Interference-Aware Online Multi-Component Service Placement in Edge Cloud Networks and Its AI Application

Pengchao Han, Yejun Liu, Lei Guo

***Abstract*— Edge computing that utilizes ubiquitous edge devices locating in close proximity to users is powerful for providing QoS guaranteed computation offloading services. Towards the limited resources of edge servers and wireless links, large services can be split into multiple inter-connected components to be served by multiple edge servers cooperatively. Current works on service placement either assume unsplittable services or ignore the geographically isolated property of edge servers. They also ignore the interference among online services that share the same physical nodes/links in terms of executing delay. Namely, every service adds load to the placed nodes/links and every increment on load of nodes/links risks delay violation of existing services. To overcome above challenges, this paper emphasizes on the Interference-Aware (IA) online multi-component service placement in edge cloud networks. Firstly, the delay of tree-like services is analyzed considering the dependency among components, based on which the IA residual capacities of physical nodes, links, and paths are defined and formulated theoretically. Furthermore, we reduce the problem of multi-component service placement to be NP-hard and transform it into an Ant colony Optimization (ACO) problem to obtain the near-optimal solution. More importantly, a level traversal component ranking method and an IA dynamic pruning method are proposed for ACO to achieve faster convergence, interference awareness, and higher acceptance ratio of services. Simulation results are presented to validate the effectiveness of proposed methods. In addition, the classic AI application of image classification is experimented to further strength the motivation of IA investigation in practical.**

***Index Terms*—Service placement, edge cloud, interference, multi-component, AI application**

I. Introduction

Cloud computing [1] has gained a lot of popularity for decoupling computation- and memory-intensive tasks from end devices to reduce the device complexity and power consumption of end users. Also, by enabling Infrastructure as a Service (IaaS) [2], users ranging from an individual to an enterprise can easily deploy customized functions by using leased resources regardless the tedious infrastructure deployment, facilitating the emerging of abundant network services and applications, such as cloud storage [3], cloud Artificial Intelligence (AI) platforms [4], and could games [5]. However, facing the ever-increasing demands of users on the Quality of Service (QoS) of networks, such as the 1 ms delay, 10 Gbps transmission rate, and ultra-high privacy requirement in 5G [6], [7], cloud computing that features servers geographically away from users is suffering the bottleneck of ultra-low delay service provisioning. Thanks to the development of high-performance and low-cost communication and computing techniques such as the optical transmission and Graphic Processing Units (GPUs) [8], [9], it becomes possible to leverage the numerous and widely deployed edge devices that locate in close proximity to users for lower-latency service provisioning, i.e., edge computing [10], [11].

Devices that are capable of edge computing can play the roles of edge servers. Edge cloud networks that consist of multiple inter-connected edge servers are essential for the next generation network that bears modern applications such as A/VR and IoT, promoting the development of intelligence at the edge [12], [13]. Against the limited resources of edge servers compared with a central cloud, services that consume high resources can be split into multiple components so as to be served by multiple edge servers cooperatively [14].

While the prospect of placing services on edge cloud networks is beyond doubt, the methods on multi-component service placement on edge cloud networks for utility maximization is still under investigation. Current works on service placement either assume unsplittable services or ignore the geographically isolated property of edge servers by considering only one edge cloud layer. In addition, the consideration on QoS guarantee of services is also insufficient. Specifically, the loads of both edge servers and communication links are influenced by each online arrived service, leaving increasing executing delay for existing services that locate at the same physical node or link, while computation units or data packets of all services on each physical node or link are put into a common queue and be executed according to First-In-First-Out (FIFO). Namely, there exists ***interference*** among services in terms of executing delay, which may lead to QoS violation of services [15]. The Ant Colony Optimization (ACO) [16] is an effective method to find the near-optimal solutions for service placement by considering exploiting historic experiences and learning from the environment [17]. However, conventional ACO is executed on a fixed search graph and relies on a large number of ants and iterations to find a good solution.

We therefore have the following open questions: 1) How to formulate and thus avoid the interference among services for QoS guarantee? 2) How to coordinate the placement of

Pengchao Han is with the School of Computer Science and Engineering, Northeastern University, Shenyang, P. R. China (email: hanpengchao199@163.com).

Yejun Liu* and Lei Guo are with the School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, P. R. China (email: {yjliu, guolei}@cqupt.edu.cn, *corresponding author).

Copyright (c) 20xx IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org.

multiple components in a service, while reducing the influence to other components/services and accelerating the convergence of ACO? To answer these questions, we make the following main contributions in this paper.

1) We mathematically formulate the delay of tree-structured services considering component dependency;
2) For the first time, the Interference-Aware (IA) residual capacities of edge servers, communication links, and physical paths are defined and formulated theoretically to set up guidance for relieving the interference among services;
3) The problem of multi-component service placement with delay guarantee is modeled and analyzed based on the NP-hardness of Maximum Edge Weighted Clique (MEWC) problem;
4) We propose the ACO algorithm based on level traversal (LT) component ranking and IA dynamic pruning for multi-component service placement to achieve interference awareness, faster convergence and higher acceptance ratio of services;
5) The AI application of image classification is experimented to further enhance the motivation of IA investigation for practical effectiveness in load balancing and parallel computing.

The reminder of this paper is organized as follows. The related works are reviewed in Section II. Section III presents the system models, based on which the problem of multi-component service placement with delay guarantee is formulated and analyzed. The IA residual capacity is defined and derived for physical nodes, links, and paths respectively in Section IV. Utilizing the results in Section IV, the ACO algorithm based on LT component ranking and IA dynamic pruning is proposed in Section V. The simulation results are presented in Section VI. In addition, the experiments on AI application is demonstrated in Section VII. Finally, Section VIII concludes this paper.

II. Related Works

Related works on service placement with QoS guarantee consider diverse service structures on different edge cloud networks with various delay considerations as shown in Table I.

Different delay models of services have been applied in current works. The simplest way is to consider only processing and/or propagation time [18], [19], [20], [21], [22], [23], [24] such that the executing delay of a service is linear to the allocated resources. To make the delay model more practical, Markov Decision Process (MDP) is used in [25], [26] to virtualize online service placement process, in which the delay of a service is represented by the queue length. Another common way for delay analysis is by utilizing queuing systems. Generally, the packet arrival process at a node that holds a service can be described as a Bernoulli process and the aggregation of multiple Bernoulli processes can be modeled as a Poisson process [27]. While the data size of a service has an exponential distribution, the service time of services also follows exponential distribution for constant resource capacity. Thus, the expected packet delay on links [28], [29] and the expected computation delay [30] can be computed using M/M/1 queuing model. Each node that holds multiple services can also be characterized by M/G/1 queuing system [31], [32], where "M" indicates that the packet arrival process follows Poisson process and "G" means that the amount of bandwidth required to process the packets can follow any distribution. The M/G/n/∞ queue model is used to formulate the delay of both traditional cellular downlink/uplink transmissions and edge cloud services [33], where the distribution of service time for any service on any homogeneous server is assumed to be identical and independent. However, services with above delay models are considered unsplittable for placement in the literature.

Generally, a service of 5G and beyond such as V/AR [34], can be divided into multiple components. Those components can be assigned to different servers and work in a distributed way. Hence, services with large resource requirements can be implemented on edge cloud networks that consists of servers with limited computation and memory resources. The component dependency is introduced in [35], [36], [37], [38], specifying the time priority for executing components based on the fact that the output of one component may be the input of another (i.e., the output/input dependency). Each multi-component service can be characterized using a graph [34], [35], [36], [37], [39], [40], [41], [42], [43], [44], [45], [46], and a graph structure can be transformed into a tree [47]. Current works on multi-component service placement determine whether or not to offload any component to edge server from the local device aiming at minimizing total energy consumption or total completion time of services. However, only processing and propagation time are considered for service placement and they have ignored the geographically isolated property of edge servers. In [38], each service can be split into multiple dependent components and be offloaded to edge servers distributively, the scheduling delay is considered while minimizing the total completion time. However, the queuing delay caused by randomly arrived tasks of a service is ignored. In [47], [48], multi-component services are assigned to edge cloud networks. The co-located virtual machine interference is considered while placing virtual functions to maximize the total throughput of accepted services [48]. However, the service delay is randomly given regardless resource allocation.

