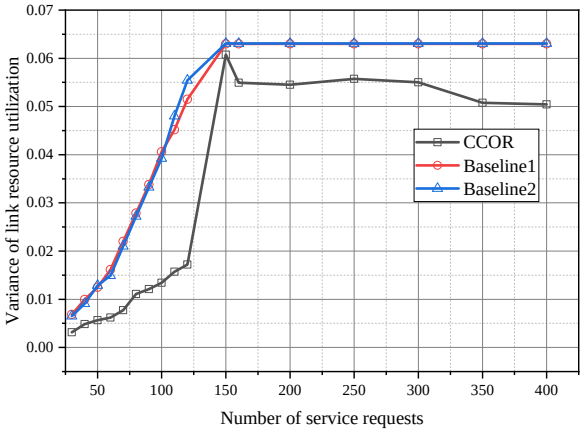


Fig. 21: Variance of link resource utilization with fixed number of functions in each service request in n11s26

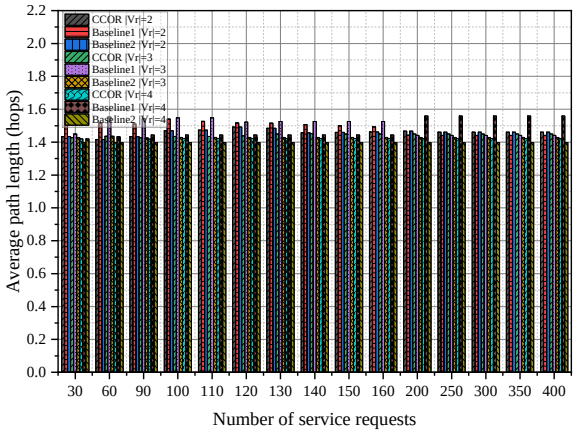


Fig. 20: Average path length (hops) for link allocation with different fixed numbers of functions in each service request in n11s26

tite matching can be expressed as $O\left(|V_r| \cdot |N| \cdot \log |N|\right)$ [46]. The time complexity of function offloading in CCOR is $O\left(|N| \cdot \log |N|\right)$, which is lower than that of the maximum bipartite matching in Baseline1. Experimentally as shown in Table II, CCOR experiences a faster running time than Baseline1. Therefore, CCOR experiences a faster running time than Baseline1.

- It should be noted that the function allocation results of CCOR are equivalent to that of the Baseline1 theoretically. However, CCOR applies the link allocation based on the maximum remaining bandwidth, instead of finding the shortest path between the offloaded edge nodes for a link request in Baseline1, resulting in better load balancing among physical links, as shown in Fig. 21. Thus, more services can be placed on the edge cloud in CCOR with lower resource cost.
- CCOR relies on the central cloud to ensure that all the service requests can be offloaded successfully compared to the pure edge cloud networks in [29].

Overall, CCOR consumes less cost compared with Baselines with higher resource utilization rates and comparable delay under different scales of physical networks, varying number of functions in service requests, and a wide range of the number of service requests.

VI. Conclusion

In this paper, we investigate the cost-minimized computation offloading in collaborative edge-cloud networks by characterizing multiple functions in each service request. The problem is formulated mathematically and is proved to be NP-hard. Based on the formulation, we propose a CCOR algorithm to offload more service requests to edge cloud networks. The service reconfiguration is applied to facilitate accepting more online-arrived services. Different topologies and different service request scenarios are selected to verify the effectiveness and scalability of the CCOR algorithm. The simulation results show that the proposed algorithm outperforms the baseline algorithms in resource utilization and the total cost.

Acknowledgments

This work was supported in part by the National Key R&D Program of China under Grant 2018YFE0206800, in part by the National Natural Science Foundation of China under Grant U21B2005, in part by the Natural Science Foundation of China under Grant 62025105, in part by the Chongqing Municipal Education Commission under Grant CXQT21019, in part by the Nature Science Foundation of Chongqing under Grant cstc2021jcyj-msxmX0404, in part by the China Postdoctoral Science Foundation under Grant 2021M700563, and in part by the Chongqing Postdoctoral Funding Project under Grant 2112012727685993.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, pp. 50–58, 2010.

[2] C. Gong, J. Liu, Q. Zhang, H. Chen, and Z. Gong, "The characteristics of cloud computing," in *2010 39th International Conference on Parallel Processing Workshops*, 2010, pp. 275–279.

[3] X. Zhang, H. Chen, Y. Zhao, Z. Ma, Y. Xu, H. Huang, H. Yin, and D. O. Wu, "Improving cloud gaming experience through mobile edge computing," *IEEE Wireless Communications*, vol. 26, no. 4, pp. 178–183, 2019.

[4] L. Dong, N. Fonseca, and Z. Zhu, "Application-driven provisioning of service function chains over heterogeneous nfv platforms," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3037–3048, 2021.

[5] C. Xu, J. Wang, Z. Zhu, and D. Niyato, "Energy-efficient wlans with resource and re-association scheduling optimization," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 563–577, 2019.

