Energy-Efficient Service Placement based on Equivalent Bandwidth in Cell Zooming enabled Mobile Edge Cloud Networks

Pengchao Han, Yejun Liu, Xu Zhang, Lei Guo

***Abstract*— Mobile edge computing is popular for providing services with low latency and high privacy. It becomes more powerful by leveraging the widely deployed Small cell Base Stations (SBS) with zooming cells for energy-efficient service placement. However, the literature ignores the joint optimization of SBS power control and service placement. The Quality of Services (QoS) guarantee is also challenging, taking into account the component dependency and parallelization in multi-component services and the resource slicing for separated service deployment. Moreover, prior works assume the same amount of allocated resources of links on a placed path for service edge allocation, leading to the ossification of resource allocation and inevitable resource fragments. Towards the above challenges, this paper addresses the energy-efficient and flexible service placement in cell zooming enabled Mobile Edge Cloud (MEC) networks. The delay of multi-component services is constructed depending on the allocated resource slice. Besides, the joint optimization of service placement and SBS power control is formulated and transformed into a Mixed Integer Linear Programming (MILP). More importantly, the equivalent bandwidth for an edge allocation is defined and analyzed to obtain flexible edge placement with minimum resource cost. Leveraging the results of MILP, an Energy-efficient Service placement algorithm based on equivalent Bandwidth in Cell zooming enabled MEC networks (ESBC) is proposed to improve the probability of successful service placement with QoS guarantee by optimizing the delay distribution among components and edges and reducing resource fragments. Finally, simulation results validate the effectiveness of the proposed methods.**

***Index Terms*—Service placement, mobile edge cloud, cell zooming, energy-efficient, equivalent bandwidth.**

I. Introduction

THE highly developed communication technology 5G [1], [2] has enabled unprecedented Quality of Services (QoS) guarantee on a diversity of services for the ever-increasing number of users. Facing the emergence of new applications in big data, the Internet of Things (IoT), Virtual Reality (VR), and Augmented Reality (AR) [3], etc., resources in networks have evolved from single communication resource to multi-dimensional resources including communication, computation, and memory. Cloud computing [4] has provided a new pattern for service provisioning where users can request services from cloud servers instead of processing data locally, resulting in higher efficiency of resources, simplified user devices, and easier service deployment. However, the central cloud features a long transmission distance and high service delay, which is intolerable for real-time applications. Towards the challenge, mobile edge computing [5], [6], [7] emerges thanks to the ubiquitous Small cell Base Stations (SBS) in 5G, low-cost computation servers, high-capacity storage devices, and popular Graphic Processing Units (GPU) [8], [9]. It is a promising technique for future multi-component services and intelligent applications, promoting the development of edge intelligence in 6G [10], [11], [12], [13].

In Mobile Edge Cloud (MEC) networks, edge servers are deployed beside SBSs to support services placed in close proximity to users, stimulating the low-latency service provisioning with high privacy. Each service in MEC networks characterizes multiple interconnected components. Edges in services specify the dependency among components. The resource-limited edge servers work cooperatively to provide large-scale multi-component services with QoS guarantee [14]. Besides, all SBSs construct a Voronoi diagram in MEC networks to achieve the full coverage of users. There is a high potential for a flexible user-SBS association through optimizing the Voronoi diagram via **cell zooming** i.e., adjusting the transmitting power of SBSs to coordinate the coverage of cells. Furthermore, exploiting different power states of SBSs, i.e., active and sleep, for energy-efficient service placement is also promising for developing MEC networks with enormous SBSs. The multi-component service placement in cell zooming enabled MEC networks features the joint optimization of SBS power control and service placement, which is neglected in prior works.

The QoS guarantee of multi-component services is also challenging due to each service's component dependency and parallelization. For a user, it is more practical to consider the delay constraint for the whole service rather than specifying the delay request of every component and edge. As a result, the delay distribution among the components and edges of a service is reconfigurable, leading to new difficulties and opportunities for successful service placement. Moreover, the heterogeneity of physical resources, e.g., SBSs with different transmitting powers and data rates and edge servers with diverse computation capacities, poses challenges for service placement. Specifically, works in the literature specify that all links on the placed path of a service edge allocate the same amount of resources to the edge. It leads to a high

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

Yejun Liu (Corresponding author), Xu Zhang, and Lei Guo are with the School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, 400065, China. E-mail: {yjliu, zhangxu, guolei}@cqupt.edu.cn.

probability of failed edge placement caused by bottleneck links and inevitable resource fragments. Thus, flexible resource allocation mechanisms are necessary.

Towards the above challenges, this paper aims to address the multi-component service placement problem in cell zooming enabled MEC networks. We mainly make the following contributions.

a) We mathematically formulate the delay of multi-component services considering allocating separated resource slices for each user;
b) The joint optimization of service placement and SBS power control in MEC networks is formulated as a Mixed Integer Non-Linear Programming (MINLP) and transformed into a Mixed Integer Linear Programming (MILP) through uniform delay splitting;
c) For the first time, we define the equivalent bandwidth for an edge placement to support flexible and diverse resource allocation of links on a placed path. Theoretically, we prove that the minimum resource consumption can be achieved in the principle of equivalent bandwidth;
d) We propose the Energy-efficient Service placement algorithm based on equivalent Bandwidth in Cell zooming enabled MEC networks (ESBC) to reduce resource fragments, improve service acceptance ratio, and guarantee the QoS of users. Iterative delay splitting and resource allocation are applied, taking the reconfigurable delay distribution of each service into account.

The remainder of this paper is organized as follows. The related works are reviewed in Section II. Section III presents the system models, including SBS power model and service delay. Then, the problem of multi-component service placement in cell zooming enabled MEC networks is formulated. The problem is transformed into a MILP in Section IV to achieve the joint optimization of SBS power control and service placement. The equivalent bandwidth is defined for service edges, based on which the edge allocation based on equivalent bandwidth and the ESBC algorithm is proposed in Section V. The simulation results are presented in Section VI. Finally, Section VII concludes this paper.

II. Related Works

To satisfy the emerging high-QoS applications and ever-increasing traffic demand for mobile communication systems, the next-generation 5G networks have evolved into a heterogeneous structure., i.e., 5G HetNet [15]. Various SBSs, e.g., femtocell BS, microcell BS, and picocell BS [16], coexist with Macrocell BS (MBS), posing new challenges to efficient resource allocation [15]. Optimizing the Voronoi diagram of SBSs has been investigated for supporting traditional communication traffics. The game theory [17] is utilized for constructing the Voronoi diagram of SBSs for serving mobile users. The constraint of outage probability [18] and the wireless energy harvesting [19], [20] are also considered for optimizing the power of zooming small cells.

Related works save energy through dynamic power control for providing traditional unsplittable services, i.e., services with only one component [21], to mobile users in MEC networks. Authors in [22] have optimized the power allocation in non-orthogonal multiple access (NOMA) based wireless networks to guarantee the QoS of users and improve the spectrum efficiency. Cell zooming is an effective technique to ensure the continuity of service provisioning while users move among multiple cells [23], especially for MEC networks with massive and large-scale deployed devices [24]. When a user moves out of the coverage of the current serving SBS, there are two ways to ensure the QoS of the user: (a) the SBS increases the transmitting power to enlarge its coverage, and (b) another SBS is selected to serve the user. Given the distribution of users, it is also promising to save energy by gathering users into several SBSs through cell zooming. For the service provisioning in cell zooming enabled MEC networks, prior works have optimized the infrastructure cost with the constrained probability of QoS violation [25]. The power state of SBSs has also been investigated for energy consumption minimization while guaranteeing the delay constraint of users [26]. However, they have considered only the active and sleep states of SBSs with fixed coverage.

Formulating the delay of multi-component services [27], [28], [29] with online tasks is essential for designing QoS-guaranteed service placement mechanisms. Considering only processing and/or propagation delay for services [30], [31], [32], [33], [34], [35], [36], [37] is not practically enough as tasks of a service arrive randomly, leading to non-negligible queuing delay. The M/M/1 [38][39][40] and M/G/1 [41][42] queuing systems play important roles in the queuing delay formulation for tasks that arrive according to Poisson distribution and be served with different processes [43]. Besides, the delay of a multi-component service should take the service architecture, component dependency [44], [45], [46], [27], and allocated resources into account [14]. The resource sharing mechanism is also not trivial. Traditional resource sharing mechanism is considered in [14], where all services that are placed on the same infrastructure, i.e., edge servers or wireless links, are put into a common queue. Therefore, there exists interference [29] among services in queuing delay. However, the new resource slicing mechanism based on network virtualization [47], [48] allows reserving different resources for different services. Tasks of each service are queued separately. Thus, the interference among services is avoided.

The equivalent bandwidth for a wireless link is initially defined as the maximum available arrival rate of data packets that the link can support under the constraint of delay satisfaction rate [49]. The equivalent bandwidth of a two-hop wireless path has also been investigated [50]. In this paper, we define the equivalent bandwidth for an edge in a service to facilitate the edge allocation on a multi-hop physical path and ensure the delay constraint of the edge is guaranteed.

There are mainly three-category approaches for multi-component service placement, i.e., two-stage approach, one-stage approach, and meta-heuristic algorithm. The two-stage service placement first assigns all the components in a service and then finds routing paths for all edges. The components can be placed according to different rules. For example, assigning components to edge servers with higher general resource capacity in priority [51] results in better load balancing of

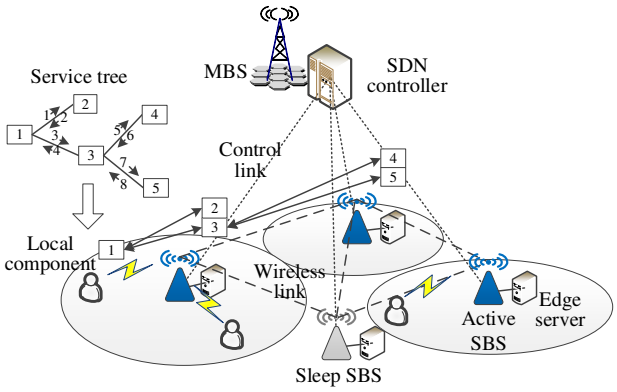


Fig. 1: Network architecture

edge servers. While leveraging Page rank for component assignment [52] takes the connectivity of a service graph into account, which benefits the edge routing in the second stage. Unlike two-stage approaches, the one-stage service placement assigns components and edges simultaneously. To be specific, a component is assigned together with all the edges that are linked to the component [53]. Last but not least, the meta-heuristic algorithms such as Ant Colony Optimization (ACO) [14] searches the near-optimal service placement results in an iterative exploration and exploitation way. To the best of our knowledge, the multi-component service placement in the cell zooming enabled MEC networks for energy efficiency optimization has not been addressed in the literature. The potential of utilizing the equivalent bandwidth for flexible resource allocation and resource fragment reduction is also neglected.

III. System Models and Problem Formulation

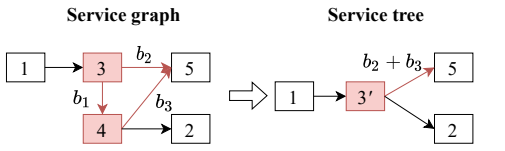
A. MEC networks

The MEC networks contain multiple MBSs and SBSs that follow homogeneous Poisson Point Process (PPP) distribution [40] as shown in Fig. 1. Each MBS acts as the controller responsible for transmitting control signals for resource allocation and management of SBSs located within its coverage. SBSs are equipped with multiple transceivers to communicate with each other via wireless channels. The data transmission between MBS and SBSs uses the traditional microwave communication in 4G/Long Term Evolution (LTE). SBSs use the millimeter wave to communicate with users and other SBSs to achieve a high data rate. Each SBS is connected to a co-located edge server with negligible communication delay. We use edge nodes to represent both SBSs and edge servers for ease of presentation when it is clear from the context.

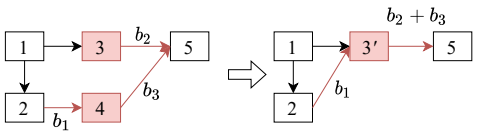
Table I shows the notations used in this paper. Let $\mathcal{N}$ and $\mathcal{L}$ denote the set of edge nodes and wireless links, respectively. The node set $\mathcal{N} = \mathcal{N}^M \cup \mathcal{N}^S \cup \mathcal{N}^U$ consists of MBSs in $\mathcal{N}^M$, SBSs in $\mathcal{N}^S$, and user devices in $\mathcal{N}^U$. The computation capacity of the node indexed by $n \in \mathcal{N}^S$ is represented by C_n CPU cycles/s. Similarly, denote $\mathcal{L} = \mathcal{L}^M \cup \mathcal{L}^S \cup \mathcal{L}^U$ the set of links, where $\mathcal{L}^M$, $\mathcal{L}^S$, and $\mathcal{L}^U$ indicate the sets of links between MSBs and SBSs, links among SBSs, and links from SBSs to users respectively. The data rate of link $l \in \mathcal{L}$ is given by R_l.