III. System Models and Problem Formulation

A. Network models

The edge cloud network contains multiple multi-antenna access points (APs) that follow Homogeneous Poisson Point Process (HPPP) distribution [30]. The AP can be Small cell Base Station (SBS), e.g., femtocell or picocell [49]. Each SBS is connected to a co-located edge server with negligible communication delay as shown in Fig. 1. We assume the wireless links between SBS pairs use frequency orthogonal channels. All SBSs and edge servers are controlled by a Software Defined Network (SDN) controller. For ease of presentation, we use edge node to represent both SBSs and edge servers.

Important notations are shown in Table II. Let $\mathcal{N}$ and $\mathcal{L}$ denote the set of edge nodes and wireless links respectively.

TABLE I: Related works

Ref.	Multi-component service?	Multi-edge servers?	Delay consideration	Remark
[18], [19], [20], [21]	N	N	Only processing time	Minimize total energy consumption under delay constraints
[22], [23]	N	Y	Only processing time	-
[24]	N	Y	Only processing / propagation time	Maximize profit under delay constraints
[25]	N	Y	MDP	Power-delay tradeoff
[26]	N	Y	MDP	-
[28], [29]	N	Y	M/M/1	Minimize total resource cost under delay constraints
[30]	N	Y	M/M/1	Minimize delay under energy budgets
[31]	N	Y	M/G/1	Power-delay tradeoff
[32]	N	Y	M/G/1	Minimize the maximal packet loss probability under delay violation probability constraints
[33]	N	N	M/G/n/∞	-
[35], [36]	Y	N	Only processing / propagation time	Component dependency, minimize total completion time
[37]	Y	N	-	Component dependency, minimize total completion time
[34], [39], [40]	Y	N	-	-
[41]	Y	N	Only processing / propagation time	Minimize weighted time and energy cost
[42]	Y	N	Only processing / propagation time	-
[43], [44]	Y	N	Only processing / propagation time	Minimize total completion time
[45]	Y	N	Only processing / propagation time	Minimize total energy consumption
[46]	Y	-	-	Tree structure
[38]	Y	Y	Processing / propagation time and scheduling time	Component dependency, minimize total completion time
[47]	Y	Y	Randomly given	Minimize total energy consumption
[48]	Y	Y	Randomly given	Interference

The computation capacity of the edge server indexed by $n \in \mathcal{N}$ is represented by C_n CPU cycles/s. Denote H the total spectrum width of a wireless link and let $p_{\max}^{trans}$ be the maximum transmitting power of a SBS. The Signal-to-Noise Ratio (SNR) of link l is $\theta_l = \phi_l p_{\max}^{trans}/(N_0 H)$ where ϕ_l is the path loss and N_0 denotes the power density of white noise. According to Shannon's formula, the data rate of l is $R_l = H\log_2(1+\theta_l)$ bps. The executing units of computation and communication resources are a CPU cycle and a data packet respectively. All computation (communication) units on an edge server (wireless link) are put into a common queue and be executed according to First-In-First-Out (FIFO) disregarding which services they belong to.

TABLE II: Important notations

Symbol(s)	Description
$\mathcal{N}$, $\mathcal{L}$	Set of edge nodes and wireless links
C_n	The computation capacity of node n
H	The bandwidth of wireless links
$p_{\max}^{trans}$	The maximum transmission power of SBSs
ϕ_l	The path loss of wireless link l
N_0	The power density of white noise
R_l	The data rate of wireless link l
$\mathcal{K}$	Set of services
G_k	The task graph of the service indexed by k
$\mathcal{V}_k$, $\mathcal{E}_k$	Set of components and edges in G_k
$c_{k,i}$	The computation resource requirement of component $i \in \mathcal{V}_k$
$b_{k,j}$	The data size to be transmitted on edge $j \in \mathcal{E}_k$
η_k	The arrival rate of G_k
d_k	The delay requirement of G_k
T_k	The duration of G_k
$\beta_{k,m}$	The mth branch of G_k
$d_{k,i}^V$	The delay of component $i \in \mathcal{V}_k$
$d_{k,j}^E$	The delay of edge $j \in \mathcal{E}_k$
d_m^B	The delay of branch $m \in G_k$
$a_{k,i,n}$	Binary variable taking 1 if component $i \in \mathcal{V}_k$ is assigned to edge server n, and 0 otherwise
$\varphi_{k,j,l}$	Binary variable taking 1 if edge $j \in \mathcal{E}_k$ is assigned to wireless link l, and 0 otherwise
λ_n	The arrival rate of computation units on node n
w_n	Expected queuing delay of components on node n
$d_{k,i}^n$	The delay of serving $i \in \mathcal{V}_k$ on node n
λ_l	The arrival rate of packets on wireless link l
F	Average packet size
ρ_l	The load factor of wireless link l
w_l	Expected queuing delay of packets on link l
$\overline{X^2}$	Expected second moment of service time of packets
$d_{k,j}^l$	The delay of serving $j \in \mathcal{E}_k$ on wireless link l
I_n	Set of links that flow into node n
O_n	Set of links that flow out of node n
j_S	The source component of edge j
j_D	The destination component of edge j
$p(n,n')$, $l(n,n')$	The shortest path and link from n to n'
$\varepsilon_{n,n'}^P$, $\varepsilon_{n,n'}^L$, ε_n^N	The MAAD of $p(n,n')$, $l(n,n')$, and n
$\tau_{n,n'}^P$	The total queuing delay of $p(n,n')$
$\nu_{\eta,n,n'}^P$, $\nu_{\eta,n,n'}^L$, $\nu_{\eta,n}^N$	The η-IA residual capacity of $p(n,n')$, $l(n,n')$, and n
$\mathcal{M}_{k,i}$	The candidate domain of component $i \in \mathcal{V}_k$
$\mathcal{G}_k$	The search graph of ACO for G_k
$\gamma_{i,i'}^{n,n'}$	The heuristic factor of the edge from $n \in \mathcal{M}_{k,i}$ to $n' \in \mathcal{M}_{k,i'}$ in $\mathcal{G}_k$
$\sigma_{i,i'}^{n,n'}$	The pheromone of the edge from $n \in \mathcal{M}_{k,i}$ to $n' \in \mathcal{M}_{k,i'}$ in $\mathcal{G}_k$
$q_{i,i'}^{n,n'}$	The transition probability from $n \in \mathcal{M}_{k,i}$ to $n' \in \mathcal{M}_{k,i'}$ in $\mathcal{G}_k$
ϵ	Parameter in ϵ-greedy algorithm
ξ	The evaporation coefficient of pheromone trail
α, δ	The importance of pheromone and heuristic factors
Δ	The amount of pheromone laid by an ant
$N_{iter}^{\max}$	The maximum number of iterations of ACO
N^{ant}	Number of ants in each iteration

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 $\mathcal{V}_k$ is the set of components and $\mathcal{E}_k$ is the set of edges. Each service G_k contains multiple homogeneous tasks with the arrival rate of η_k in a duration of T_k. The computation resource requirement 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

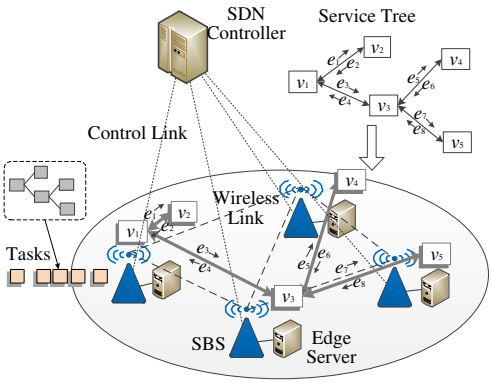

Fig. 1: Network architecture

data size of the upstream and downstream transmissions of a bidirectional edge in a service can be asymmetric.