[6] X. Zhang, W. Hou, L. Guo, Q. Zhang, P. Guo, and R. Li, "Joint optimization of latency monitoring and traffic scheduling in software defined heterogeneous networks," *Mobile Networks and Applications*, vol. 25, pp. 102–113, 2020.

[7] C. Yi, J. Cai, and Z. Su, "A multi-user mobile computation offloading and transmission scheduling mechanism for delay-sensitive applications," *IEEE Transactions on Mobile Computing*, vol. 19, no. 1, pp. 29–43, 2020.

[8] Z. Ning, P. Dong, X. Wang, X. Hu, J. Liu, L. Guo, B. Hu, R. Kwok, and V. C. M. Leung, "Partial computation offloading and adaptive task scheduling for 5g-enabled vehicular networks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 4, pp. 1319–1333, 2022.

[9] Y. Li, X. Wang, X. Gan, H. Jin, L. Fu, and X. Wang, "Learning-aided computation offloading for trusted collaborative mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 19, no. 12, pp. 2833–2849, 2020.

[10] B. Gao, Z. Zhou, F. Liu, and F. Xu, "Winning at the starting line: Joint network selection and service placement for mobile edge computing," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 1459–1467.

[11] U. Saleem, Y. Liu, S. Jangsher, X. Tao, and Y. Li, "Latency minimization for d2d-enabled partial computation offloading in mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4472–4486, 2020.

[12] W. Zhang, Y. Wen, and D. O. Wu, "Collaborative task execution in mobile cloud computing under a stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 14, no. 1, pp. 81–93, 2015.

[13] P. Han, Y. Liu, and L. Guo, "Interference-aware online multicomponent service placement in edge cloud networks and its ai application," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10 557–10 572, 2021.

[14] M. Amiri, H. A. Osman, S. Shirmohammadi, and M. Abdallah, "Toward delay-efficient game-aware data centers for cloud gaming," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 12, no. 5s, pp. 1–19, 2016.

[15] Y. Li, Y. Deng, X.Tang, W. Cai, X. Liu, and G. Wang, "Cost-efficient server provisioning for cloud gaming," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 14, no. 3s, pp. 1–22, 2018.

[16] Y. Deng, Y. Li, R. Seet, X. Tang, and W. Cai, "The server allocation problem for session-based multiplayer cloud gaming," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1233–1245, 2018.

[17] S. Pandi, R. S. Schmoll, P. J. Braun, and F. H. P. Fitzek, "Demonstration of mobile edge cloud for tactile internet using a 5g gaming application," in *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2017, pp. 607–608.

[18] A. Ksentini, T. Taleb, and M. Chen, "A markov decision process-based service migration procedure for follow me cloud," in *2014 IEEE International Conference on Communications (ICC)*, 2014, pp. 1350–1354.

[19] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. S. Shen, "Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach," *IEEE Transactions on Mobile Computing*, vol. 20, no. 3, pp. 939–951, 2021.

[20] P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Risk-aware data offloading in multi-server multi-access edge computing environment," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1405–1418, 2020.

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

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

[23] M. Liu and Y. Liu, "Price-based distributed offloading for mobile-edge computing with computation capacity constraints," *IEEE Wireless Communications Letters*, vol. 7, no. 3, pp. 420–423, 2018.

[24] T. Zhang and W. Chen, "Computation offloading in heterogeneous mobile edge computing with energy harvesting," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 1, pp. 552–565, 2021.

[25] Y. Sahni, J. Cao, L. Yang, and Y. Ji, "Multi-hop multi-task partial computation offloading in collaborative edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1133–1145, 2021.

[26] L. Li, T. Q. S. Quek, J. Ren, H. H. Yang, Z. Chen, and Y. Zhang, "An incentive-aware job offloading control framework for multi-access edge computing," *IEEE Transactions on Mobile Computing*, vol. 20, no. 1, pp. 63–75, 2021.

[27] L. Qu, C. Assi, and K. Shaban, "Delay-aware scheduling and resource optimization with network function virtualization," *IEEE Transactions on Communications*, vol. 64, no. 9, pp. 3746–3758, 2016.

[28] B. Yang, W. K. Chai, Z. Xu, K. V. Katsaros, and G. Pavlou, "Cost-efficient nfv-enabled mobile edge-cloud for low latency mobile applications," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 475–488, 2018.

[29] Z. Xu, W. Liang, M. Jia, M. Huang, and G. Mao, "Task offloading with network function requirements in a mobile edge-cloud network," *IEEE Transactions on Mobile Computing*, vol. 18, no. 11, pp. 2672–2685, 2019.

[30] R. Durner and W. Kellerer, "Network function offloading through classification of elephant flows," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 807–820, 2020.

[31] M. Jia, J. Cao, and L. Yang, "Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing," in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2014, pp. 352–357.