TABLE I: Notations

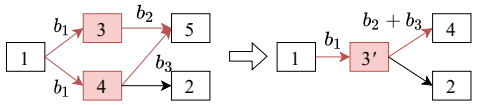
Symbol(s)	Description
$\mathcal{N} = \mathcal{N}^M \cup \mathcal{N}^S \cup \mathcal{N}^U$	Set of edge nodes where $\mathcal{N}^M$, $\mathcal{N}^S$, and $\mathcal{N}^U$ denote the sets of MBSs, SBSs, and users respectively
$\mathcal{L} = \mathcal{L}^M \cup \mathcal{L}^S \cup \mathcal{L}^U$	Set of wireless links where $\mathcal{L}^M$, $\mathcal{L}^S$, and $\mathcal{L}^U$ denote the sets of links between MBSs and SBSs, links among SBSs, and links from SBSs to users respectively
C_n	The computation capacity of node $n \in \mathcal{N}^S$
H	The bandwidth of wireless links $l \in \mathcal{L}^S \cup \mathcal{L}^U$
N_0	The power density of white noise
$\mathcal{G}_l$	The channel gain of wireless link $l \in \mathcal{L}^S \cup \mathcal{L}^U$
R_l	The data rate of wireless link $l \in \mathcal{L}$
α	The constant path loss factor of wireless links
β	The path loss multiplier of wireless links
ϕ_l	Transmitting distance of wireless link $l \in \mathcal{L}$
$p^T_{\max}$	The maximum transmitting power of SBSs
p^A	The static power of an active SBS
p^T_l	The transmitting power of wireless link $l \in \mathcal{L}$
p^C	The power for processing per CPU cycle
p^S	The power of a sleep SBS
ρ_l	The load factor of wireless link l, $\rho_l \in [0, 1]$
ρ_n	The load factor of edge server n, $\rho_n \in [0, 1]$
I_n	Set of links that flow into node n
O_n	Set of links that flow out of node n
F	The average packet size
$\overline{X^2}$	The expected second moment of service time of packets
Q	A very big value
f_l	The residual bandwidth capacity of wireless link l
g_n	The residual computation resource of edge server n
$\mathcal{K}$	Set of services
$\mathcal{K}_n$	Set of services of user $n \in \mathcal{N}^U$
$G_k = \{\mathcal{V}_k, \mathcal{E}_k\}$	The task graph of the service indexed by k, where $\mathcal{V}_k$ and $\mathcal{E}_k$ are sets of components and edges respectively
u_k	The user that requests G_k
η_k	The arrival rate of G_k
d_k	The delay request of G_k
T_k	The duration of G_k
$c_{k,i}$	The computation resource request of component $i \in \mathcal{V}_k$
$b_{k,j}$	The data size to be transmitted on edge $j \in \mathcal{E}_k$
$\chi_{k,m}$	The mth branch of G_k
$w_{k,i}$	The expected queuing time of computation units on component $i \in \mathcal{V}_k$
$w_{k,j,l}$	The expected queuing time of data packets on $j \in \mathcal{E}_k$
$\tau^V_{k,i}$	The delay of component $i \in \mathcal{V}_k$
$\tau^E_{k,j}$	The delay of edge $j \in \mathcal{E}_k$
τ^B_m	The delay of branch $m \in G_k$
j_S	The source component of edge $j \in \mathcal{E}_k$
j_D	The destination component of edge $j \in \mathcal{E}_k$
$\psi_{k,z}$	The weighted height of $z \in \mathcal{V}_k$ or $z \in \mathcal{E}_k$
$\varsigma_{k,m}$	The weighted length of branch $m \in G_k$
$Y_{k,j}$	The placed path of edge $j \in \mathcal{E}_k$
$\lvert Y_{k,j} \rvert$	The path length of $Y_{k,j}$
$\hat{d}_{k,z}$	The assigned delay constraint of $z \in \mathcal{V}_k$ or $z \in \mathcal{E}_k$
$\hat{v}_{k,i}$	The resource request of component $i \in \mathcal{V}_k$ based on the delay constraint $d_{k,i}$
$\hat{\zeta}_{k,j}$	The resource request of edge $j \in \mathcal{E}_k$ based on the delay constraint $d_{k,j}$
$a_{k,i,n}$	Binary variable, taking 1 if component $i \in \mathcal{V}_k$ is placed to edge server $n \in \mathcal{N}^S$ and 0 otherwise
$\varphi_{k,j,l}$	Binary variable, taking 1 if edge $j \in \mathcal{E}_k$ is placed to a path that contains the wireless link $l \in \mathcal{L}^S \cup \mathcal{L}^U$ and 0 otherwise
$v_{k,i,n}$	The amount of computation resource that edge server $n \in \mathcal{N}^S$ allocates to component $i \in \mathcal{V}_k$
$\zeta_{k,j,l}$	The amount of communication resource that wireless link $l \in \mathcal{L}^S \cup \mathcal{L}^U$ allocates to edge $j \in \mathcal{E}_k$
σ_k	Binary variable, taking 1 if service $k \in \mathcal{K}$ is placed successfully and 0 otherwise
δ_n	Binary variable, taking 1 if SBS $n \in \mathcal{N}^S$ is in active state, and 0 for sleep state



(a) The predecessor components of component 5 are dependent



(b) The predecessors of component 5 have different predecessors



(c) The predecessors of component 5 have the same predecessor

Fig. 2: Illustration of transforming a service graph to a service tree

B. Multi-component services

1) Service graph: The set of services is denoted by $\mathcal{K}$. Any service $k \in \mathcal{K}$ can be described using a call graph $G_k = \{\mathcal{V}_k, \mathcal{E}_k\}$, where $\mathcal{V}_k$ and $\mathcal{E}_k$ are the set of components and edges, respectively. Each service G_k contains multiple homogeneous tasks with the arrival rate of η_k in a duration of T_k. Each edge in a service specifies the dependency between its start and end components. We assume the first and last components (which are the same component) in any service, e.g., data pre-processing and results demonstration, should be executed locally in a user device $u_k \in \mathcal{N}^U$. The computation resource request of the ith component in $\mathcal{V}_k$ is $c_{k,i}$ in the unit of CPU cycle, and the data size to be transmitted on the edge indexed by j is denoted by $b_{k,j}$ bits. Note that the data size for the upstream and downstream transmissions of a bidirectional edge in a service can be asymmetric.

2) Service tree: Any call graph of a service can be organized using a service tree [54]. We can transform a service graph into a service tree by merging all the predecessor components of any component. The resource requests of the merged component and the related edges are determined by adding up all the resource requests of the predecessor components/input edges. For example in Fig. 2 where the original service graph contains 5 components indexed by $\{1, \cdots, 5\}$, the component 5 has two predecessor components, i.e., 3 and 4, which are expected to be merged as $3'$. The directed edges indicate the dependency between components, and the transmitted data size is marked beside the corresponding edge with b_1, b_2, and b_3. Note that the processing results of components should be returned according to the reverse direction of the edges. There are three cases considering the relationships among the predecessor components, resulting in different merging results:

a) When the predecessor components of the target component are dependent, as shown in Fig. 2 (a) where component 4 requires the output of component 3, merging the predecessor components saves the resource of the interconnected edges between the predecessors, i.e., b_1 for the edge $< 3 - 4 >$.

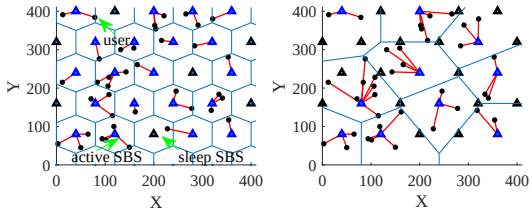


Fig. 3: SBS states and Voronoi diagram. Left: original distribution of SBSs; Right: Voronio diagram of SBSs considering cell zooming and SBS sleeping

b) When the predecessors of the target component have different predecessors, as shown in Fig. 2 (b) where components 3 and 4 depend respectively on components 1 and 2, merging the predecessor components gives the same total resource request as before.

c) When the predecessors of the target component have the same predecessor, as shown in Fig. 2 (c) where both components 3 and 4 depend on component 1, the transmitted data size from component 1 to the merged component $3'$ equals the output of component 1. Thus, the communication resource for transmitting multiple copies of the output of component 1 (to components 3 and 4) is avoided.

Overall, the benefits of converting service graphs to service trees are twofold. First, we can save the communication resources for placing services by avoiding redundant data transmission. More importantly, organizing services with trees allows for efficient service placement algorithms as presented in Sections IV and V.

Therefore, we can model any service $k \in \mathcal{K}$ as $G_k = \{\chi_{k,m}, \forall m\}$ where $\chi_{k,m}$ indicates the mth branch in G_k. Branches of a service G_k have the following properties,

a) The root of the tree corresponds to the input and the output of a service, i.e., the first and last components, which should be processed locally;

b) Each branch in G_k contains a set of continuous components and edges, constructing a path from the root of G_k to one of its leaf components. All components in a branch should be executed in sequence;

c) The number of branches of a service equals the number of leaf components in the service tree;

d) Different components on different branches can be executed in parallel.

3) The delay of services: The delay request of G_k is d_k, constraining the total time for completing all components in the service. Taking the service tree (for the kth service without loss of generality) in Fig. 1 for example where the number and arrow beside an edge between two components indicate the index and two directions of the edge. Let $\tau^V_{k,i}$, $\tau^E_{k,j}$, and $\tau^B_{k,m}$ denote the delay of component i, edge j, and branch m respectively. There are three branches in the service, that is, $\chi_{k,1} = \{v_1 \leftrightarrow v_2\}$, $\chi_{k,2} = \{v_1 \leftrightarrow v_3 \leftrightarrow v_4\}$ and $\chi_{k,3} = \{v_1 \leftrightarrow v_3 \leftrightarrow v_5\}$ with respective delay of $\tau^B_{k,1} = \tau^V_{k,1} + \tau^E_{k,1} + \tau^E_{k,2} + \tau^V_{k,2}$, $\tau^B_{k,2} = \tau^V_{k,1} + \tau^E_{k,3} + \tau^E_{k,4} + \tau^V_{k,3} + \tau^E_{k,5} + \tau^E_{k,6} + \tau^V_{k,4}$ and $\tau^B_{k,3} = \tau^V_{k,1} + \tau^E_{k,3} + \tau^E_{k,4} + \tau^V_{k,3} + \tau^E_{k,7} + \tau^E_{k,8} + \tau^V_{k,5}$. Thus, the delay of a service G_k is equivalent to the largest accumulated delay of branches on G_k, i.e., $\max\left\{\tau^B_{k,1}, \tau^B_{k,2}, \tau^B_{k,3}\right\}$. The delay constraint of the service G_k is $\max\left\{\tau^B_{k,1}, \tau^B_{k,2}, \tau^B_{k,3}\right\} \leq d_k$.

C. Power models

The MBSs are always powered on for controlling SBSs. Each SBS has two power states: active and sleep. An active SBS is capable of transmitting/receiving data to/from other SBSs and users within its coverage. According to Shannon's formula, the transmitting power of a wireless link indexed by l is

$$p_l^T = \frac{HN_0}{\mathcal{G}_l \cdot \left(2^{R_l/H} - 1\right)}, \tag{1}$$

where H denotes the spectrum width, N_0 indicates the power density of white noise, and $\mathcal{G}_l$ is the channel gain, which is related with path-loss factor α, path-loss exponent β and the link transmission distance ϕ_l as shown in Eq. (2).

$$\mathcal{G}_l = \alpha + \beta \log \phi_l. \tag{2}$$

The power of an active SBS $n \in \mathcal{N}^S$ is composed of the static power p^A and the load dependent transmitting power of its transceivers. When a SBS $n \in \mathcal{N}^S$ is zero loaded, it can be switched into sleep state with the power of p^S. The power amplifier, radio frequency transceivers, and most of hardwares will be turned off for a sleep SBS to reduce power consumption. Let δ_n be the power state of SBS n, i.e., there are $\delta_n = 1$ for an active SBS and $\delta_n = 0$ for a sleep SBS, the total power of a SBS is

$$p_n = \delta_n \cdot \left(p^A + \sum_{l \in \mathcal{L}^U \cap O_n} p_l^T + \sum_{l \in \mathcal{L}^S \cap O_n} \left(\rho_l p_{\max}^T\right) + \rho_n C_n p^C \right) + (1 - \delta_n)\, p^S, \tag{3}$$

where O_n indicates the set of links that flow out of SBS n, ρ_n and p^C are the load factor of node n and the power consumption per CPU cycle, respectively. The maximum transmitting power of a transmitter on a SBS is represented by $p_{\max}^T$ and ρ_l is the load factor of link l.

Fig. 3 demonstrates the power states and Voronoi diagram of SBSs. The blue and black triangles indicate the active and sleep SBSs, respectively, and the small black squares represent users. On the left-hand side of Fig. 3, each SBS has a hexagon-shaped coverage, guaranteeing the full coverage of the whole area. Each user gets access to the nearest SBS by default, shown in red lines. SBSs that do not serve any user are switched into sleep state. However, the Voronoi diagram can be optimized not only by adjusting the transmitting power of SBSs, i.e., cell zooming, but also by controlling the power states of SBSs, as shown in the right-hand side of Fig. 3. More SBSs are in sleep state by gathering users to several SBSs. In this process, we can guarantee the delay requests of users by using the appropriate transmitting power of SBSs. As a result, we can save more energy.

D. Delay models

The processing units of computation and communication resources are a CPU cycle and a data packet, respectively. Computation (communication) units of each service on an (a) edge server (wireless link) are put into a separate queue and be executed according to First-In-First-Out (FIFO) as shown in Fig. 4 such that there is no interference among services.