The delay requirement of G_k is d_k, restricting the total time for completing all components in the service. It is meaningful to mention that each edge in a service represents a dependency between its start and end components. Taking the dependencies among components into account and based on the fact that any call graph of a service can be organized using a service tree [50], we also model any service $k \in \mathcal{K}$ as $G_k = \{\beta_{k,m}, \forall m\}$ where $\beta_{k,m}$ indicates the mth branch in G_k. Branches of a service G_k have following properties,

1) The root of a tree corresponds to the input of a service;
2) 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;
3) Each branch in G_k is a path of dependent components, which should be executed in sequence due to the output/input dependency;
4) The number of branches of a service equals to the number of leaf components in the service tree;
5) Different components on different branches can be executed simultaneously;

Therefore, the delay of a service G_k is equivalent to the largest accumulated delay of branches on G_k.

Taking the service tree (for the kth service without loss of generality) in Fig. 1 for example, let $d^V_{k,i}$, $d^E_{k,j}$, and $d^B_{k,m}$ denote the delay of component i, edge j, and branch m respectively. There are three branches in the service, that is, $\beta_{k,1} = \{v_1 \longleftrightarrow v_2\}$, $\beta_{k,2} = \{v_1 \longleftrightarrow v_3 \longleftrightarrow v_4\}$ and $\beta_{k,3} = \{v_1 \longleftrightarrow v_3 \longleftrightarrow v_5\}$ with respective delay of $d^B_{k,1} = d^V_{k,1} + d^E_{k,1} + d^E_{k,2} + d^V_{k,2}$, $d^B_{k,2} = d^V_{k,1} + d^E_{k,3} + d^E_{k,4} + d^V_{k,3} + d^E_{k,5} + d^E_{k,6} + d^V_{k,4}$ and $d^B_{k,3} = d^V_{k,1} + d^E_{k,3} + d^E_{k,4} + d^V_{k,3} + d^E_{k,7} + d^E_{k,8} + d^V_{k,5}$. Thus, the total delay is $\max\left\{d^B_{k,1}, d^B_{k,2}, d^B_{k,3}\right\}$. Therefore, the delay constraint for this service is $\max\left\{d^B_{k,1}, d^B_{k,2}, d^B_{k,3}\right\} \leq d_k$.

B. Delay models

Based on the models of networks and services, we formulate the delay of components and edges in this section. Let $a_{k,i,n}$ denote 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. Let $\boldsymbol{a}$ and $\boldsymbol{\varphi}$ be the set of $a_{k,i,n}$ and $\varphi_{k,j,l}$ respectively. The arrival rates of computation units (i.e., CPU cycles) of edge node $n \in \mathcal{N}$ and data packets of wireless link $l \in \mathcal{L}$ are formulated as $\lambda_n = \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{V}_k} a_{k,i,n} \cdot c_{k,i} \cdot \eta_k$ and $\lambda_l = \sum_{k\in\mathcal{K}} \sum_{j\in\mathcal{E}_k} \varphi_{k,j,l} \cdot b_{k,j} \cdot \eta_k / F$ respectively, where F denotes the average packet size.

1) Delay of service components: All components that are placed on one edge node share the computation resources of the edge server. According to the Little's formula for M/M/1 queuing system [30], for edge server $n \in \mathcal{N}$ with computation unit arrival rate λ_n and service rate C_n, the expected queuing delay of computation units on edge server n is

$$w_n = \frac{1}{C_n - \lambda_n}. \tag{1}$$

The delay of a component (e.g., component i of service k) on edge server n includes the expected queuing delay and the processing delay, i.e.,

$$d^n_{k,i} = a_{k,i,n} \left(w_n + c_{k,i}/C_n\right). \tag{2}$$

2) Delay of service edges: For wireless links, the packet arrival process follows Poisson distribution and the service process follows any distribution. According to Pollaczek-Khinchin (P-K) formula of M/G/1 queuing system [51], the expected queuing delay of data packets on any link $l \in \mathcal{L}$ is

$$w_l = \frac{\lambda_l \overline{X^2}}{2\left(1-\rho_l\right)} = \frac{\overline{X^2} R_l^2}{2F\left(R_l - \lambda_l F\right)} - \frac{\overline{X^2} R_l}{2F}, \tag{3}$$

where λ_l is the arrival rate of packets on link l and ρ_l is the load factor of link l defined by the ratio of λ_l over the packet service rate of R_l/F. $\overline{X^2}$ denotes the second moment of service time for data packets.

The delay of any edge j in G_k on a specific link l includes the expected packet queuing delay and the data transmitting delay, that is,

$$d^l_{k,j} = \varphi_{k,j,l} \left(w_l + b_{k,j}/R_l\right). \tag{4}$$

Obviously, the executing delay of a component or an edge in a service depends on the load of the placed infrastructure (i.e., edge servers and wireless links). Hence, newly arrived services that increase the load of edge servers and wireless links lead to higher queuing delay of services. Namely, every increment on load of nodes/links risks delay violation of existing services. Thus, we emphasize the interference among services that share the resources (i.e., computation and bandwidth) of the same physical infrastructure, which is different from the conventional co-channel interference in wireless networks where one user on the same channel decreases the transmitting rate of another by adding noise. To the best of our knowledge, this is the first work investigating the interference among services in edge cloud networks from the above perspective for both computation and communication resources.

C. Problem formulation

The objective of service placement is to maximize overall resource-related utility defined as

$$\begin{aligned} U\left(\boldsymbol{a}, \boldsymbol{\varphi}\right) = \sum_k T_k \eta_k \Big(\sum_i \sum_n \zeta \cdot a_{k,i,n} \cdot c_{k,i} \\ + \sum_j \sum_l \phi \cdot \varphi_{k,j,l} \cdot b_{k,j}\Big), \end{aligned} \tag{5}$$

where ζ and ϕ indicate the importance of computation and communication resources respectively. Note that the utility function can be extended to any resource-dependent revenue and cost functions such as infrastructure provider revenue, total resource cost, energy consumption and momentary budget.

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

$P1$:

$$\max\ U\left(\boldsymbol{a}, \boldsymbol{\varphi}\right). \tag{6}$$

$s.t.$

$$\sum_{n\in\mathcal{N}} a_{k,i,n} \leq 1, \forall k \in \mathcal{K}, i \in \mathcal{V}_k, \tag{7}$$

$$\begin{gathered}\textstyle\sum_{l\in I_n} \varphi_{k,j,l} - \sum_{l\in O_n} \varphi_{k,j,l} = -a_{k,j_S,n} + a_{k,j_D,n}, \\ \forall k \in \mathcal{K}, j \in \mathcal{E}_k, n \in \mathcal{N},\end{gathered} \tag{8}$$

$$\sum_{l\in I_n} \varphi_{k,j,l} \leq 1, \forall k \in \mathcal{K}, j \in \mathcal{E}_k, n \in \mathcal{N}, \tag{9}$$

$$\sum_{l\in O_n} \varphi_{k,j,l} \leq 1, \forall k \in \mathcal{K}, j \in \mathcal{E}_k, n \in \mathcal{N}, \tag{10}$$

$$C_n - \lambda_n \geq 0, \forall n \in \mathcal{N}, \tag{11}$$

$$R_l/F - \lambda_l \geq 0, \forall l \in \mathcal{L}, \tag{12}$$

$$\begin{gathered}\max\left\{\textstyle\sum_{i,j\in\beta_{k,m}} \left(\sum_{n\in\mathcal{N}} d^n_{k,i} + \sum_{l\in\mathcal{L}} d^l_{k,j}\right), \forall \beta_{k,m} \in G_k\right\} \\ \leq d_k, \forall k \in \mathcal{K},\end{gathered} \tag{13}$$

$$a_{k,i,n} = \{0,1\}, \forall k \in \mathcal{K}, i \in \mathcal{V}_k, n \in \mathcal{N}, \tag{14}$$

$$\varphi_{k,j,l} = \{0,1\}, \forall k \in \mathcal{K}, j \in \mathcal{E}_k, l \in \mathcal{L}. \tag{15}$$

Solving the problem $P1$ is equivalent to find the placement results of components (i.e., variables $\boldsymbol{a}$) and edges (i.e., variables $\boldsymbol{\varphi}$) with the maximum utility. Constraint (7) specifies that every component in a service should be assigned to one and only one edge node. The flow conservation in (8) 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 (9) and (10) specify that the edge assignment is not splittable, namely one edge to one physical path. The computation and bandwidth resource capacity constraints of edge server and wireless links are formulated in (11) and (12) respectively. The delay constraints of services are shown in (13), restricting the delay of each branch to be lower than d_k. All placement variables are constrained to be binary as in (14) and (15).