[32] S. E. Mahmoodi, R. N. Uma, and K. P. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 301–313, 2019.

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

[34] L. Long, Z. Liu, Y. Zhou, L. Liu, J. Shi, and Q. Sun, "Delay optimized computation offloading and resource allocation for mobile edge computing," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, 2019, pp. 1–5.

[35] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3435–3447, 2017.

[36] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2019.

[37] S. Jošilo and G. Dán, "Computation offloading scheduling for periodic tasks in mobile edge computing," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 667–680, 2020.

[38] C. Feng, P. Han, X. Zhang, B. Yang, Y. Liu, and L. Guo, "Computation offloading in mobile edge computing networks: A survey," *Journal of Network and Computer Applications*, vol. 202, p. 103366, 2022.

[39] M. Deng, H. Tian, and B.Fan, "Fine-granularity based application offloading policy in cloud-enhanced small cell networks," in *2016 IEEE International Conference on Communications Workshops (ICC)*, 2016, pp. 638–643.

[40] C. Singhal and S. De, *Resource Allocation in Next-Generation Broadband Wireless Access Networks*. IGI Global, 2017.

[41] A. Ceselli, M. Premoli, and S. Secci, "Mobile edge cloud network design optimization," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1818–1831, 2017.

[42] A. Jarray and A. Karmouch, "Cost-efficient mapping for fault-tolerant virtual networks," *IEEE Transactions on Computers*, vol. 64, no. 3, pp. 668–681, 2015.

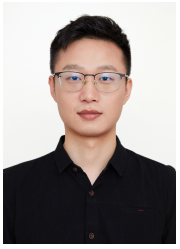
[43] Z. Wang, Y. Han, T. Lin, H.Tang, and S.Ci, "Virtual network embedding by exploiting topological information," in *2012 IEEE Global Communications Conference (GLOBECOM)*, 2012, pp. 2603–2608.

[44] S. Misra and S. Bera, "Soft-van: Mobility-aware task offloading in software-defined vehicular network," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2071–2078, 2020.

[45] S. Yang, F. Li, M. Shen, X. Chen, X. Fu, and Y. Wang, "Cloudlet placement and task allocation in mobile edge computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5853–5863, 2019.
[46] R. M. Karp, "An algorithm to solve the m × n assignment problem in expected time o(mn log n)," *Networks*, vol. 10, no. 2, pp. 143–152, 1980.

Chuan Feng is currently pursuing the Ph.D. degree in communication and information systems at Northeastern University, Shenyang, China. Her research interests are in mobile edge computing networks, with an emphasis on resource allocation techniques for computation offloading.

Pengchao Han received the Ph.D. degree in communication and information systems at Northeastern University, Shenyang, China. She is currently a Postdoc research associate at The Chinese University of Hong Kong, Shenzhen, China. Her research interests include wireless and optical networks, mobile edge computing, federated learning, and knowledge distillation.



Xu Zhang (Member, IEEE) received the B. Eng. degree in 2014 and the Ph.D. degree in communication and information systems in 2019 from Northeastern University, Shenyang, China. From 2017 to 2018, he conducted academic research with the University of Tennessee, Knoxville, TN, USA. He is currently a Lecturer with the School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing, China. His research interests include software-defined networking, optical network, resilient communication, traffic engineering, and network optimization. He has published over 20 technical papers in the above areas in international journals and conferences. Dr. Zhang was the recipient of the Best Paper Award of Qshine, 2017. He is also a member of OPTICA.

Qihan Zhang received the M.S. in communication and information system in the School of Computer Science and Engineering, Northeastern University, Shenyang, China in 2018. He is currently a Ph.D. student of communication and information system in Northeastern University, Shenyang, China. His research interests include optical signal processing and secure optical transmission system.

Yue Zong received the Ph.D. degree in communication and information systems from Northeastern University, Shenyang, China, in 2021. From 2016 to 2017, she was a visiting student with University of Bristol, Bristol, UK. She is currently the Postdoc in Power China Huadong Engineering Corporation Limited, Hangzhou, China. Her research interests include energy communication, network virtualization.

Yejun Liu (Member, IEEE) received the Ph.D. degree in communication and information systems from Northeastern University, Shenyang, China, in 2015. He is currently a professor in the School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, China. His research interests include wireless optical communication and converged fiber-wireless access network.

Lei Guo (Member, IEEE) received the Ph.D. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2006. He is currently a Professor with the Chongqing University of Posts and Telecommunications, Chongqing, China. He has authored or coauthored more than 200 technical papers in the above areas in international journals and conferences, such as the IEEE Transactions on Communications, the IEEE Transactions on Wireless Communications, the IEEE/OSA Journal of Lightwave Technology, the IEEE/OSA Journal of Optical Communications and Networking, the IEEE GLOBECOM, and the IEEE ICC. His current research interests include communication networks, optical communications, and wireless communications. He is a member of OSA, and also a Senior Member of CIC. He is currently an editor for five international journals.