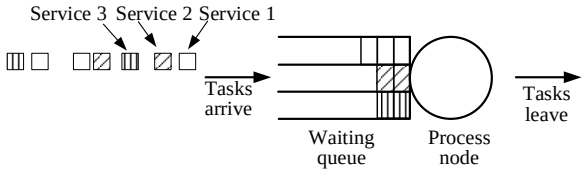


Fig. 4: Queue model of edge servers and wireless links

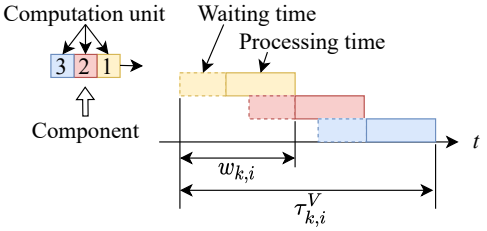


Fig. 5: Delay of a component

We decouple the placement of components and edges into two sets of variables for binary assignment indicators and continuous resource allocation, respectively. Specifically, let $a_{k,i,n}$ be the binary variable of component assignment, indicating if component i in service k is placed onto the edge node n or not. Similarly, $\varphi_{k,j,l}$ is the binary variable for edge assignment, taking 1 if edge j in service k traverses wireless link l and 0 otherwise. Moreover, denote $\upsilon_{k,i,n}$ the amount of computation resources that node n allocates to the component i in service k. $\zeta_{k,j,l}$ represents the amount the bandwidth that link l allocates to the edge j in service k. Let $\boldsymbol{a}$, $\boldsymbol{\varphi}$, $\boldsymbol{\upsilon}$ and $\boldsymbol{\zeta}$ be the set of $a_{k,i,n}$, $\varphi_{k,j,l}$, $\upsilon_{k,i,n}$ and $\zeta_{k,j,l}$ respectively.

1) Delay of service components: According to the Little's formula for M/M/1 queuing system [40], for component i in service k with allocated resource $\upsilon_{k,i,n}$ from edge server n, the expected queuing delay of computation units on component i in G_k is

$$w_{k,i} = \frac{1}{\upsilon_{k,i,n} - c_{k,i}\eta_k}, \exists a_{k,i,n} = 1. \tag{4}$$

The delay of a component i in service k includes the expected queuing delay and the processing delay, i.e.,

$$\tau_{k,i}^V = \sum_{n \in \mathcal{N}^S \cup \mathcal{N}^U} a_{k,i,n} \left(w_{k,i} + \frac{c_{k,i}}{\upsilon_{k,i,n}} \right). \tag{5}$$

2) Delay of service edges: For service edges, the packet arrival process follows Poisson distribution and the packet processing follows any distribution. According to Pollaczek-Khinchin (P-K) formula of M/G/1 queuing system [55], the expected queuing delay of data packets on edge j in service k with allocated bandwidth $\zeta_{k,j,l}$ from link $l \in \mathcal{L}^S \cup \mathcal{L}^U$ is

$$w_{k,j,l} = \frac{b_{k,j}\eta_k \overline{X^2}/F}{2\left(1 - b_{k,j}\eta_k/\zeta_{k,j,l}\right)}, \exists \varphi_{k,j,l} = 1, \tag{6}$$

where F is the average packet size and $\overline{X^2}$ denotes the second moment of service time for data packets.

The delay of any edge j in G_k includes the expected packet queuing delay and the data transmitting delay of all links on the placed path of the edge, that is,

$$\tau_{k,j}^E = \sum_{l \in \mathcal{L}^S \cup \mathcal{L}^U} \varphi_{k,j,l} \left(w_{k,j,l} + \frac{b_{k,j}}{\zeta_{k,j,l}} \right). \tag{7}$$

It should be noted that $w_{k,i}$ represents the expected queuing delay of computation units whose arrival process follows the Poisson distribution. However, a service component contains multiple computation units to process. The delay of a component $\tau^V_{k,i}$ is the duration from the time when the first computation unit arrives to the time when the last computation unit leaves. Considering that all the computation units arrive almost at the same time, the waiting time and processing time of computation units overlap with each other, as shown in Fig. 5. Thus, the component delay is the sum of the expected waiting time of a computation unit and the total processing time of all the computation units of the component, which is given by

$$\tilde{\tau}^V_{k,i} = \sum_{n\in\mathcal{N}^S\cup\mathcal{N}^U} a_{k,i,n}\left(w_{k,i} - \frac{1}{v_{k,i,n}} + \frac{c_{k,i}}{v_{k,i,n}}\right). \quad (8)$$

We omit the middle term in the above parentheses, i.e., the processing time of one computation unit, as each component generally contains several Megacyles (i.e., $c_{k,i} >> 1$). Therefore, we obtain (5). Similarly, the data transmitted on a service edge contains multiple data packets. The total delay of a service edge is formulated as (7).

E. Problem formulation

The objective of service placement is to minimize the total power consumption while accepting as many services as possible defined as

$$\begin{aligned} P(\boldsymbol{a},\boldsymbol{v},\boldsymbol{\varphi},\boldsymbol{\zeta}) &= \sum_{n\in\mathcal{N}^S} p_n - Q\sum_{k\in\mathcal{K}}\sigma_k \\ &= \sum_{n\in\mathcal{N}^S}\Bigg[\delta_n\cdot\Bigg(p^A + \sum_{l\in\mathcal{L}^U\cap O_n} p^T_l \\ &\quad + \sum_{l\in\mathcal{L}^S\cap O_n} \frac{\sum_{k\in\mathcal{K},j\in\mathcal{E}_k}\varphi_{k,j,l}\zeta_{k,j,l}}{R_l}\cdot p^T_{\max} \\ &\quad + \sum_{k\in\mathcal{K},i\in\mathcal{V}_k} a_{k,i,n}v_{k,i,n}\cdot p^C\Bigg) + (1-\delta_n)\,p^S\Bigg] \\ &\quad - \sum_{k\in\mathcal{K}} Q\sigma_k. \end{aligned} \quad (9)$$

where Q is a big value to facilitate accepting more service requests, in case no service is accepted gives the minimum power consumption.

The problem of service placement is formulated as $P1$.

$P1$:

$$\min\ P(\boldsymbol{a},\boldsymbol{v},\boldsymbol{\varphi},\boldsymbol{\zeta}). \quad (10)$$

$s.t.$

$$a_{k,i,u_k} = \sigma_k, \forall k\in\mathcal{K}_n, i\in\mathcal{V}_k, i=0, \quad (11)$$

$$\sum_{n\in\mathcal{N}^S} a_{k,i,n} = \sigma_k, \forall k\in\mathcal{K}, i\in\mathcal{V}_k, i\neq 0, \quad (12)$$

$$\begin{aligned} &\sum_{l\in I_n}\varphi_{k,j,l} - \sum_{l\in O_n}\varphi_{k,j,l} = -a_{k,j_S,n} + a_{k,j_D,n}, \\ &\forall k\in\mathcal{K}, j\in\mathcal{E}_k, n\in\mathcal{N}^S\cup\mathcal{N}^U, \end{aligned} \quad (13)$$

$$\sum_{l\in I_n}\varphi_{k,j,l} \le \sigma_k, \forall k\in\mathcal{K}, j\in\mathcal{E}_k, n\in\mathcal{N}^S\cup\mathcal{N}^U, \quad (14)$$

$$\sum_{l\in O_n}\varphi_{k,j,l} \le \sigma_k, \forall k\in\mathcal{K}, j\in\mathcal{E}_k, n\in\mathcal{N}^S\cup\mathcal{N}^U, \quad (15)$$

$$\varphi_{k,j,l} = \varphi_{k,j',l}, \text{if } j_S = j'_D \text{ and } j_D = j'_D, \quad (16)$$

$$C_n - \sum_{k\in\mathcal{K},i\in\mathcal{V}_k} v_{k,i,n} \ge 0, \forall n\in\mathcal{N}^S, \quad (17)$$

$$R_l - \sum_{k\in\mathcal{K},j\in\mathcal{E}_k}\zeta_{k,j,l} \ge 0, \forall l\in\mathcal{L}^S, \quad (18)$$

$$\delta_n \ge a_{k,i,n}, \forall k\in\mathcal{K}, i\in\mathcal{V}_k, n\in\mathcal{N}^S, \quad (19)$$

$$\delta_n \ge \varphi_{k,j,l}, \forall k\in\mathcal{K}, j\in\mathcal{E}_k, n\in\mathcal{N}^S, l\in O_n, \quad (20)$$

$$\begin{aligned} p^T_l \ge \sum_{k\in\mathcal{K},j\in\mathcal{E}_k}\varphi_{k,j,l}\sum_{k\in\mathcal{K},j\in\mathcal{E}_k}\frac{HN_0}{(\alpha+\beta\log\phi_l)\left(2^{\zeta_{k,j,l}/H}-1\right)}, \\ \forall l\in\mathcal{L}^U, \end{aligned} \quad (21)$$

$$0 \le \sum_{l\in\mathcal{L}^U\cap O_n} p^T_l \le p^T_{\max}, \forall n\in\mathcal{N}^S, \quad (22)$$

$$\max\left\{\sum_{i,j\in\chi_{k,m}}\left(\tau^V_{k,i}+\tau^E_{k,j}\right), \forall\chi_{k,m}\in G_k\right\} \le d_k, \forall k\in\mathcal{K}, \quad (23)$$

$$a_{k,i,n} = \{0,1\}, \forall k\in\mathcal{K}, i\in\mathcal{V}_k, n\in\mathcal{N}^S\cup\mathcal{N}^U, \quad (24)$$

$$\varphi_{k,j,l} = \{0,1\}, \forall k\in\mathcal{K}, j\in\mathcal{E}_k, l\in\mathcal{L}^S\cup\mathcal{L}^U, \quad (25)$$

$$\sigma_k = \{0,1\}, \forall k\in\mathcal{K}, \quad (26)$$

$$\delta_n = \{0,1\}, \forall n\in\mathcal{N}^S, \quad (27)$$

$$v_{k,i,n} \ge 0, \forall k\in\mathcal{K}, i\in\mathcal{V}_k, n\in\mathcal{N}^S\cup\mathcal{N}^U, \quad (28)$$

$$\zeta_{k,j,l} \ge 0, \forall k\in\mathcal{K}, j\in\mathcal{E}_k, l\in\mathcal{L}^S\cup\mathcal{L}^U. \quad (29)$$

Solving the problem $P1$ is equivalent to finding the placement results of components (i.e., variables $\boldsymbol{a}$ and $\boldsymbol{v}$) and edges (i.e., variables $\boldsymbol{\varphi}$ and $\boldsymbol{\zeta}$) with the minimum power consumption and the maximum placed services. Processing the first (also the last) component locally is specified in (11) where $\mathcal{K}_n$ denotes the set of local services of edge node $n\in\mathcal{N}^U$. Constraint (12) specifies that every component in a service should be assigned to one and only one edge node. The flow conservation in (13) aims to assign each edge in a service to a continuous and loop-free physical path between the placed edge nodes of its start and end components. Constraints (14) and (15) specify that the edge assignment is not splittable, namely one edge to one physical path. Constraint (16) indicates that each bidirectional edge should be placed to the same path in different directions. The computation and bandwidth resource capacity constraints of edge server and wireless links among SBSs are formulated in (17) and (18) respectively. Any SBS that carries services should be in active state as described in (19) and (20). The transmitting power of wireless link l from a SBS to a user is formulated in (21) and the maximum transmitting power constraint is specified in (22). The delay constraints of services are shown in (23), restricting the delay of each branch to be lower than d_k. All placement variables are constrained to be binary as in (24) and (25). The service placement indicators $\sigma_k, \forall k\in\mathcal{K}$ and SBS power state variables $\delta_n, \forall n\in\mathcal{N}^S$ are also binary as in (26) and (27). The resource allocation variables are positive as in (28) and (29).

Based on the problem $P1$, there are possibilities to reduce the total power consumption by adjusting the transmitting

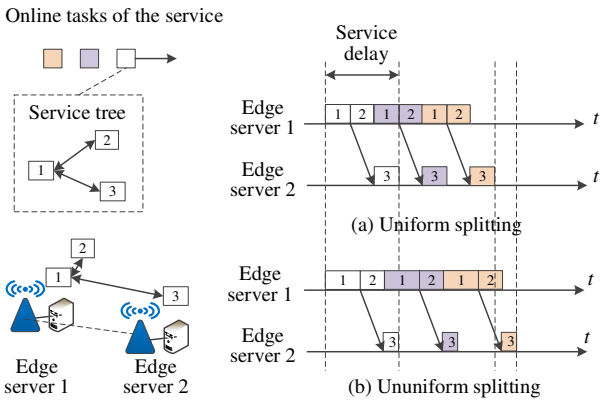


Fig. 6: Parallel task processing for different delay splitting mechanisms

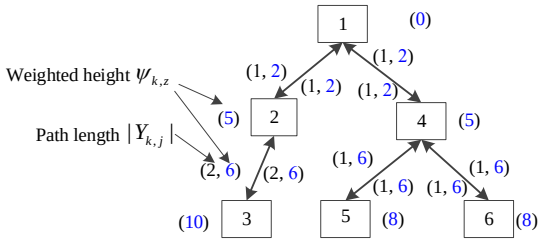


Fig. 7: Weighted heights of components and edges in a service

power of wireless links from SBSs to users in (21) through optimizing the placement of services. We can save energy by allowing as many as SBSs to switch into sleep state.

IV. Energy-Efficient Service Placement Based on Cell Zooming

The problem $P1$ is a Mixed Integer Non-Linear Programming (MINLP) due to the power and delay constraints (21) and (23), which is hard to be solved within exponential time. Thus, we transform the problem $P1$ to be a Mixed Integer Linear Programming (MILP) through delay splitting. Lots of existing methods, e.g., branch and bound can be applied to a MILP to find the optimal solution. To be specific, we first decouple the constraint (23) through delay splitting and then reformulate the problem $P1$ with computed resource requests $\hat{v}_{k,i}$ and $\hat{\zeta}_{k,j}, \forall k \in \mathcal{K}, i \in \mathcal{V}_k, j \in \mathcal{E}_k$ that satisfy the delay constraints.