D. Problem analysis

We have following Lemmas and Theorem.

Lemma 1. *Constraints (7) - (15) are equivalent to a Boolean Satisfaction (SAT) problem.*

Proof. See Appendix-A. □

Based on Lemma 1, we have Lemma 2 as follows.

Lemma 2. *The problem P1 can be reduced to Maximum Edge Weighted Clique (MEWC) problem.*

Proof. See Appendix-B. □

Theorem 1. *The problem P1 is NP-hard.*

Proof. Theorem 1 can be proved based on Lemma 2 as in Appendix-C. □

IV. Interference-Aware (IA) Residual Capacity

As we have analyzed, Eqs. (2) and (4) indicate that delay of a(n) component(edge) depends on $\lambda_n(\lambda_l)$ due to $C_n(R_l)$ is constant for given resource capacity. In this sense, services influence each other in terms of changing loads of physical nodes and links. Intuitively, we can transform the delay constraint of a service into the IA residual capacities of the placed physical paths and nodes. The interference that future services will leave on the existing services can be avoided through using the IA residual capacity for placement instead of the general residual capacity. The shortest paths are always used for placing service edges to improve the probability of delay satisfaction.

In order to calculate the IA residual capacity, we first introduce the Maximum Allowable Additional Delay (MAAD) that a path, link, or server can tolerate. Since judging the delay satisfaction of a service is based on the executing delay of branches on the service as in (13), while a physical path, link, or node only serves a part of a branch, the MAAD of a path is defined as the minimum value of allowable additional delays of all service edges on the path such that the belonged branches (of one or multiple services) of the edges will experience acceptable delay. The MAAD of a physical link or a server is obtained from its belonged paths. We define the IA residual capacity for a physical path, link, or node as the maximum resources that the path, link, or node can provide for future services, such that the delay requirements of its existing services will not be violated.

A. IA residual capacity of physical path

As each edge in a service is placed onto a physical path $p\left(n, n'\right)$, i.e., the shortest path from node n to n', the MAAD of $p\left(n, n'\right)$, denoted by $\varepsilon^P_{n,n'}$, can be computed as the minimal residual delay of all branches it serves,

$$\begin{gathered}\varepsilon^P_{n,n'} = \min\left\{d_k - \textstyle\sum_{i,j\in\beta_{k,m}} \left(\sum_{n\in\mathcal{N}} d^n_{k,i} + \sum_{l\in\mathcal{L}} d^l_{k,j}\right), \right. \\ \left. \forall k \in \mathcal{K}, \beta_{k,m} \in G_k, \exists j \in \beta_{k,m} \text{ is placed to } p\left(n, n'\right)\right\}.\end{gathered} \tag{16}$$

Definition 1. *The η-IA residual capacity of path $p\left(n, n'\right)$ is defined as the maximum volume of communication data size $\nu^P_{\eta,n,n'}$, from a service with the task arrival rate of η, that can be added to the path, such that the delay requirements of existing services on the path are not violated.*

Remark 1. *The definition of IA residual capacity of a path (also for links and nodes) is service-wise, relating with the task arrival rate of a newly arrived service, i.e., η.*

Based on $\varepsilon^P_{n,n'}$, $\nu^P_{\eta,n,n'}$ can be obtained using the binary search method and then take the minimum compared with the η-IA residual capacities of all links and end nodes on $p\left(n, n'\right)$

Algorithm 1: $BinarySearch\left(\varepsilon^P_{n,n'},\eta\right)$

Construct $\mathcal{K}'$ by gathering all services placed in the system;
$v^{\max} \leftarrow \min\left\{\left(R_l - \sum_{k\in\mathcal{K}'}\sum_{j\in\mathcal{E}_k}\varphi_{k,j,l}\cdot b_{k,j}\cdot\eta_k\right)/\eta, \forall l\in p\left(n,n'\right)\right\}$, $v^{\min}\leftarrow 0$, $v_{prev}\leftarrow 0$;
$\tau_0 \leftarrow \tau^P_{n,n'}$;
$v \leftarrow \left(v^{\max}+v^{\min}\right)/2$;
while $|v - v_{prev}| < 0.001$ **do**
  $\lambda_l \leftarrow \lambda_l + v\cdot\eta/F, \forall l\in p\left(n,n'\right)$;
  Obtain $\tau^P_{n,n'}$ from (18) based on the new $\lambda_l, \forall l\in p\left(n,n'\right)$;
  if $\tau^P_{n,n'} > \tau_0 + \varepsilon^P_{n,n'}$ **then**
    $v^{\max}\leftarrow v$;
  else
    $v^{\min}\leftarrow v$;
  $v_{prev}\leftarrow v$;
  $v\leftarrow\left(v^{\max}+v^{\min}\right)/2$;
Return v;

for the purpose of being compatible with the IA results of links and servers,

$$\begin{aligned}\nu^P_{\eta,n,n'} = \min\Big\{&BinarySearch\left(\varepsilon^P_{n,n'},\eta\right),\\ &\min\left\{\nu^L_{\eta,z,z'},\forall z,z'\in\mathcal{N}, l\left(z,z'\right)\in p\left(n,n'\right)\right\},\\ &\min\left\{\nu^N_{\eta,z},\forall z\in\mathcal{N},(z=n \text{ or } z=n')\right\}\Big\},\end{aligned} \tag{17}$$

where $\nu^L_{\eta,n,n'}$ is the η-IA residual capacity of the link from n to n'. Let $\tau^P_{n,n'}$ be the total queuing delay of $p\left(n,n'\right)$ including queuing delay of source and destination nodes as well as all links on the path, i.e.,

$$\tau^P_{n,n'} = w_n + \sum_{l\in p(n,n')} w_l + w_{n'}. \tag{18}$$

The procedure $BinarySearch\left(\varepsilon^P_{n,n'},\eta\right)$ is to find a maximum volume of communication data size to be added on all links of $p\left(n,n'\right)$ without exceeding the delay constraint of $\varepsilon^P_{n,n'} + \tau^P_{n,n'}$. As shown in Algorithm 1, the maximal and minimal values of η-IA residual capacity of $p\left(n,n'\right)$ are initialized to be the minimal data size for links on $p\left(n,n'\right)$ with satisfied (12) and 0, respectively. We find the value by calculating the delay of the center of search space and narrow the search space to be half after each calculation.

B. IA residual capacity of wireless link

Generally a physical link may belong to multiple paths, thus the MAAD of a link from n to n', i.e., $l\left(n,n'\right)$, is defined to guarantee the MAAD of all belonged paths,

$$\varepsilon^L_{n,n'} = \min\left\{\varepsilon^P_{z,z'},\forall z,z'\in\mathcal{N}, l\left(n,n'\right)\in p(z,z')\right\}. \tag{19}$$

When it is clear from the context, we use l for short of $l\left(n,n'\right)$.

Definition 2. *The η-IA residual capacity of link $l\left(n,n'\right)$ is defined as the maximum volume of communication data size $\nu^L_{\eta,n,n'}$, from a service with the task arrival rate of η, that can be added to the link, such that the delay requirement of existing services are not violated, given by (20) and (21),*

$$\begin{aligned}\frac{\overline{X^2}R_l^2}{2F\left(R_l-\left(\lambda_l F+\nu^L_{\eta,n,n'}\cdot\eta\right)\right)} - \frac{\overline{X^2}R_l}{2F} + \frac{\nu^L_{\eta,n,n'}}{R_l}\\ \le \varepsilon^L_{n,n'} + w_l,\end{aligned} \tag{20}$$

$$R_l/F - \left(\lambda_l + \nu^L_{\eta,n,n'}\cdot\eta/F\right) > 0. \tag{21}$$

Obviously, (20) is equivalent to a $\varepsilon^L_{n,n'} + w_l$ constrained link delay in (4) with additional communication load of $\nu^L_{\eta,n,n'}$ for a service with the task arrival rate of η. Considering the additional transmitting delay of $\nu^L_{\eta,n,n'}$ in (20) is necessary in case the newly arrived service is a part of an existing service, i.e., placing a new component for existing services. Eq. (21) reflects the maximum link capacity constraint in (12) while considering $\nu^L_{\eta,n,n'}$. Based on above definition, we can derive $\nu^L_{\eta,n,n'}$ to support the calculation of $\nu^P_{\eta,n,n'}$ in (17).

Theorem 2. *The η-IA residual capacity of $l\left(n,n'\right)$ is*

$$\nu^L_{\eta,n,n'} = \max\left\{\Phi_l - \sqrt{\Omega_l}, 0\right\}, \tag{22}$$

where

$$\Phi_l := \frac{\left(\varepsilon^L_{n,n'}+w_l\right)R_l}{2} + \frac{\overline{X^2}R_l^2}{4F} + \frac{R_l-\lambda_l F}{2\eta},$$

$$\Omega_l := \frac{\overline{X^2}R_l^3}{2F\eta} + \left(\frac{\left(\varepsilon^L_{n,n'}+w_l\right)R_l}{2} + \frac{\overline{X^2}R_l^2}{4F} - \frac{R_l-\lambda_l F}{2\eta}\right)^2.$$

Proof. See Appendix-D. □

C. IA residual capacity of edge server

Similarly with $\varepsilon^L_{n,n'}$, the MAAD of edge node n can be formulated as

$$\varepsilon^N_n \le \min\left\{\varepsilon^P_{z,z'},\forall z,z'\in\mathcal{N},(n=z \text{ or } n=z')\right\}. \tag{23}$$

Note that only the delay of start and end nodes is considered in the total delay of a path. Therefore, ε^N_n is only influenced by the paths starting from or ending at n.

Definition 3. *The η-IA residual capacity of node n is defined as the maximum volume of computation resource $\nu^N_{\eta,n}$, from a service with the task arrival rate of η, that can be added to the node, such that the delay requirements of existing services are not violated, given by (24) and (25),*

$$\frac{1}{C_n - \left(\lambda_n + \nu^N_{\eta,n}\cdot\eta\right)} + \frac{\nu^N_{\eta,n}}{C_n} \le \varepsilon^N_n + w_n, \tag{24}$$

$$C_n - \left(\lambda_n + \nu^N_{\eta,n}\cdot\eta\right) > 0. \tag{25}$$

Similarly, (24) and (25) are variants of (2) and (11) respectively.

Theorem 3. *The η-IA residual capacity of node n is*

$$\nu^N_{\eta,n} = \max\left\{\Phi_n - \sqrt{\Omega_n}, 0\right\}, \tag{26}$$

where

$$\Phi_n := \frac{\left(\varepsilon^N_n+w_n\right)C_n}{2} + \frac{C_n-\lambda_n}{2\eta},$$

$$\Omega_n := \frac{C_n V}{\eta} + \left(\frac{\left(\varepsilon^N_n+w_n\right)C_n}{2} - \frac{C_n-\lambda_n}{2\eta}\right)^2.$$

Proof. See Appendix-E. □

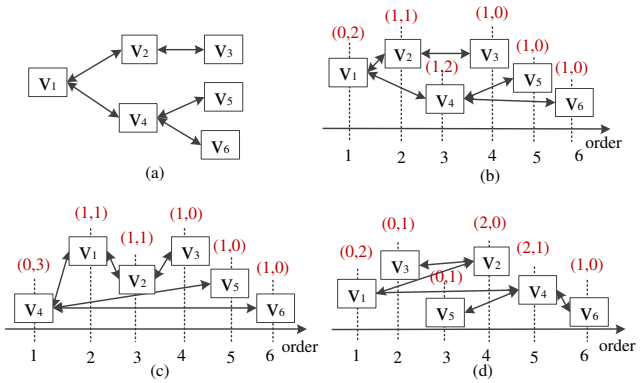

Fig. 2: Comparison of the number of connectivity constraints and induced connectivity constraints of different component ranking methods. (a) Example service tree; (b) Level traversal; (c) Page rank; (d) Random rank

V. ACO Algorithm Based on LT Component Ranking and IA Dynamic Pruning

The near-optimal solution of IA service placement can be achieved using ACO on a constructed search graph. However, directly searching the fixed graph takes a long time to converge and may break off the delay constraints. Thus, we propose a new component ranking method to accelerate ant search and an IA pruning procedure to dynamically prune the search graph, making sure the delay constraints of existing services will not be violated. The whole algorithm is described in Algorithm 2.

A. Component ranking based on LT

Component ranking, based on which components are assigned in order, is of vital importance for improving the possibility of successful placement due to the influences among components in aspects of not only delay requirements but also resource constraints. Assignment of a component should take both of its resource requirement and the connectivity with other already assigned components into account, the later helps to stimulate edge placement. As services are organized in tree structure, we propose to use the order of components in the LT of a service tree for component ranking **(line 3 in Algorithm 2)**. We define the **connectivity constraint** while assigning a component to describe the assignment of an edge between one component to be assigned with another one already assigned. The **induced connectivity constraint** is defined as the assignment of an edge between one component to be assigned with another one that is not assigned.

Level traversal order signifies that components are assigned from the root of a service to its leafs. In this way, there is a high probability that edges connected to a component are distributed to the connectivity constraints of different components and surely the edges will become both connectivity constraints and induced connectivity constraints (not either connectivity constraints or induced connectivity constraints) of itself. Thus, both the maximum number of connectivity constraints and induced connectivity constraints of components are reduced. Therefore it becomes easier to find assignment results.

For example in Fig. 2, the assignment orders of components in service (a) using different methods are depicted in (b), (c), and (d). For each component, the number of connectivity constraints and induced connectivity constraints to others is marked in the parentheses above the component. In the proposed component ranking method based on LT, components are assigned in the order of their depths in the service tree. By the definition of connectivity constraints and induced connectivity constraint, the edge between v_1 and v_4 is a connectivity constraint for component v_4 because the assignment of v_1 is completed while assigning v_4. The edges $\mathrm{v}_4-\mathrm{v}_5$ and $\mathrm{v}_4-\mathrm{v}_6$ are induced connectivity constraints for v_4. Therefore, the maximum number of connectivity constraints and induced connectivity constraints of components in the service in (a) is $(1,2)$ while using LT order. The results of Page rank [52], [53] and a random rank are shown in (c) and (d), whose maximum numbers of connectivity constraints and induced connectivity constraints of components are $(1,3)$ and $(2,2)$ respectively. Apparently component ranking based on LT can alleviate the influence among components.

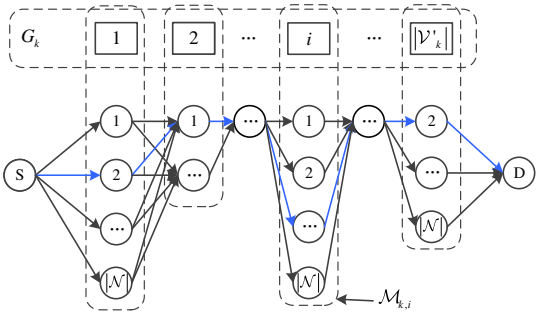

Fig. 3: ACO search graph constructing.

B. Search graph constructing

For each service G_k, a search graph $\mathcal{G}_k$ is constructed for ACO **(lines 4 - 9 in Algorithm 2)** as shown in Fig. 3. Firstly, the candidate domain of each component indexed by i in $\mathcal{V}_k$, i.e., $\mathcal{M}_{k,i}$, is created by finding the edge nodes with satisfied required computation resources **(line 2)**. Then, we fully connect the candidate domains sequentially according to component ranking results. Two auxiliary vertice S and D are introduced as the source and destination of ant search. The path length calculated in physical network of any pair of connected nodes in $\mathcal{G}_k$ is set as the edge weight. Weights of edges beside S and D in $\mathcal{G}_k$ are set to be 1. Apparently, the assignment of components is determined if a trail from S to D is found and any edge is placed to the shortest physical path between the placed edge nodes of its start and end components.