A. Delay splitting

1) Uniform delay splitting: Multiple online tasks can be executed in parallel for each service to improve resource efficiency and reduce delay. Namely, components in different tasks of a service can be processed simultaneously as different components are allocated with separated resource slices. It is plausible to improve resource utilization by maximizing the probability of task parallelization. It happens when the delay of different components and edges are the same, i.e., uniform delay splitting, as shown in Fig. 6. The service has three components connected with two edges, and there are three online tasks for the service. Note that all tasks of the service have the same request architecture as the service tree. Assume that components 1 and 2 are placed to edge server 1, and component 3 is assigned to edge server 2. In Fig. 6 (a), all the components have the same length in terms of time, that is, uniform delay splitting. While different components have different delay in Fig. 6 (b). Online tasks that are assigned to different edge servers can be processed in parallel, e.g., component 3 of the first (white) task and component 1 of the second (purple) task are processed simultaneously on edge servers 2 and 1, respectively. It can be obtained that even though different delay splitting mechanisms have the same service delay, the uniform splitting allows a higher degree of parallelization among tasks. Thus, higher resource utilization can be achieved.

2) Weighted heights of components and edges: The total delay constraint is averagely distributed among all elements, i.e., components and edges, on a branch for uniform delay splitting. The allocated delay of an edge is weighted by the placed path length of the edge. We define a weighted height $\psi_{k,z}$ for each component and edge z of a service to facilitate uniform delay splitting. For example in Fig. 7, the service has six components (i.e., the squares). The first number in the parentheses beside each edge indicates the path length $|Y_{k,j}|$ that the edge $j \in \mathcal{E}_k$ is placed. The weighted height of any component or edge is shown in blue. The height of the root component is 0 by default. Bi-directional edges share the same weighted height, which equals the weighted height of its father component plus 1. The weighted height of a component considers the placed path length of two directional edges between the component and its father. Thus, the weighted height of component 2 is 1+2*2=5. For component 3, its weighted height is 6+2*2=10.

Algorithm 1 shows the computation of weighted heights of components and edges in a service. The procedure Assign_Height() aims to assign the element z, i.e., either a component or an edge, with weighted height h. It is called recursively according of depth-first traversal of the service tree (lines 10 and 16). The weighted heights of bidirectional edges are same (line 14). The **Algorithm 1** can also be used for the weighted height computation given partial placement decision $\boldsymbol{\alpha}$ and $\boldsymbol{\varphi}$ to facilitate delay re-spitting. When the resource allocation of some components and edges has been completed, i.e., $v_{k,i,n} > 0, \exists a_{k,i,n} = 1$ and $\zeta_{k,j,l} > 0, \exists \varphi_{k,j,l} = 1$ for some $i \in \mathcal{V}_k, j \in \mathcal{E}_k$, the actual delay of these components and edges is subtracted from the total delay. The residual delay is split among all elements whose resource allocation has not been completed, as described in **Algorithm 4**. Thus, only elements whose resource allocation have not completed are considered for the weighted height computation (lines 5 and 12).

3) Algorithm description: The total delay of a service is split layer by layer using the weighted heights of components and edges. Define the weighted length of branch $\chi_{k,m}$, denoted by $\varsigma_{k,m}$, as the maximum weighted height of components along the branch. Let $\xi_{k,m}$ be the residual delay of branch $\chi_{k,m}$. The assigned delay constraint of component i can be formulated as

$$d_{k,z} \leq \xi_k / \left(\max\left\{\varsigma_{k,m}, \forall \chi_{k,m}, \exists z \in \chi_{k,m}\right\} - \psi_{k,z}\right), \forall z \in \mathcal{V}_k. \quad (30)$$

Algorithm 1: Weighted_Height_Computing

Input: $G_k, d_k, Y_{k,i}, a_{k,i,n}, \varphi_{k,j,l}, v_{k,i,n}, \zeta_{k,j,l}, \forall i \in \mathcal{V}_k, j \in \mathcal{E}_k, n \in \mathcal{N}^S \cup \mathcal{N}^U, l \in \mathcal{L}^S \cup \mathcal{L}^U$
Output: $\psi_{k,z}, \forall z \in \mathcal{V}_k \cup \mathcal{E}_k$

1 Initialize $z \leftarrow 0, for\ z \in \mathcal{V}_k, h \leftarrow 0$;
2 Assign_Height(z, h).
3 **Assign_Height**(z, h):
4 **if** $z \in \mathcal{V}_k$ **then**
5 **if** $v_{k,z,n} \leq 0, \exists a_{k,z,n} = 1$ *or* $a_{k,z,n} = 0, \forall n \in \mathcal{N}^S$ **then**
6 $\psi_{k,z} \leftarrow h$;
7 $h \leftarrow h + 1$;
8 **if** *z is not a leaf component* **then**
9 **for** *$j \in \mathcal{E}_k, j_S = z$ and j_D is a son of j_S* **do**
10 Assign_Height(j, h);
11 **else**
12 **if** $\zeta_{k,z,l} \leq 0, \exists \varphi_{k,z,l} = 1\ \varphi_{k,z,l} = 0, \forall l \in \mathcal{L}^S \cup \mathcal{L}^U$ **then**
13 $\psi_{k,z} \leftarrow h$;
14 $\psi_{k,z'} \leftarrow h, \exists z' \in \mathcal{E}_k, z'_S = z_D$ *and* $z'_D = z_S$;
15 $h \leftarrow h + 2 * |Y_{k,j}| + 1$;
16 Assign_Height(z_D, h).

For a service edge, the allocated delay is weighted by its placed path length:

$$d_{k,z} \leq |Y_{k,z}| \cdot \xi_k / \left(\max\left\{\varsigma_{k,m}, \forall \chi_{k,m}, \exists z \in \chi_{k,m}\right\} - \psi_{k,z}\right), \quad \forall z \in \mathcal{E}_k, \tag{31}$$

By allocating delay for each element with the minimum average residual delay of all belonged branches, the total delay of all branches is not higher than d_k. We achieve the delay splitting in an iterative way as in **Algorithm 2**. Initially, we have $a_{k,i,n} = 0, \varphi_{k,j,l} = 0, v_{k,i,n} = -1, \zeta_{k,j,l} = -1, \forall i \in \mathcal{V}_k, j \in \mathcal{E}_k, n \in \mathcal{N}^S \cup \mathcal{N}^U, l \in \mathcal{L}^S \cup \mathcal{L}^U$ for a service k. However, **Algorithm 2** is also used for the delay re-splitting when the resource allocation of a service is partially completed, as described in **Algorithm 4**. The delay of a component or an edge whose resource allocation has been completed is determined according to the actual delay based on the allocated resources (lines 9, 15, and 16).

Algorithm 2: Delay_Splitting

Input: $G_k, d_k, \psi_{k,z}, \forall z \in \mathcal{V}_k \cup \mathcal{E}_k, a_{k,i,n}, \varphi_{k,j,l}, v_{k,i,n}, \zeta_{k,j,l}, \forall i \in \mathcal{V}_k, j \in \mathcal{E}_k, n \in \mathcal{N}^S \cup \mathcal{N}^U, l \in \mathcal{L}^S \cup \mathcal{L}^U$
Output: $d_{k,z}, \forall z \in \mathcal{V}_k \cup \mathcal{E}_k$

1 Initialize $\xi_k \leftarrow d_k, z \leftarrow 0,\ for\ z \in \mathcal{V}_k$;
2 Assign_Delay(ξ_k, z).
3 **Assign_Delay**(ξ_k, z):
4 **if** $z \in \mathcal{V}_k$ **then**
5 **if** $z = 0$ **then**
6 $d_{k,z} \leftarrow$ the actual delay of local component;
7 **else**
8 **if** $v_{k,z,n} > 0, \exists a_{k,z,n} = 1$ **then**
9 $d_{k,z} \leftarrow$ the actual delay of the component z by (5);
10 **else**
11 $d_{k,z} \leftarrow \xi_k / \left(\max\left\{\varsigma_{k,m}, \forall \chi_{k,m}, \exists z \in \chi_{k,m}\right\} - \psi_{k,z}\right)$;
12 $\xi_k \leftarrow \xi_k - d_{k,z}$;
13 **else**
14 **if** $\zeta_{k,z,l} > 0, \forall \varphi_{k,z,l} = 1$ **then**
15 $d_{k,z} \leftarrow$ the actual delay of the edge z by (7);
16 $d_{k,z'} \leftarrow$ the actual delay of the edge $z', \exists z' \in \mathcal{E}_k, z'_D = z_D$ *and* $z'_D = z_S$ by (7);
17 **else**
18 $d_{k,z} \leftarrow |Y_{k,z}| \cdot \xi_k / \left(\max\left\{\varsigma_{k,m}, \forall \chi_{k,m}, \exists z \in \chi_{k,m}\right\} - \psi_{k,z}\right)$;
19 $d_{k,z'} \leftarrow d_{k,z}, \exists z' \in \mathcal{E}_k, z'_D = z_D$ *and* $z'_D = z_S$;
20 $\xi_k \leftarrow \xi_k - d_{k,z} - d_{k,z'}$;
21 **if** $z \in \mathcal{V}_k$ **then**
22 **for** *$z' \in \mathcal{E}_k, z'_S = z$ and z' has not been assigned with a delay constraint* **do**
23 Assign_Delay(ξ_k, z');
24 **else**
25 Assign_Delay(ξ_k, z_D).

B. Problem transformation

1) Transforming delay to resource requests: Based on the split delay, we can transform the delay constraint of services into the computation and bandwidth resource requests for components and service edges, ie., $\hat{v}_{k,i}$ and $\hat{\zeta}_{k,j}$. Since the placed path lengths of service edges are unknown initially, we assume that each edge is placed on a one-hop path. Thus, the delay constraint in (23) can be replaced by

$$d_{k,i} = \frac{1}{\hat{v}_{k,i} - c_{k,i}\eta_k} + \frac{c_{k,i}}{\hat{v}_{k,i}}, \forall k \in \mathcal{K}, i \in \mathcal{V}_k, \tag{32}$$

$$d_{k,j} = \frac{b_{k,j}\eta_k \hat{\zeta}_{k,j} \overline{X^2}}{2F\left(\hat{\zeta}_{k,j} - b_{k,j}\eta_k\right)} + \frac{b_{k,j}}{\hat{\zeta}_{k,j}}, \forall k \in \mathcal{K}, j \in \mathcal{E}_k, \tag{33}$$

Recall that $\hat{v}_{k,i}$ and $\hat{\zeta}_{k,j}$ represent the resource requests of component i and edge j respectively. The quadratic equation with one unknown in (32) can be rearranged as

$$\hat{v}_{k,i}^2 - \left(c_{k,i}\eta_k + \frac{1 + c_{k,i}}{d_{k,i}} + \right)\hat{v}_{k,i} + \frac{c_{k,i}^2\eta_k}{d_{k,i}} = 0, \tag{34}$$

whose solutions are

$$\hat{v}_{k,i} = \frac{c_{k,i}\eta_k + \frac{1+c_{k,i}}{d_{k,i}} - \sqrt{\left(c_{k,i}\eta_k + \frac{1+c_{k,i}}{d_{k,i}}\right)^2 - \frac{4c_{k,i}^2\eta_k}{d_{k,i}}}}{2}, \tag{35}$$

or

$$\hat{v}_{k,i} = \frac{c_{k,i}\eta_k + \frac{1+c_{k,i}}{d_{k,i}} + \sqrt{\left(c_{k,i}\eta_k + \frac{1+c_{k,i}}{d_{k,i}}\right)^2 - \frac{4c_{k,i}^2\eta_k}{d_{k,i}}}}{2}, \tag{36}$$

with $\hat{v}_{k,i} > c_{k,i}\eta_k$. Comparing $\hat{v}_{k,i}$ in (35) to $c_{k,i}\eta_k$ gives

$$0 < 4c_{k,i}\eta_k \frac{1+c_{k,i}}{d_{k,i}} - \frac{4c_{k,i}^2\eta_k}{d_{k,i}} = \frac{4c_{k,i}\eta_k}{d_{k,i}},$$

$$\frac{1+c_{k,i}}{d_{k,i}} - c_{k,i}\eta_k < \sqrt{\left(c_{k,i}\eta_k + \frac{1+c_{k,i}}{d_{k,i}}\right)^2 - \frac{4c_{k,i}^2\eta_k}{d_{k,i}}},$$

$$\frac{1+c_{k,i}}{d_{k,i}} - \sqrt{\left(c_{k,i}\eta_k + \frac{1+c_{k,i}}{d_{k,i}}\right)^2 - \frac{4c_{k,i}^2\eta_k}{d_{k,i}}} < c_{k,i}\eta_k,$$

$$\frac{c_{k,i}\eta_k + \frac{1+c_{k,i}}{d_{k,i}} - \sqrt{\left(c_{k,i}\eta_k + \frac{1+c_{k,i}}{d_{k,i}}\right)^2 - \frac{4c_{k,i}^2\eta_k}{d_{k,i}}}}{2} < c_{k,i}\eta_k.$$

Based on the fact that the allocated resources should be higher than the required resource $c_{k,i}\eta_k$, the final solution of (32) is (36). Similarly we can find the resource request of $\hat{\zeta}_{k,j}$ for communication resource.