C. Single ant searching

The heuristic factor of an edge in $\mathcal{G}_k$ is formulated as $\sigma_{i,i'}^{n,n'} = 1/\text{the path length of } p(n,n')$ for $n \in \mathcal{M}_{k,i}$ and $n' \in \mathcal{M}_{k,i'}$ **(line 11 in Algorithm 2)**. The pheromone trail between any two vertice $n \in \mathcal{M}_{k,i}$ and $n' \in \mathcal{M}_{k,i'}$ in $\mathcal{G}_k$, denoted by $\gamma_{i,i'}^{n,n'}$, refers to the desirability of placing component i' onto edge node n' after component i is placed to node n. The initial value of pheromone trail on each edge is identical when constructing $\mathcal{G}_k$. After each iteration, the pheromone on each trail evaporates as $\gamma_{i,i'}^{n,n'} = (1-\xi)\,\gamma_{i,i'}^{n,n'}$ where ξ is the evaporation coefficient **(line 15 in Algorithm 2)**. Meanwhile, any ant lays the pheromone on the edges along

Algorithm 2: ACO for Online Service Placement

Input: G_k, $\mathcal{N}$, $\mathcal{L}$
Output: $\boldsymbol{a}, \boldsymbol{\varphi}$
1 **for** $k \in \mathcal{K}$ **do**
2 Find $\mathcal{M}_{k,i}, \forall i \in \mathcal{V}_k$;
3 Rank components in $\mathcal{V}_k$ using LT as $\mathcal{V'}_k$;
4 **Construct ACO search graph $\mathcal{G}_k$:**
5 Add two auxiliary nodes S and D;
6 Link S to $n, \forall n \in \mathcal{M}_{k,\mathcal{V'}_{k,1}}$ with weight 1;
7 **for** $i \in \mathcal{V'}_k$ *and* $i \neq \mathcal{V'}_{k,|\mathcal{V'}_k|}$ **do**
8 Link $n, \forall n \in \mathcal{M}_{k,i}$ to $n', \forall n' \in \mathcal{M}_{k,i+1}$ and set its weight as the path length (in hops) of $l\,(n, n')$;
9 Link $n, \forall n \in \mathcal{M}_{k,\mathcal{V'}_{k,|\mathcal{V'}_k|}}$ to D with weight 1;
10 **ACO search:**
11 Compute the heuristic factor for each edge in $\mathcal{G}_k$;
12 $iter \leftarrow 1$;
13 **for** $iter \leq N_{iter}^{\max}$ **do**
14 $ant \leftarrow 1$;
15 Evaporate pheromone;
16 **for** $ant \leq N^{ant}$ **do**
17 $\mathcal{G}_k(0) \leftarrow \mathcal{G}_k$;
18 Initialize its start position as S, $i \leftarrow 0$;
19 **for** $i \leq |\mathcal{V}_k| + 1$ **do**
20 Choose next position using ϵ-greedy algorithm;
21 Update $v_{\eta_k,n}^N$, $v_{\eta_k,n,n'}^L$, and $v_{\eta_k,n,n'}^P$, $\forall n, n' \in \mathcal{N}$ according to **(17)** and **Theorems 2 and 3**;
22 $\mathcal{G}_k(i+1) \leftarrow$ IA_pruning$(\mathcal{G}_k(i))$;
23 Update pheromone trails;
24 Extract $\boldsymbol{a}, \boldsymbol{\varphi}$ for G_k from the travel path of the ant;
25 Calculate $U\,(\boldsymbol{a}, \boldsymbol{\varphi})$;
26 Save the $\boldsymbol{a}, \boldsymbol{\varphi}$ with the maximum $U\,(\boldsymbol{a}, \boldsymbol{\varphi})$;
27 $iter \leftarrow iter + 1$, $ant \leftarrow ant + 1$.

its trail as $\gamma_{i,i'}^{n,n'} = \gamma_{i,i'}^{n,n'} + \xi \cdot \Delta$ where Δ is the amount of pheromone laid by the ant **(line 23 in Algorithm 2)**. Here, we use the utility function for measuring the effectiveness of the search solution of an ant and define Δ as $\Delta = U\,(\boldsymbol{a}, \boldsymbol{\varphi})$. Higher utility leads to higher pheromone laid by an ant.

Based on the heuristic factor and pheromone factor, the transition probability from $n \in \mathcal{M}_{k,i}$ to $n' \in \mathcal{M}_{k,i'}$ is

$$q_{i,i'}^{n,n'} = \frac{\left(\gamma_{i,i'}^{n,n'}\right)^{\alpha}\left(\sigma_{i,i'}^{n,n'}\right)^{\delta}}{\sum_{n'' \in \mathcal{M}_{k,i'}}\left(\gamma_{i,i'}^{n,n''}\right)^{\alpha}\left(\sigma_{i,i'}^{n,n''}\right)^{\delta}} \tag{27}$$

where α, δ reflect the importance of pheromone factor and heuristic factor respectively. We use ϵ-greedy algorithm for ants to choose next position **(line 20 in Algorithm 2)**. It specifies that each ant selects vertex along the edge that has the largest transition probability in (27) with the probability of ϵ. Otherwise, with the probability of $1-\epsilon$, an ant randomly choose the next position in $\mathcal{G}_k$ to prevent trapping in local optimum of ACO.

D. Dynamic pruning based on IA residual capacity

The IA_Pruning procedure is described in Algorithm 3 to tackle the interference among components as well as edges **(line 22 in Algorithm 2)**. The edge between components h and h' is represented by $j(h, h')$. Every time an ant moves a step, two conditions are checked on the rest of search graph **(lines 3 and 7 in Algorithm 3)**. (1) If the IA residual capacities of the rest candidate nodes and paths are enough for existing services? (2) If the rest candidate nodes satisfy the delay requirements of the task? The candidate nodes that violate

Algorithm 3: IA_Pruning

Input: i, $\mathcal{G}_k(i)$
Output: $\mathcal{G}_k(i+1)$
1 **for** $h \in \mathcal{V'}_k$ *and* $h > i$ **do**
2 **for** $n \in \mathcal{M}_{k,h}$ **do**
3 **if** $c_{k,h} > v_{\eta_k,n}^N$ *or (13) is not satisfied when* $a_{k,h,n} = 1$ **then**
4 Delete n from $\mathcal{G}_k(i)$;
5 **for** $h' \in \mathcal{V'}_k$ *and* $h' \leq i$ **do**
6 **for** $n' \in \mathcal{M}_{k,h'}$ **do**
7 **if** $b_{k,j(h,h')} > v_{\eta_k,n,n'}^P$ *or (13) is not satisfied when* $\phi_{k,j(h,h'),l(n,n')} = 1$ **then**
8 Delete n from $\mathcal{G}_k(i)$;
9 $\mathcal{G}_k(i+1) \leftarrow \mathcal{G}_k(i)$.

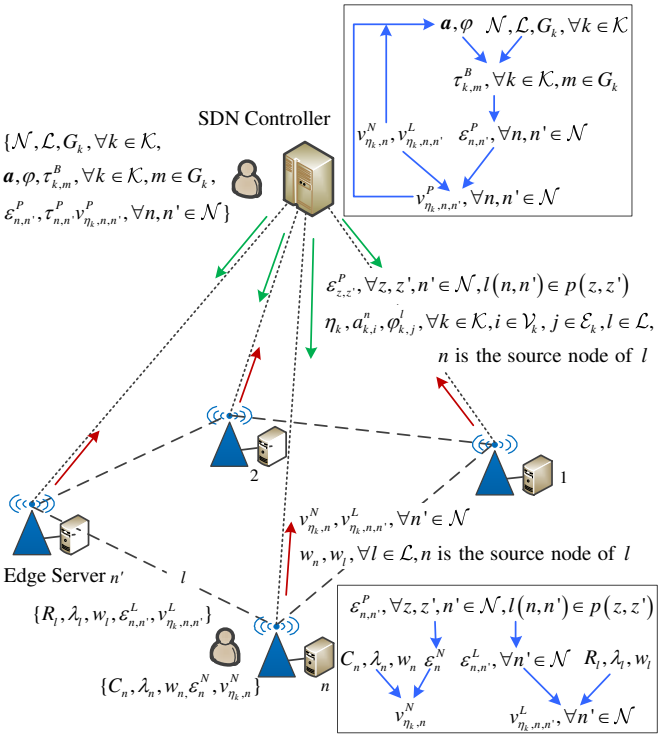

Fig. 4: Distributed framework for computing IA residual capacity

either above conditions are removed temporarily for current ant resulting in a new search graph $\mathcal{G}_k(i+1)$ for the search of $(i+1)$th step **(lines 4 and 8 in Algorithm 3)**. Any ant should move according to its dynamically pruned graph and once a feasible path is found, the pheromone factors of edges along the trail will be updated [17].

E. Overall distributed framework

For ease of accelerating calculation of IA residual capacity, the distributed framework is proposed as in Fig. 4. The SDN controller holds the full picture of physical infrastructure and is responsible for executing ACO. The calculation of queuing delay and IA residual capacities of nodes and links are decoupled from SDN controller to respective physical nodes. The computing results are uploaded to the controller for calculating IA residual capacities of paths.

F. Complexity analysis

For any service k with $|\mathcal{V}_k|$ components, the time complexity of LT ranking is $O\,(|\mathcal{V}_k|)$. The complexity of LT is the

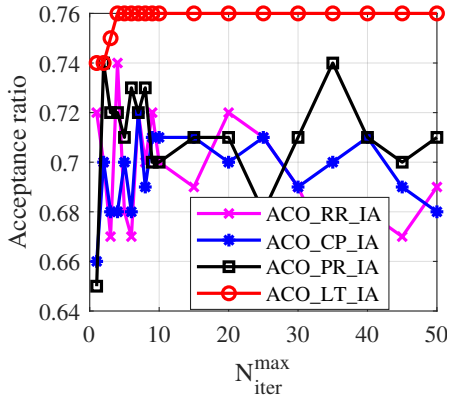

Fig. 5: Acceptance ratio of ACO using different component ranking methods

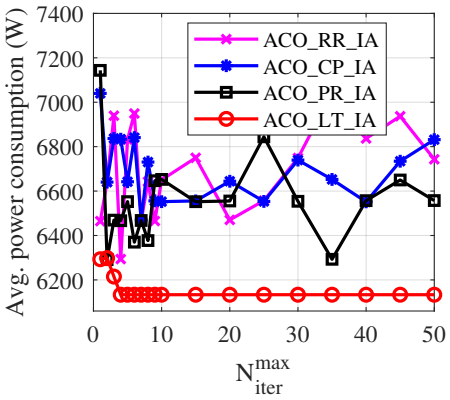

Fig. 6: Average power consumption of ACO using different component ranking methods

same as random ranking (RR), and higher than computation resource ranking (CR), i.e., $O\left(|\mathcal{V}_k|\log|\mathcal{V}_k|\right)$ using quick sort, and Page rank (PR) [54], i.e., $O\left(|\mathcal{V}_k|^2\cdot T\right)$ where T denotes the number of iterations taking $T=\log|\mathcal{V}_k|$ in general.

The ACO takes $O\left(|\mathcal{V}_k|\cdot N^{ant}\cdot N_{iter}^{\max}\right)$ steps in total for searching optimal path of each service G_k. The IA_Pruning costs $O\left(|\mathcal{V}_k|^2\cdot|\mathcal{N}|^2\right)$ steps where $|\mathcal{N}|$ indicates the maximum number of candidate nodes in $\mathcal{M}_{k,i}$. It takes $O\left(|\mathcal{N}|^2\right)$ steps to find the shortest paths using Dijkstra. Overall, the time complexity of the proposed ACO based on LT component ranking and IA dynamic pruning is $O\left(|\mathcal{V}_k|+|\mathcal{V}_k|^3\cdot|\mathcal{N}|^2\cdot N^{ant}\cdot N_{iter}^{\max}+|\mathcal{N}|^2\right)=O\left(|\mathcal{V}_k|^3\cdot|\mathcal{N}|^2\right)$.

The existing method based on general resource capacity (GRC) [55] costs $O\left(|\mathcal{V}_k|\log|\mathcal{V}_k|\right)$ steps for component ranking, $O\left(|\mathcal{V}_k|\,|\mathcal{N}|^2\right)$ steps for calculating GRC for all components and $O\left(|\mathcal{N}|^2\right)$ steps to find the shortest paths. Therefore, its total complexity is $O\left(|\mathcal{V}_k|\log|\mathcal{V}_k|+|\mathcal{V}_k|\,|\mathcal{N}|^2+|\mathcal{N}|^2\right)=O\left(|\mathcal{V}_k|\,|\mathcal{N}|^2\right)$. The existing method based on Page rank costs $O\left(|\mathcal{N}|^2\cdot\log|\mathcal{N}|\right)$, $O\left(|\mathcal{N}|\log|\mathcal{N}|\right)$, and $O\left(|\mathcal{N}|^2\right)$ steps for calculating Page rank values, ranking these values in the decreasing order and finding the shortest paths, respectively. Its time complexity is then $O\left(|\mathcal{N}|^2\cdot\log|\mathcal{N}|\right)$. However, the method based on GRC neglects the influence among online services and the method based on Page rank further ignores the features of services such as service architecture and resource requirements. The proposed ACO based on LT component ranking and IA dynamic pruning takes the interference and features of services into account with **linearly additional time complexity with respect to** $O\left(|\mathcal{N}|^2\right)$ based on the fact that there exists $|\mathcal{N}|>>|\mathcal{V}_k|$ in practical, while **the performance has been improved remarkably** as described in Section VI-C.

VI. Simulation Results

A. Parameter settings and comparative approaches

In the simulation, we generate 34 SBSs, each of which is equipped with an edge server over a 200×200 m^2 area according to HPPP process with the density of 10^{-3} [30]. The computation capacity of each edge server is 10 Gigacycles/s. The transmission power of an SBS is 30dBm. The path loss of wireless links over millimeter wave is $157.4+32\log_{10}(dis)$ where dis is the link length in km. The power density of white noise is $N_0=-174.0$ dBm/Hz. The spectrum width for each wireless link is 20MHz. Services arrive according to Poisson distribution. The computation requirements 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 in [1, 5] uniformly and the average duration of services is 10 according to exponential distribution. The delay requirements of services are uniformly between [0.1, 0.5]s. Moreover, we set F = 20 Bytes and $\overline{X^2}=1.137\,(\mu s)^2$. For ACO, there are $\epsilon=0.6$, $\xi=0.1$, $\alpha=2$, $\delta=3$, and $N^{ant}=5$. In all simulations, totally 100 services are generated for the placement.

We use power consumption as the objective function by minimizing the total power consumption P of services,

$$P=\sum_{l\in\mathcal{L}}p_{idle}^{trans}+\sum_{k\in\mathcal{K}}T_k\eta_k\left[\sum_{i\in\mathcal{V}_k}c_{k,i}\cdot p^{comp}+\sum_{j\in\mathcal{E}_k}\sum_{l\in\mathcal{L}}\varphi_{k,j,l}\cdot b_{k,j}/R_l\cdot p_{\max}^{trans}\right], \tag{28}$$

where $p^{comp}=8.2\times10^{-9}$ W/cycle [30] denotes the power consumption of per CPU cycle and $p_{idle}^{trans}=14.9$ W [56], [57] is the power of idle SBSs without data transmission. The power consumption is equivalent to the utility function in (5) with $\zeta=-p^{comp}$ and $\phi=-p_{\max}^{trans}/R_l$.

The proposed ACO algorithm based on LT component ranking and IA dynamic pruning (labeled by ACO_LT_IA) is compared with following approaches:

1) ACO_PR_IA: Use Page rank instead of LT for component ranking in the proposed ACO framework;
2) ACO_CR_IA: Rank components in the decreasing order of their computation resource requirements instead of LT in the proposed ACO framework;
3) ACO_RR_IA: Replace LT with random ranking order in the proposed algorithm;
4) ACO_LT: The proposed ACO algorithm based on LT component ranking but without IA dynamic pruning;
5) Baseline_GRC [55]: 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 sum of bandwidth capacities on its neighbor links, and then assign edges to the shortest paths.