2) MILP problem: Based on the resource requests $\hat{v}_{k,i}$ and $\hat{\zeta}_{k,j}$, we then have problem $P2$:

$P2$:

$$\begin{aligned}\min P'(\boldsymbol{a},\boldsymbol{\varphi}) = \sum_{n\in\mathcal{N}^S}\Bigg[\delta_n\cdot\Bigg(p^A + \sum_{l\in\mathcal{L}^U\cap O_n} p_l^T \\ + \sum_{l\in\mathcal{L}^S\cap O_n}\frac{\sum\limits_{k\in\mathcal{K},j\in\mathcal{E}_k}\varphi_{k,j,l}\hat{\zeta}_{k,j}}{R_l}\cdot p_{\max}^T \\ + \sum_{k\in\mathcal{K},i\in\mathcal{V}_k} a_{k,i,n}\hat{v}_{k,i}\cdot p^C\Bigg) + (1-\delta_n)\,p^S\Bigg] \\ - \sum_{k\in\mathcal{K}} Q\sigma_k.\end{aligned} \tag{37}$$

s.t.

$$(12)-(16),(19)-(20),(22),(24)-(27),$$

$$C_n - \sum_{k\in\mathcal{K},i\in\mathcal{V}_k} a_{k,i,n}\hat{v}_{k,i} \geq 0, \forall n\in\mathcal{N}^S, \tag{38}$$

$$R_l - \sum_{k\in\mathcal{K},j\in\mathcal{E}_k}\varphi_{k,j,l}\hat{\zeta}_{k,j} \geq 0, \forall l\in\mathcal{L}^S, \tag{39}$$

$$p_l^T \geq \sum_{k\in\mathcal{K},j\in\mathcal{E}_k}\varphi_{k,j,l}\sum_{k\in\mathcal{K},j\in\mathcal{E}_k}\frac{HN_0}{(\alpha+\beta\log\phi_l)\left(2^{\hat{\zeta}_{k,j}/H}-1\right)}, \quad \forall l\in\mathcal{L}^U. \tag{40}$$

The problem $P2$ leverages the split delay from constraint (23) and the computed $v_{k,i}^{\hat{}}$ and $\hat{\zeta_{k,j}}, \forall k\in\mathcal{K}, i\in\mathcal{V}_k, j\in\mathcal{E}_k$. Thus, the constraints (17) and (18) in the original problem $P1$ are transformed into (38) and (39) respectively. Constraint (21) becomes (40). Consequently, the problem $P2$ is a MILP where we only need to determine the binary assignment variables $\boldsymbol{a}$ and $\boldsymbol{\varphi}$.

V. Flexible Service Placement based on Equivalent Bandwidth

Solving the problem $P2$ gives a candidate placement for services. However, the solution of $P2$ may not be feasible and optimal for the original problem $P1$ for the following reasons.

a) We assume each edge is placed on a one-hop path to calculate the resource request $\hat{\zeta}_{k,j}$ based on $d_{k,j}$, which may be inconsistent with the final placement result. The delay constraint may be violated.
b) When multiple components of a service are placed to the same edge server, the actual delay of their intermediate edges is negligible. Reserving delay to the intermediate edges is unnecessary. More than needed resources may be allocated to the service.

Thus, we need to reconfigure the delay splitting of services to avoid resource waste and guarantee the total delay constraint. Moreover, to improve the flexibility of resource allocation, we use the equivalent bandwidth to allocate resources to edges that are placed to paths with multiple hops.

A. Equivalent bandwidth

Assume the edge j in service G_k is placed to a physical path $Y_{k,j}$. In order to bring the superiority of flexible resource allocation into full play, the diversity in terms of the amount of allocated resources for links on $Y_{k,j}$ is allowed, that is $\zeta_{k,j,l} = \zeta_{k,j,l'}, \forall l,l'\in Y_{k,j}, l\neq l'$ is not always true. Therefore, the equivalent bandwidth is defined as follows.

Definition 1. *The **equivalent bandwidth** of any edge j in service G_k with delay constraint $d_{k,j}$ is a set of resources $\{\zeta_{k,j,l}, \forall l\in Y_{k,j}\}$ that satisfies*

$$\arg\min_{\{\zeta_{k,j,l},\forall l\in Y_{k,j}\}}\left\{\sum_{l\in Y_{k,j}}\zeta_{k,j,l} | \tau_{k,j}^E \leq d_{k,j}, \zeta_{k,j,l}\leq f_l, \forall l\in Y_{k,j}\right\}, \tag{41}$$

where f_l denotes the residual bandwidth capacity of link l defined as the total maximum data rate R_l minus the current load of link l.

The equivalent bandwidth intends to find a set of allocated resources along $Y_{k,j}$ to minimize the total resource consumption while satisfying the delay request of the edge. By allocating different amounts of resources to links on each placed path, bottleneck links whose residual bandwidth capacity is lower than others would not lead to placement failure. In order to find the optimal equivalent bandwidth in (41), we propose the following theorem.

Theorem 1. *The problem in (41) is equivalent to*

$$\arg\min_{\{\zeta_{k,j,l},\forall l\in Y_{k,j}\}}\left\{\sum_{l\in Y_{k,j}}|\zeta_{k,j,l}-\zeta_0| \tau_{k,j}^E \leq d_{k,j}, \zeta_{k,j,l}\leq f_l, \forall l\in Y_{k,j}\right\}, \tag{42}$$

where ζ_0 ensures the delay request $d_{k,j}$ is satisfied when $\zeta_{k,j,l} = \zeta_0, \forall l\in Y_{k,j}$.

Proof. See Appendix VII-A. □

B. Edge allocation based on equivalent bandwidth

It can be drawn by Theorem 1 that the edge placement with the minimum resource cost is to allocate the same amount of resources from all links on the placed path. However, when the resources of some links are not enough, allocating the amount of resources as similar as possible is desirable for the objective of resource consumption minimization. Thus, we propose the edge allocation algorithm based on equivalent bandwidth as in **Algorithm 3**.

Algorithm 3 allocates resources for links with less residual bandwidth in priority to make sure the delay constraint of the edge is guaranteed and reduce the differences between the allocated amount of resources of physical links. Binary search is used for computing ζ_0 (lines 5-16), where the initial maximum and minimum values of ζ_0, i.e., $\zeta_{\max}$ and $\zeta_{\min}$ are set to be the maximum residual bandwidth capacity and 0 respectively. In each iteration, the center of the search space $[\zeta_{\min}, \zeta_{\max}]$ is taken to compute the delay, based on which the search space is narrowed till its range is less than a small value ε. Given the computed ζ_0, links whose residual bandwidth capacity is less than ζ_0 will allocate all the residual bandwidth capacity f_l to the edge (line 19). After that, a new ζ_0 is computed for the left unallocated links, where the delay of already allocated links is determined based on the allocated resources (line 12).

Algorithm 3: Edge allocation based on equivalent bandwidth

Input: $b_{k,j}, F, \overline{X^2}, \eta_k, d_{k,j}, Y_{k,j} f_l, \forall l \in Y_{k,j}, \varepsilon$
Output: $\zeta_{k,j,l}, \forall l \in Y_{k,j}$

1 Initialize $\zeta_{k,j,l} \leftarrow -1, \forall l \in Y_{k,j}, M \leftarrow \eta_k \overline{X^2} b_{k,j} / (2F), i \leftarrow 0$;
2 **while** $\exists \zeta_{k,j,l} = -1, \forall l \in Y_{k,j}$ *or* $i \leq |Y_{k,j}|$ **do**
3 $\zeta_{\min} \leftarrow 0$;
4 $\zeta_{\max} \leftarrow \max\{f_l, \forall l \in Y_{k,j}\}$;
5 **while** $|\zeta_{\max} - \zeta_{\min}| \leq \varepsilon$ **do**
6 $\zeta_0 \leftarrow (\zeta_{\max} + \zeta_{\min})/2$;
7 $d_0 \leftarrow 0$;
8 **for** $l \in Y_{k,j}$ **do**
9 **if** $\zeta_{k,j,l} = -1$ **then**
10 $d_0 \leftarrow d_0 + \frac{M\zeta_0}{\zeta_0 - b_{k,j}} + \frac{b_{k,j}}{\zeta_0}$;
11 **else**
12 $d_0 \leftarrow d_0 + \frac{M\zeta_{k,j,l}}{\zeta_{k,j,l} - b_{k,j}} + \frac{b_{k,j}}{\zeta_{k,j,l}}$;
13 **if** $d_0 < d_{k,j}$ **then**
14 $\zeta_{\min} \leftarrow \zeta_0$
15 **else**
16 $\zeta_{\max} \leftarrow \zeta_0$
17 **for** $l \in Y_{k,j}$ **do**
18 **if** $\zeta_{i,j,k} = -1$ *and* $f_l < \zeta_0$ **then**
19 $\zeta_{k,j,l} \leftarrow f_l$;
20 $f_l \leftarrow f_l - \zeta_{k,j,l}$;
21 $i \leftarrow i + 1$;
22 **if** $\{\zeta_{k,j,l}, \forall l \in Y_{k,j}\}$ *satisfies the delay constraint of edge* j **then**
23 return succeed;
24 **else**
25 return fail;

C. ESBC algorithm

On the basis of the problem $P2$ and **Algorithm 3**, the ESBC algorithm is proposed. In **Algorithm 4**, the continuous time is divided into multiple time windows. Services that arrive in each time window collected as $\mathcal{K}'$ are placed simultaneously. For each service, **Algorithms 1** and **2** are executed first to split the whole delay constraint to every component and edge (lines 6-7), based on which the resource requests of components and edges are computed (line 8). The problem $P2$ is then solved to obtain the placed servers and paths for all components and edges (line 9). In order to satisfy the delay constraint of services and avoid resource waste, the delay of each service is re-split. For each service with candidate placement solution, we iteratively do the following till all the delay constraints of components and edges can be satisfied (line 25).

a) Fix the allocation of components (line 18) or edges (line 23) whose placed servers or paths are short of resources.
b) Re-split the delay of the service by considering the fixed delay of already allocated components and edges (lines 12-14). In the process, the allocated delay of service edges are weighted by the placed path length and the delay of edges whose end components are placed together is set to 0 such that the delay of services is strictly guaranteed.

Note that **Algorithm 3** is used to achieve flexible edge allocation.

D. Complexity analysis

Algorithm 1 iteratively carries out the **Assign_Height**() procedure for each component and edge in a service, costing $|\mathcal{V}_k| + |\mathcal{E}_k|$ steps. The time complexity of Algorithm 1 can be expressed as $O(|\mathcal{V}_k|)$ due to $|\mathcal{E}_k| = |\mathcal{V}_k| - 1, \forall k$. Similarly, Algorithm 2 conducts **Assign_Delay**() with the same time

Algorithm 4: ESBC algrithm

Input: $G_k, \eta_k, d_k, T_k, \forall k \in \mathcal{K}, F, \overline{X^2}, \mathcal{N}, \mathcal{L}, p^T_{\max}, \varepsilon$
Output: $a_{k,i,n}, \varphi_{k,j,l}, v_{k,i,n}, \zeta_{k,j,l}, \sigma_k, \forall k \in \mathcal{K}, i \in \mathcal{V}_k, j \in \mathcal{E}_k, n \in \mathcal{N}^S, l \in \mathcal{L}^S \cup \mathcal{L}^U$

1 Initialize $v_{k,i,n} \leftarrow -1, \zeta_{k,j,l} \leftarrow -1, \forall k \in \mathcal{K}, i \in \mathcal{V}_k, j \in \mathcal{E}_k, n \in \mathcal{N}^S, l \in \mathcal{L}^S \cup \mathcal{L}^U$;
2 **for** $t = 1, 2, \cdots$ **do**
3 Collect all services that arrive in the last time slot t as set $\mathcal{K}'$;
4 **for** $k \in \mathcal{K}'$ **do**
5 $|Y_{k,j}| \leftarrow 1, \forall j \in \mathcal{E}_k$;
6 Obtain $\psi_{k,z}, \forall z \in \mathcal{V}_k$ or $z \in \mathcal{E}_k$ using Algorithm 1;
7 Obtain $d_{k,z}, \forall z \in \mathcal{V}_k$ or $z \in \mathcal{E}_k$ using Algorithm 2;
8 Compute $\hat{v}_{k,i}, \hat{\zeta}_{k,j}, \forall k \in \mathcal{K}', i \in \mathcal{V}_k, j \in \mathcal{E}_k$;
9 Solve the problem $P2$ and extract $a_{k,i,n}, \varphi_{k,j,l}, Y_{k,z}, \sigma_k, \forall k \in \mathcal{K}', i \in \mathcal{V}_k, j \in \mathcal{E}_k$;
10 **for** $k \in \mathcal{K}'$ *with* $\sigma_k = 1$ **do**
11 **while** $\exists v_{k,i,n} = -1$ *or* $\zeta_{k,j,l} = -1$, $\forall k \in \mathcal{K}', i \in \mathcal{V}_k, j \in \mathcal{E}_k, n \in \mathcal{N}^S, l \in \mathcal{L}^S \cup \mathcal{L}^U, a_{k,i,n} = 1, \varphi_{k,j,l} = 1$ **do**
12 Obtain $\psi_{k,z}, \forall z \in \mathcal{V}_k$ or $z \in \mathcal{E}_k$ using Algorithm 1;
13 Obtain $d_{k,z}, \forall z \in \mathcal{V}_k$ or $z \in \mathcal{E}_k$ using Algorithm 2;
14 Compute $\hat{v}_{k,i}, \hat{\zeta}_{k,j}, \forall i \in \mathcal{V}_k, j \in \mathcal{E}_k$;
15 $\Omega_k \leftarrow 0$; // Judge the resource constraint
16 **for** $i \in \mathcal{V}_k$ **do**
17 **if** $\hat{v}_{k,i} \geq g_n, \exists a_{k,i,n} = 1$ *and* $v_{k,i,n} = -1$ **then**
18 $v_{k,i,n} \leftarrow g_n$;
19 $\Omega_k \leftarrow 1$;
20 **for** $j \in \mathcal{E}_k$ **do**
21 Execute Algorithm 3 and obtain $\{\zeta_{k,j,l}, \forall l \in Y_{k,j}\}$;
22 **if** *Algorithm 3 succeeds* **then**
23 $\zeta_{k,j,l} \leftarrow f_l, \forall l \in Y_{k,j}$;
24 $\Omega_k \leftarrow 1$;
25 **if** $\Omega_k = 0$ **then**
26 $v_{k,i,n} \leftarrow \hat{v}_{k,i}, \forall i \in \mathcal{V}_k, \exists a_{k,i,n} = 1$ *and* $v_{k,i,n} = -1$;
27 Fix $\zeta_{k,j,l}, \forall j \in \mathcal{E}_k, l \in Y_{k,j}$ according to Algorithm 3;
28 Update $g_n, f_l, \forall n \in \mathcal{N}^S, l \in \mathcal{L}^S \cup \mathcal{L}^U$.

complexity of $O(|\mathcal{V}_k|)$. In algorithm 3, the complexity of the binary search for computing ζ_0 is $O(\log_2(\zeta_{\max} - \zeta_{\min}))$ where $\zeta_{\max}$ and $\zeta_{\min}$ correspond to the highest data rate of physical links R_l and 0, respectively. The binary search process is executed multiple times (at most $|Y_{k,j}|$) in Algorithm 3, resulting in the overall time complexity of $O(\log_2 R_l)$.

The worst case of ESBC algorithm happens when all services in $\mathcal{K}$ arrive simultaneously. The initial delay splitting of Algorithms 1 and 2 costs $O(|\mathcal{K}|\,|\mathcal{V}_k|)$ steps. Besides, the problem $P2$ can be solved using existing methods with polynomial complexity. Moreover, the resource reconfiguration costs $O(|\mathcal{V}_k| + |\mathcal{V}_k| + |\mathcal{E}_k| \log_2 R_l) = O(|\mathcal{V}_k| + |\mathcal{V}_k| \log_2 R_l)$ each time (including Algorithms 1, 2, and 3) and is carried out $|\mathcal{V}_k| + |\mathcal{E}_k|$ times at worst case. Finally, the time complexity of ESBC except for solving the problem $P2$ is given by $O(|\mathcal{K}|\,(|\mathcal{V}_k| + |\mathcal{E}_k|)\,(|\mathcal{V}_k| + |\mathcal{V}_k| \log_2 R_l)) = O\left(|\mathcal{K}|\,|\mathcal{V}_k|^2\,(1 + |\mathcal{V}_k| \log_2 R_l)\right)$. Overall, the time complexity of the proposed ESBC algorithm is polynomial to the size of both MEC networks and services.

VI. Simulation Results

A. Parameter settings

In the simulation, 1 MBS and 50 SBSs are generated over a 200×200 m^2 area. Each BS is equipped with an edge server. The computation capacity of each edge server is 10

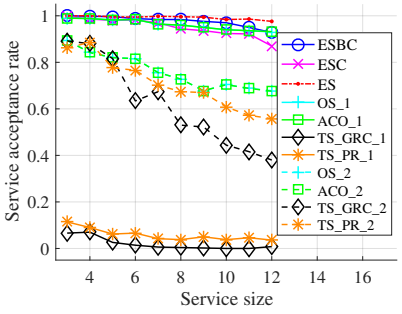


Fig. 8: Service acceptance rate under different average service sizes

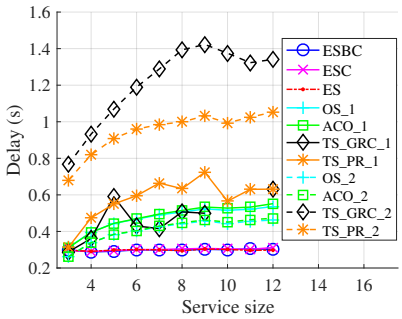


Fig. 9: Delay under different average service sizes

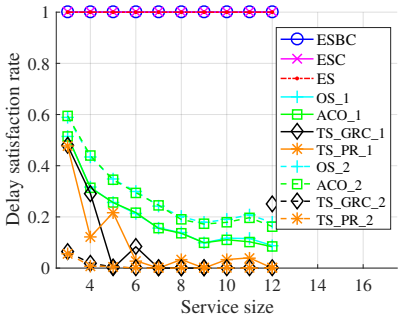


Fig. 10: Delay satisfaction rate under different average service sizes

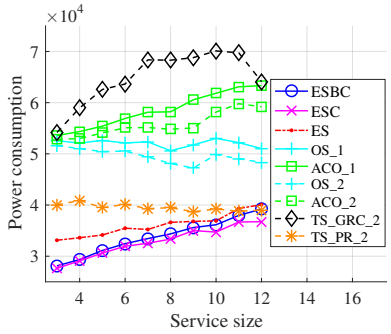


Fig. 11: Total power consumption under different average service sizes

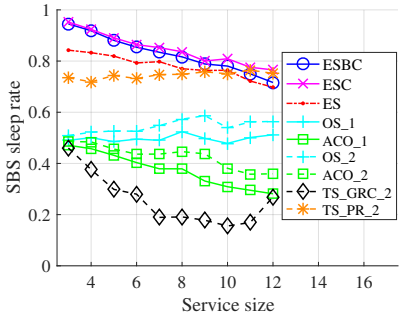


Fig. 12: SBS sleep rate under different average service sizes

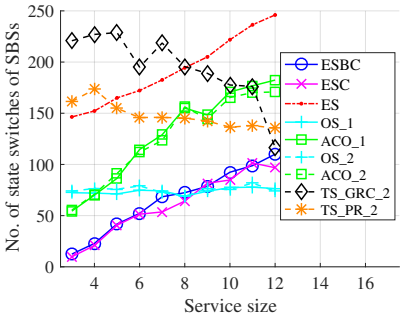


Fig. 13: Number of SBS state switches under different average service sizes

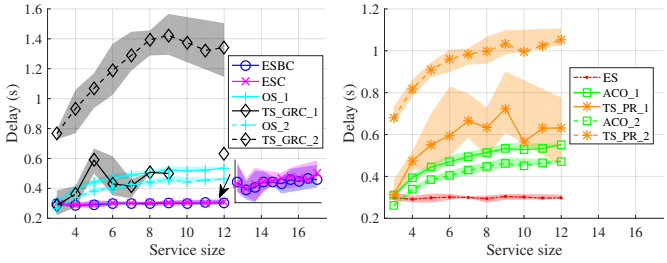


Fig. 14: The error interval of delay under different average service sizes

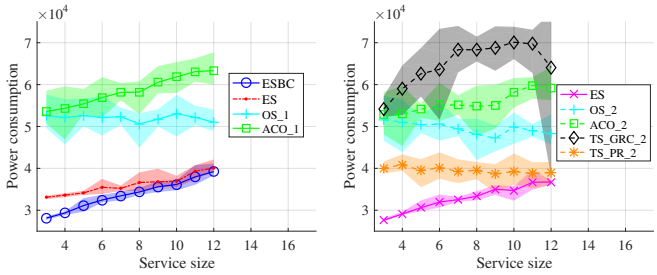


Fig. 15: The error interval of power under different average service sizes

Gigacycles/s. The path loss of wireless links over millimeter wave is $\alpha = 157.4$ and $\beta = 32$. The power density of white noise is $N_0 = -174.0$ dBm/Hz. The spectrum width for each wireless link is 20MHz. The maximum transmitting power of SBS transmitter is $p^T_{\max} = 30$dBm. The power of unit computation is $p^C = 8.2 \times 10^{-9}$W/cycle. The static power of an active SBS is $p^A = 14.9$W and the power of a sleep SBS is $p^S = 5$W.

Services arrive according to Poisson distribution. The computation requests of components and the transmitting data size of edges follow uniform distributions in [100, 1000] Megacycles and [0.5, 1] Mbits, respectively. The task arrival rate of each service is 1, and the average duration of services is 10 according to an exponential distribution. The delay requests of services are uniformly distributed in [0.1, 0.5]s. Moreover, we set F = 20 Bytes and $\overline{X^2} = 1.137\,(\mu s)^2$. In all simulations, totally 100 services are generated, each of which locates randomly in the 200×200 m^2 area. We use the IBM CPLEX optimizer to solve the MILP of problem $P2$. We conduct multiple runs with different random seeds for each approach under specific parameter settings and take an average as the final results.

B. Comparative approaches

We compare the proposed ESBC with the following approaches:

a) ESC: The ESBC without the edge allocation based on equivalent bandwidth, i.e., all links on the placed path of an edge allocate the same amount of resources to the edge;
b) ES: The ESC without cell zooming. Namely, all SBSs are initially in the active state in problem $P2$. Zero-loaded SBSs are switched into sleep state after placement;
c) ACO [14]: Using ACO with residual capacity-based dynamic pruning to search the placement of services. Zero-loaded SBSs are switched into sleep state after placement;
d) OS [53]: Place the components and edges of a service simultaneously using backtracking method. Zero-loaded SBSs are switched into sleep state after placement;
e) TS_GRC [51]: First, assign all components in a service to the physical nodes with higher general resource capacity, which is a combination of node computation capacity and

the sum of bandwidth capacities on its neighbor links. Then, assign edges to the shortest paths. Zero-loaded SBSs are switched into sleep state after placement;

f) TS_PR [52]: First, assign all components in a service using Page rank. Then, assign edges to the shortest paths. Zero-loaded SBSs are switched into sleep state after placement.

Taking the queuing delay into account, the allocated resources of both components and edges should be higher than the requests such that the expected queuing delay in (4) and (6) will not be infinite. However, the amount of additional reserved resources is unknown due to the unknown delay splitting. Thus, we analyze different amounts of additional reserved resources for ACO, OS, TS_GRC, and TS_PR as follows:

a) ACO_1, OS_1, TS_GRC_1, TS_PR_1: Allocate additional 2000M CPU cycles for each component and 100Mbits for each edge;
b) ACO_2 and OS_2: Allocate additional 2500M CPU cycles for each component and 150Mbits for each edge;
c) TS_GRC_2 and TS_PR_2: Allocate additional 1000M CPU cycles for each component and 10Mbits for each edge;

C. Performance metrics

We compare the ESBC algorithm with others on the following performance:

a) Service acceptance rate: The number of successfully placed services over the total number of services;
b) Delay: The average actual delay of all accepted services;
c) Delay satisfaction rate: The number of services whose actual delay satisfies its delay constraint over the total number of accepted services;
d) Power: Total power consumption;
e) SBS sleep rate: The number of sleep SBSs over the total number of SBSs;
f) Number of state switches of SBSs: The total number of state switches, i.e., from active (sleep) state to sleep (active) state, of all SBSs.

D. Performance analysis

1) Performance over varying service size: The comparison between ESBC and other methods under fixed service arrival rate and varying service size (i.e., the average number of components in each service) is shown in Figs. 8-15. It is demonstrated from Fig. 8 that the service acceptance rate decreases with the increase of service size on account that it becomes harder to place services with more components, more complex topology, and higher resource consumption. If we increase the additional reserved resources, the service acceptance rates decrease due to resource capacity limitation comparing ACO_2 to ACO_1, OS_2 to OS_1, TS_GRC_1 to TS_GRC_2, and TS_PR_1 to TS_PR_2. More importantly, compared to OS, ACO, TS_GRC, and TS_PR, the service acceptance rates of ESBC, ESC, and ES decrease less while the service size increases. More services are placed successfully in ESBC than ESC due to flexible edge allocation based on equivalent bandwidth. A similar number of services are placed successfully in ESBC and ES. However, ES experiences higher power consumption as shown in Fig. 11.

For methods with fixed additional reserved resources, i.e., ACO, OS, TS_GRC, and TS_PR, the delay of services rises with the increase of service size, as shown in Fig. 9, resulting in the decrease of delay satisfaction rate as shown in Fig. 10. Increasing the additional reserved resources brings lower delay and higher service acceptance rate. However, finding a fixed amount of additional reserved resources for different services with different delay constraints is difficult. ESBC, ESC, and EC execute the delay splitting and resource allocation iteratively to satisfy the delay constraints of all services. The average delay is lower than 0.4s, and a 100% delay satisfaction rate is performed.

The power consumption of all methods goes up while the service size increases, as depicted in Fig. 11. In OS, ACO, TS_GRC, TS_PR, and ES, services are placed to the SBSs that locate close users, resulting in lower SBS sleep rates as shown in Fig. 12. Higher SBS sleep rate and lower power are achieved in ESBC and ESC. Note that TS_GRC_1 and TS_PR_1 have lower power consumption and higher SBS sleep rate because their service acceptance rates are extremely low as shown in Fig. 8. The power consumption and SBS sleep rate of ESBC is slightly worse than ESC due to its higher service acceptance rate. As high as 0.96 of SBS sleep rate can be achieved in ESBC and the SBS sleep rate can still be 0.78 for large scale services.

The number of SBS state switches of ESBC is also smaller than other methods, as shown in Fig. 13. That is because the current state of SBSs is considered at the beginning of each time window in ESBC to prevent from unnecessary state switches. Overall, ESBC outperforms OS, ACO, TS_GRC, and TS_PR in service acceptance rate, delay, delay satisfaction rate, SBS sleep rate, and the number of state switches of SBSs. A higher service acceptance rate is achieved in ESBC compared to ESC, with similar performance in other aspects. Compared to ES, lower power consumption, higher SBS sleep rate and lower state switches of SBSs are performed by ESBC.

The minimum and maximum delay of different approaches under different service sizes is depicted in Fig. 14 using the dashed area behind each curve. Compared to ASO, OS, TS_GRC, and TS_PR, ESBC has a remarkably narrower error interval, which is comparable to ESC and ES. Moreover, the error interval of power consumption for different approaches under varying service sizes is demonstrated in Fig. 15 where ESBC also performs the best. Due to page limitations, we will not put all the interval error results of all curves. Instead, the variance of all the performance metrics for all approaches with different service sizes is shown in Fig. 23 in Appendix VII-B.

2) Performance over varying arrival rates of services: ESBC outperforms others for varying arrival rates of services while the service size stays in [2,8] randomly as shown in Figs. 16-22. We fix the total running time to 10 time windows and vary the number of services for each arrival rate setting. The service acceptance rate of ESBC reaches almost 100% while the arrival rate of services changes from 1 to 12. The average delay of services for ESBC is below 0.5s, which 100%

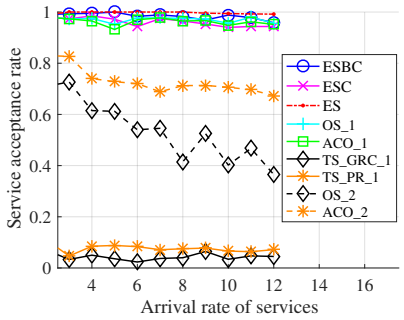


Fig. 16: Service acceptance rate under different service arrival rates

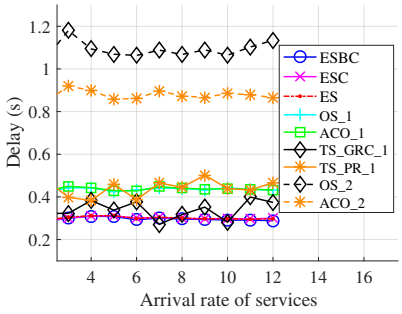


Fig. 17: Delay under different service arrival rates

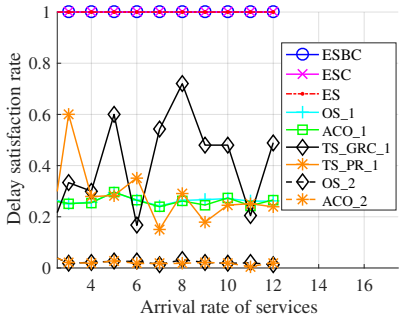


Fig. 18: Delay satisfaction rate under different service arrival rates

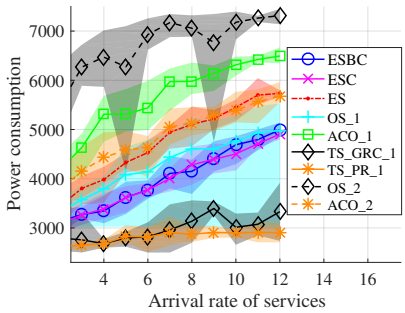


Fig. 19: Total power consumption under different service arrival rates

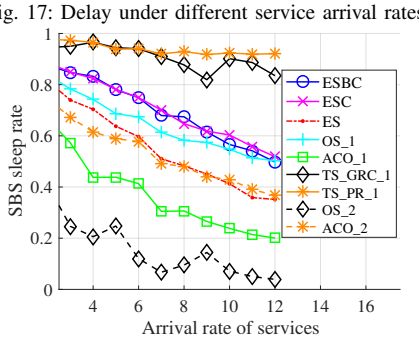


Fig. 20: SBS sleep rate under different service arrival rates

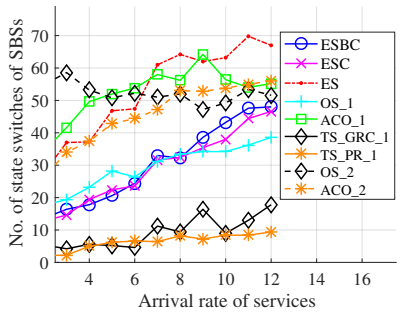


Fig. 21: Number of SBS power state switches under different service arrival rates

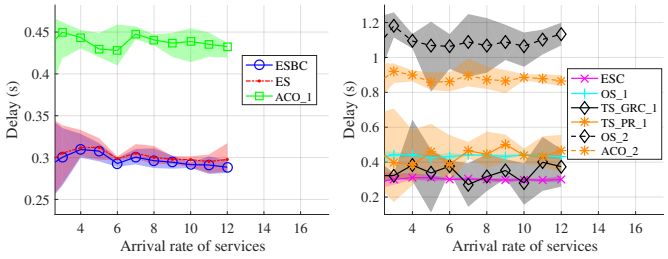


Fig. 22: The error interval of delay under different service arrival rates

satisfies the delay constraints of services. However, the delay satisfaction rates of ACO, OS, TS_GRC, and TS_PR are lower. The power consumption of all approaches increases with the increase of arrival rate of services, as shown in Fig. 19. ESBC consumes the least power compared to others except for OS_2 and ACO_2 whose service acceptance rates are extremely low. The highest SBS sleep rate of ESBC reaches 87% when the arrival rate of services is 1 and still 50% for the arrival rate of 12. ESBC also shows a lower number of state switches of SBSs than others. Furthermore, the error intervals of power consumption and delay of ESBC under different service arrival rates as shown in Figs. 19 and 22 are relatively small. More results about the variance of different metrics for approaches with different service arrival rates can be found in Fig. 24 of Appendix VII-B.

VII. Conclusion

In this paper, we have proposed the ESBC algorithm to achieve energy-efficient and flexible service placement in MEC networks with cell zooming. First, an efficient delay splitting strategy is designed to transform the joint service placement and SBS power control problem into a MILP. Furthermore, the equivalent bandwidth for an edge is defined. The edge allocation based on equivalent bandwidth is proposed to improve the probability of successful service placement and reduce resource fragments. Simulation results have shown that ESBC can achieve better performance in service acceptance rate, power, delay, SBS sleep rate, and the number of state switches of SBSs. In future work, we will consider the mobility of users and more advanced intelligent services that feature neural network architectures to enable more efficient edge intelligence.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (U21B2005), in part by the Nature Science Foundation of Chongqing under Grant cstc2021jcyj-msxmX0404, in part by the Chongqing Municipal Education Commission under Grant KJQN202100643, CXQT21019, and in part by the China Postdoctoral Science Foundation under Grant 2021M700563.

Appendix

A. Proof of Theorem 1

Proof. Define $M := \eta_k \overline{X^2} b_{k,j}/2F$, by the definition of ζ_0, we have,

$$\sum_{l \in Y_{k,j}} \left(\frac{M\zeta_0}{\zeta_0 - b_{k,j}\eta_k} + \frac{b_{k,j}}{\zeta_0} \right) \leq d_{k,j}. \tag{43}$$

If we decrease the allocated resource of a link l by $\Delta > 0$ with resulted increased delay, then the allocated resource of

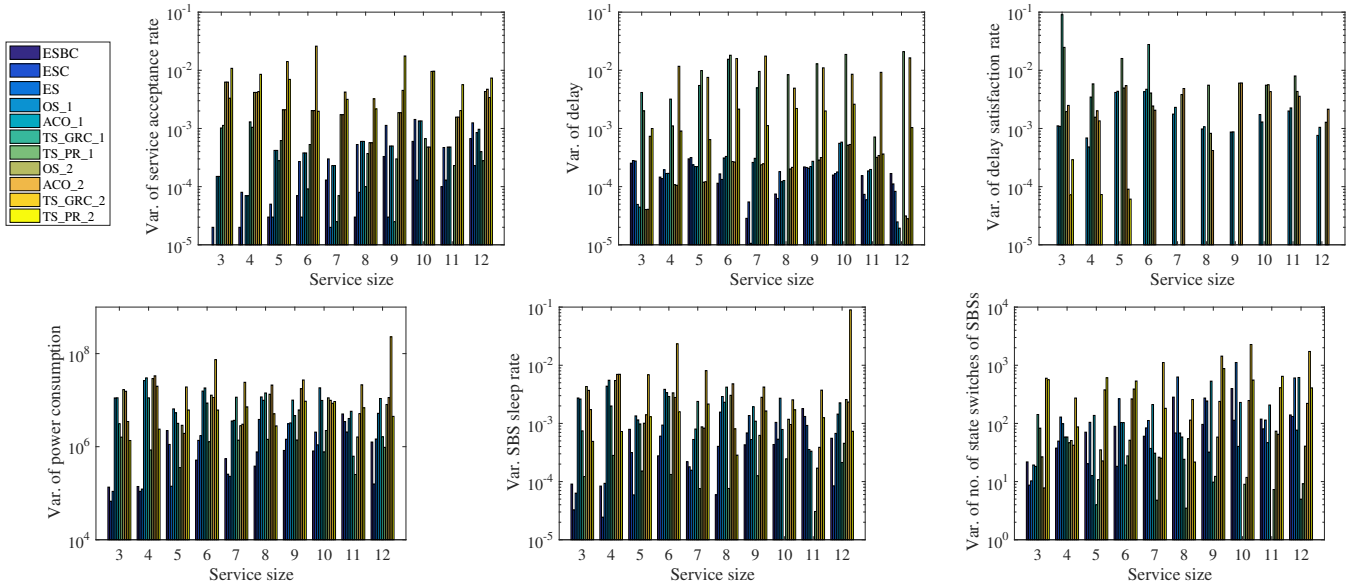


Fig. 23: Variance of different performance metrics under different average service sizes

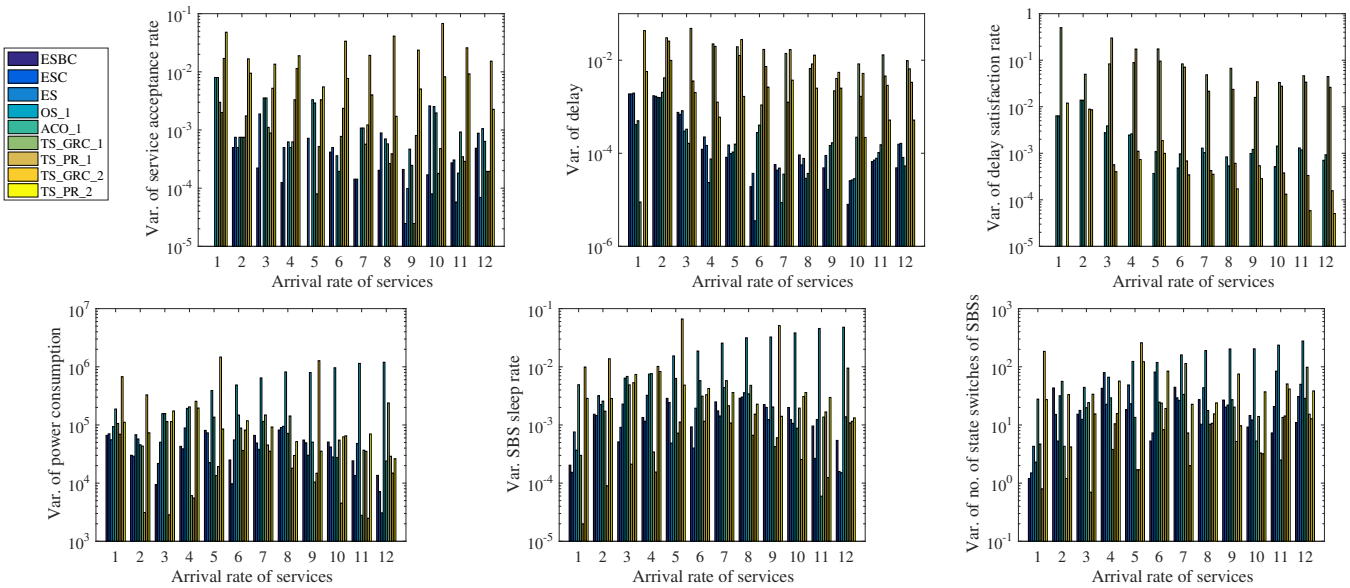


Fig. 24: Variance of different performance metrics under different service arrival rates

another link l' should be increased by $\Delta' > 0$ such that the total delay request can still be hold. Thus,

$$\left(\frac{M(\zeta_0-\Delta)}{\zeta_0-\Delta-b_{k,j}\eta_k}+\frac{b_{k,j}}{\zeta_0-\Delta}+\frac{M\left(\zeta_0+\Delta'\right)}{\zeta_0+\Delta'-b_{k,j}\eta_k}+\frac{b_{k,j}}{\zeta_0+\Delta'}\right) \sum_{l''\in Y_{k,j},l''\neq l,l''\neq l'}\left(\frac{M\zeta_0}{\zeta_0-b_{k,j}\eta_k}+\frac{b_{k,j}}{\zeta_0}\right)= \sum_{l\in Y_{k,j}}\left(\frac{M\zeta_0}{\zeta_0-b_{k,j}\eta_k}+\frac{b_{k,j}}{\zeta_0}\right), \tag{44}$$

which is equivalent to

$$\frac{M(\zeta_0-\Delta)}{\zeta_0-\Delta-b_{k,j}\eta_k}+\frac{b_{k,j}}{\zeta_0-\Delta}+\frac{M\left(\zeta_0+\Delta'\right)}{\zeta_0+\Delta'-b_{k,j}\eta_k}+\frac{b_{k,j}}{\zeta_0+\Delta'} = 2\left(\frac{M\zeta_0}{\zeta_0-b_{k,j}\eta_k}+\frac{b_{k,j}}{\zeta_0}\right). \tag{45}$$

The left-hand side of the above can be rearranged as

$$\begin{aligned}
&\frac{M(\zeta_0-\Delta)}{\zeta_0-\Delta-b_{k,j}\eta_k}+\frac{b_{k,j}}{\zeta_0-\Delta}+\frac{M\left(\zeta_0+\Delta'\right)}{\zeta_0+\Delta'-b_{k,j}\eta_k}+\frac{b_{k,j}}{\zeta_0+\Delta'}\\
&= M+\frac{Mb_{k,j}\eta_k}{\zeta_0-b_{k,j}\eta_k-\Delta}+M+\frac{Mb_{k,j}\eta_k}{\zeta_0-b_{k,j}\eta_k+\Delta'}+\frac{b_{k,j}}{\zeta_0-\Delta}+\frac{b_{k,j}}{\zeta_0+\Delta'}\\
&= \frac{Mb_{k,j}\eta_k\left(2\zeta_0-2b_{k,j}\eta_k+\Delta'-\Delta\right)}{(\zeta_0-b_{k,j}\eta_k-\Delta)(\zeta_0-b_{k,j}\eta_k+\Delta')}+\frac{b_{k,j}\left(2\zeta_0+\Delta'-\Delta\right)}{(\zeta_0-\Delta)(\zeta_0+\Delta')}+2M\\
&= \frac{Mb_{k,j}\eta_k\left(2\zeta_0-2b_{k,j}\eta_k+\Delta'-\Delta-\frac{\Delta\Delta'}{\zeta_0-b_{k,j}\eta_k}\right)+\frac{Mb_{k,j}\eta_k\Delta\Delta'}{\zeta_0-b_{k,j}\eta_k}}{(\zeta_0-b_{k,j}\eta_k)^2+(\zeta_0-b_{k,j}\eta_k)(\Delta'-\Delta)-\Delta\Delta'}+\\
&\quad\frac{b_{k,j}\left(2\zeta_0+\Delta'-\Delta-\frac{\Delta\Delta'}{\zeta_0}\right)+\frac{b_{k,j}\Delta\Delta'}{\zeta_0}}{\zeta_0^2+\zeta_0(\Delta'-\Delta)-\Delta\Delta'}+2M\\
&= \frac{Mb_{k,j}\eta_k(\zeta_0-b_{k,j}\eta_k)+\frac{Mb_{k,j}\eta_k\Delta\Delta'}{\zeta_0-b_{k,j}\eta_k}}{(\zeta_0-b_{k,j}\eta_k)^2+(\zeta_0-b_{k,j}\eta_k)(\Delta'-\Delta)-\Delta\Delta'}+\frac{b_{k,j}\zeta_0+\frac{b_{k,j}\Delta\Delta'}{\zeta_0}}{\zeta_0^2+\zeta_0(\Delta'-\Delta)-\Delta\Delta'}\\
&\quad+2M+\frac{Mb_{k,j}\eta_k}{\zeta_0-b_{k,j}\eta_k}+\frac{b_{k,j}}{\zeta_0}=2\left(\frac{M\zeta_0}{\zeta_0-b_{k,j}\eta_k}+\frac{b_{k,j}}{\zeta_0}\right).
\end{aligned}$$

Thus,

$$\frac{Mb_{k,j}\eta_k(\zeta_0-b_{k,j}\eta_k)+\frac{Mb_{k,j}\eta_k\Delta\Delta'}{\zeta_0-b_{k,j}\eta_k}}{(\zeta_0-b_{k,j}\eta_k)^2+(\zeta_0-b_{k,j}\eta_k)(\Delta'-\Delta)-\Delta\Delta'}+\frac{b_{k,j}\zeta_0+\frac{b_{k,j}\Delta\Delta'}{\zeta_0}}{\zeta_0^2+\zeta_0(\Delta'-\Delta)-\Delta\Delta'} = \frac{M\zeta_0}{\zeta_0-b_{k,j}\eta_k}+\frac{b_{k,j}}{\zeta_0}-M. \quad (46)$$

Based on the fact that higher Δ leads to higher Δ' and $\Delta\Delta'$, it can be observed from (46) that, in order to maintain the equality for given $b_{k,j}$, F, $\overline{X^2}$ and η_k, $\Delta'-\Delta$ should be higher too, meaning that $\sum_l \zeta_{k,j,l} = \sum_l \zeta_0 + \Delta - \Delta'$ will increase for increasing Δ. Therefore, $\min \sum_l \zeta_{k,j,l}$ is equivalent to $\min(\Delta-\Delta')$ as well as $\min \sum_l \zeta_{k,j,l} - \sum_l \zeta_0$, that is $\min \sum_l |\zeta_{k,j,l} - \zeta_0|$. Thus, the theorem is proved. □

B. Additional simulation results

The variance of different performance metrics for approaches with varying service sizes and service arrival rates are respectively shown in Figs. 23 and 24.

REFERENCES

[1] G. Kalfas, C. Vagionas, A. Antonopoulos, E. Kartsakli, A. Mesodiakaki, S. Papaioannou, P. Maniotis, J. S. Vardakas, C. Verikoukis, and N. Pleros, "Next generation fiber-wireless fronthaul for 5G mmwave networks," *IEEE Communications Magazine*, vol. 57, no. 3, pp. 138–144, 2019.

[2] D. Soldani and A. Manzalini, "Horizon 2020 and beyond - on the 5g operating system for a true digital society," *Vehicular Technology Magazine*, vol. 10, no. 1, pp. 32–42, 2015.

[3] J. Ren, Y. He, G. Huang, G. Yu, Y. Cai, and Z. Zhang, "An edge-computing based architecture for mobile augmented reality," *IEEE Network*, vol. 33, no. 4, pp. 162–169, 2019.

[4] 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, 04 2010.

[5] W. Shi, C. Jie, Z. Quan, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[6] S. M, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[7] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5031–5044, 2019.

[8] M. McHugh, "GPUs are the new star of moore's law, nvidia channel boss claims," 2018. [Online]. Available: https://www.channelweb.co.uk/crn-uk/news/3032004/gpus-are-the-new-star-of-moores-law-nvidia-channel-boss-claims

[9] A. Wong, "The mobile GPU comparison guide rev. 18.2," 2018. [Online]. Available: https://www.techarp.com/computer/ mobile-gpu-comparison-guide/

[10] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y. J. A. Zhang, "The roadmap to 6g - ai empowered wireless networks," *IEEE Communications Magazine*, vol. 57, no. 8, pp. 84–90, 2019.

[11] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the edge," *Proceedings of the IEEE*, vol. 107, no. 11, pp. 2204–2239, 2019.

[12] X. Zhang, Y. Wang, S. Lu, L. Liu, L. Xu, and W. Shi, "Openei: An open framework for edge intelligence," *ArXiv preprint*, vol. abs/1906.01864, 2019.

[13] G. Gui, M. Liu, F. Tang, N. Kato, and F. Adachi, "6g: Opening new horizons for integration of comfort, security, and intelligence," *IEEE Wireless Communications*, vol. 27, no. 5, pp. 126–132, 2020.

[14] P. Han, Y. Liu, and L. Guo, "Interference-aware online multi-component service placement in edge cloud networks and its ai application," *IEEE Internet of Things Journal*, pp. 1–1, 01 2021.

[15] Y. Xu, G. Gui, H. Gacanin, and F. Adachi, "A survey on resource allocation for 5g heterogeneous networks: Current research, future trends, and challenges," *IEEE Communications Surveys Tutorials*, vol. 23, no. 2, pp. 668–695, 2021.

[16] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, 2015.

[17] G. Subha, D. Debashis, D. Priti, and M. Anwesha, "5g-zoom-game: small cell zooming using weighted majority cooperative game for energy efficient 5g mobile network," *Wireless Networks*, vol. 26, no. 3, p. 349–372, 2018.

[18] H. Y. Lateef, M. Z. Shakir, M. Ismail, A. Mohamed, and K. Qaraqe, "Towards energy efficient and quality of service aware cell zooming in 5g wireless networks," in *IEEE Vehicular Technology Conference (VTC2015-Fall)*, 2015, pp. 1–5.

[19] M. Wakaiki, K. Suto, K. Koiwa, K. Liu, and T. Zanma, "Model predictive cell zooming for energy-harvesting small cell networks," in *IEEE International Conference on Communications*, 2018, pp. 1–6.

[20] M. Wakaiki, K. Suto, K. Koiwa, K. Z. Liu, and T. Zanma, "A control-theoretic approach for cell zooming of energy harvesting small cell networks," *IEEE Transactions on Green Communications and Networking*, vol. 3, no. 2, pp. 329 – 342, 2019.

[21] 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*, p. 103366, 2022.

[22] Y. Yin, M. Liu, G. Gui, H. Gacanin, H. Sari, and F. Adachi, "Qos-oriented dynamic power allocation in noma-based wireless caching networks," *IEEE Wireless Communications Letters*, vol. 10, no. 1, pp. 82–86, 2020.

[23] T. G. Rodrigues, K. Suto, H. Nishiyama, N. Kato, and K. Temma, "Cloudlets activation scheme for scalable mobile edge computing with transmission power control and virtual machine migration," *IEEE Transactions on Computers*, vol. 67, no. 9, pp. 1287–1300, 2018.

[24] H. Jiang, Z. Xiao, Z. Li, J. Xu, and D. Wang, "An energy-efficient framework for internet of things underlaying heterogeneous small cell networks," *IEEE Transactions on Mobile Computing*, vol. pp, no. 99, pp. 1–1, 2020.

[25] V. D. Maio and I. Brandic, "Multi-objective mobile edge provisioning in small cell clouds," in *ACM/SPEC International Conference on Performance Engineering*, 2019, p. 127–138.

[26] A. Alnoman and A. S. Anpalagan, "Computing-aware base station sleeping mechanism in h-cran-cloud-edge networks," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2019.

[27] C. Shu, Z. Zhao, Y. Han, G. Min, and H. Duan, "Multi-user offloading for edge computing networks: A dependency-aware and latency-optimal approach," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 1678–1689, 2020.

[28] S. Yu, R. Langar, W. Li, and X. Chen, "Coalition-based energy efficient offloading strategy for immersive collaborative applications in femto-cloud," in *IEEE International Conference on Communications (ICC)*, 2016, pp. 1–6.

[29] Q. Zhang, F. Liu, and C. Zeng, "Adaptive interference-aware vnf placement for service-customized 5g network slices," in *IEEE INFOCOM - IEEE Conference on Computer Communications*, 2019.

[30] 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.

[31] X. Lyu, H. Tian, W. Ni, Y. Zhang, P. Zhang, and R. P. Liu, "Energy-efficient admission of delay-sensitive tasks for mobile edge computing," *IEEE Transactions on Communications*, vol. 66, no. 6, pp. 2603–2616, 2018.

[32] S. Yang, F. L, M. She, 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.

[33] T. T. Vu, N. V. Huynh, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, "Offloading energy efficiency with delay constraint for cooperative mobile edge computing networks," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[34] Q. Pham, T. Leanh, N. H. Tran, B. J. Park, and C. S. Hong, "Decentralized computation offloading and resource allocation for mobile-edge computing: A matching game approach," *IEEE Access*, vol. 6, pp. 75 868–75 885, 2018.

[35] M. Li, F. R. Yu, P. Si, W. Wu, and Y. Zhang, "Resource optimization for delay-tolerant data in blockchain-enabled iot with edge computing: A deep reinforcement learning approach," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[36] Q. Wang, S. Guo, J. Liu, C. Pan, and L. Yang, "Profit maximization incentive mechanism for resource providers in mobile edge computing," *IEEE Transactions on Services Computing*, 2019.

[37] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2092–2104, 2020.

[38] Y.-D. Lin, Y.-C. Lai, J.-X. Huang, and H.-T. Chien, "Three-tier capacity and traffic allocation for core,edges, and devices for mobile edge computing," *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 923–933, 2018.

[39] X. Ma, S. Wang, S. Zhang, P. Yang, C. Lin, and X. Shen, "Cost-efficient resource provisioning for dynamic requests in cloud assisted mobile edge computing," *IEEE Transactions on Cloud Computing*, vol. PP, pp. 1–1, 03 2019.

[40] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1619–1632, 2018.

[41] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 3, pp. 361–373, 2017.

[42] C. She, Y. Duan, G. Zhao, T. Q. Quek, Y. Li, and B. Vucetic, "Cross-layer design for mission-critical iot in mobile edge computing systems," *IEEE Internet of Things Journal*, vol. PP, pp. 1–1, 07 2019.

[43] L. Zhao and J. Liu, "Optimal placement of virtual machines for supporting multiple applications in mobile edge networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 7, pp. 6533–6545, 2018.

[44] L. Chen, W. Jigang, X. Long, and Z. Zhang, "Engine: Cost effective offloading in mobile edge computing with fog-cloud cooperation," *ArXiv preprint*, vol. abs/1711.01683, 2017.

[45] Maofei Deng, Hui Tian, and Bo 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.

[46] 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.

[47] Y. K. Tun, M. Alsenwi, S. R. Pandey, C. W. Zaw, and C. S. Hong, "Energy efficient multi-tenant resource slicing in virtualized multi-access edge computing," in *Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2019, pp. 1–4.

[48] M. Richart, J. Baliosian, J. Serrat, and J.-L. Gorricho, "Resource slicing in virtual wireless networks: A survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 462–476, 2016.

[49] Chen-Shang, Chang, Thomas, and A. J., "Effective bandwidth in high-speed digital networks," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 6, pp. 1091–1100, 1995.

[50] X. Zhang and Q. Zhu, "Statistical qos provisioning over d2d-offloading based 5g multimedia big-data mobile wireless networks," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2018, pp. 742–747.

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

[52] C. Xiang, S. Su, Z. Zhang, S. Kai, F. Yang, L. Yan, and W. Jie, "Virtual network embedding through topology awareness and optimization," *Computer Networks*, vol. 56, no. 6, pp. 1797–1813, 2012.

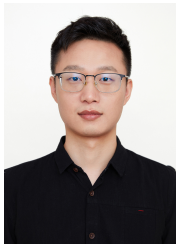
[53] P. Han, Y. Liu, and L. Guo, "QoS satisfaction aware and network reconfiguration enabled resource allocation for virtual network embedding in fiber-wireless access network," *Computer Networks*, vol. 143, pp. 30–48, 2018.

[54] Yuan Zhang, Hao Liu, Lei Jiao, and Xiaoming Fu, "To offload or not to offload: An efficient code partition algorithm for mobile cloud computing," in *IEEE International Conference on Cloud Networking (CLOUDNET)*, Nov 2012, pp. 80–86.

[55] D. Bertsekas and R. Gallager, *Data Networks (2nd Ed.)*. USA: Prentice-Hall, Inc., 1992.

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. She conducted academic research at the Department of Electrical and Electronic Engineering, Imperial College London, United Kingdom. Her research interests include wireless and optical networks, mobile edge computing, federated learning, and knowledge distillation.

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.



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 networks, 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.

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.