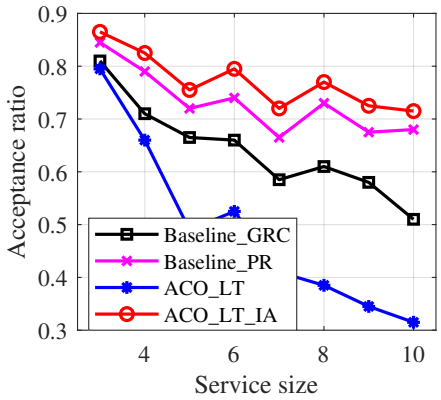

Fig. 7: Acceptance ratio of different approaches under different service sizes

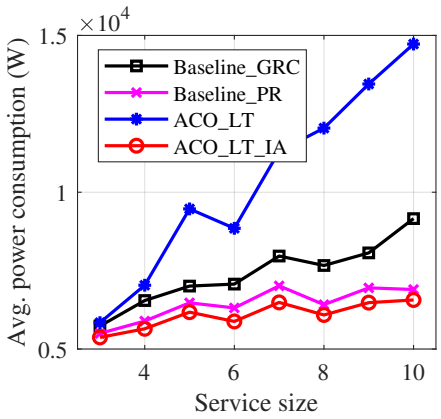

Fig. 8: Average power consumption of different approaches under different service sizes

6) Baseline_PR [58]: First assign all components in a service using Page rank, and then assign edges to the shortest paths.

We compare the proposed algorithm with others on the average power consumption defined as the ratio of total consuming power in (28) over all accepted services and the acceptance ratio which is defined as the ratio of successfully placed services over all services.

B. Performance of component ranking based on LT

Varying the number of components in a service (i.e., service size) uniformly in [2, 8], we evaluate our proposed ACO_LT_IA in comparison with other component ranking methods as shown in Figs. 5 and 6. It can be observed that ACO_LT_IA converges faster than other methods because LT promotes each ant finding feasible placement path due to less (induced) connectivity constraints. There is a higher probability of failed search in other three methods. Specifically, Page rank prefers to place a component that requires more resources and has more connections with other components in priority. However, a large number of induced connectivity constraints is generated, i.e., the maximum number of induced connectivity constraints is the same as the largest component degree. ACO_CR_IA and ACO_RR_IA produce a high degree of randomness regardless the edges among components, resulting in lower convergence rate. Moreover, higher service acceptance ratio and lower average power consumption are achieved in ACO_LT_IA.

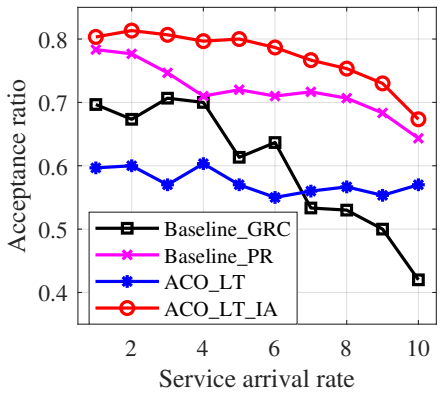

Fig. 9: Acceptance ratio of different approaches under different service arrival rates

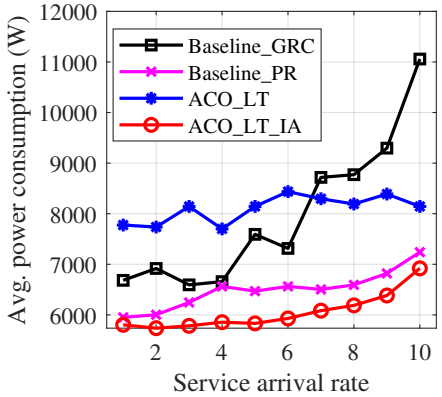

Fig. 10: Average power consumption of different approaches under different service arrival rates

C. Performance of ACO based on IA dynamic pruning

The proposed ACO_LT_IA is also evaluated compared with other service placement approaches when the service arrival rate is 1 (not the arrival rate of tasks for a specific service). The acceptance ratio and average power consumption of different approaches under different service sizes are demonstrated in Figs. 7 and 8. The acceptance ratio decreases with the increase of service size on account that complex services are hard to place. However, the average power consumption grows up because the power consumed by each service is higher for large services. Similarly, ACO_LT_IA outperforms others for consuming lower average power per service and accepting more services. The superiority of ACO_LT_IA can be further validated while varying system load. When the arrival rate of services varies from 1 to 10 and the service size distributes uniformly in [2,8], the acceptance ratio drops down and the average power consumption grows up as shown in Figs. 9 and 10. For all arrival rates, ACO_LT_IA achieves the highest acceptance ratio and lowest average power consumption. Overall, the proposed ACO_LT_IA outperforms all the other comparative approaches in a wide range of service sizes and service arrival rates.

VII. Application

In this section, we use the multi-class image classification as an example to enhance the motivation of IA investigation and show its practical effectiveness.

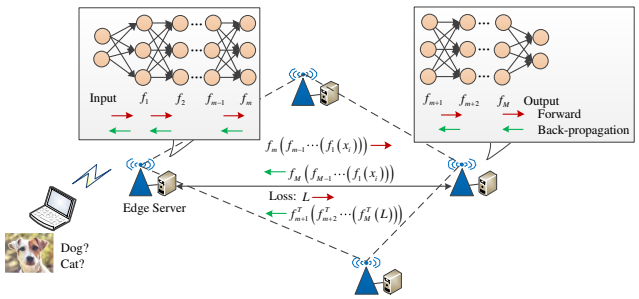

Fig. 11: Illustration of the communication between two components in a multi-class classification service with M layers where f_m and $f_m(\cdot)$ denote the outputs of the mth layer and $f_m^T(\cdot)$ is the gradients of the mth layer

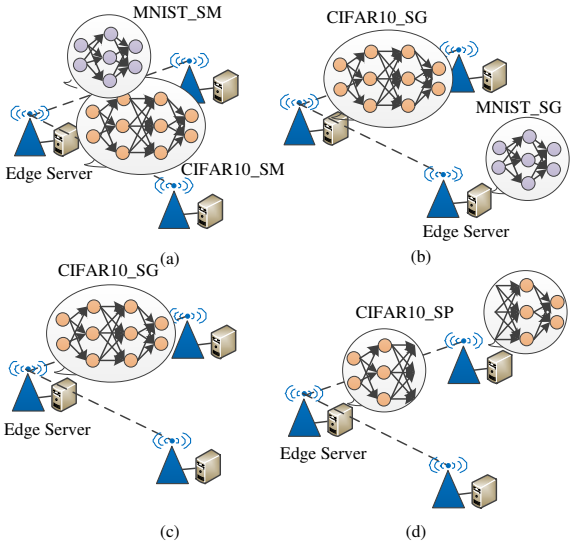

Fig. 12: Illustration of the assignment results of two services; (a) Two services are placed to the same edge server simultaneously (SM); (b) Each service is assigned to a single edge server (SG); (c) Assign a large service to one server; (d) Different components in a service are assigned to different servers separately (SP)

A. Multi-class image classification based on IA mechanism

Given a training dataset $\Lambda = \{(x_i, y_i)\}_{i=1}^{X}$ with an unknown distribution over instances x_i and labels $y_i \in \{1, 2, \cdots, Y\}$, where X and Y are the number of data samples in Λ and total number of possible output classes, respectively. The goal of multi-class classification [59] is to learn a predictor $f : x_i \rightarrow y_i, \forall i$ so as to minimize the expected loss for a random instance and label, i.e., $\min E_{(x_i,y_i)}[L(y_i, f(x_i))]$. The loss function L means for label y_i and prediction vector $f(x_i)$, $L(y_i, f(x_i))$ is the loss incurred for predicting $f(x_i)$ when the true label is y_i. Generally, the cross-entropy between y_i and $f(x_i)$ is used as the loss function of multi-class classification. The predictor f is represented by a neural network with multiple layers, which is trained using machine learning to minimize L. The Stochastic Gradient Descent (SGD) is used for updating parameters in f. Thus, the training of f consists of two steps in each iteration, the forward process for predicting y_i of input x_i and the back-propagation for parameter updating.

Allowing the placement of a service to more than one server is beneficial for implementing artificial intelligence services to edge servers, whose resource (e.g., computation and memory) capacity is limited compared with a central cloud. For an AI service like image classification, its neural network can be split into multiple components, each of which contains several continuous neural layers [60], [61]. Communication data among the components includes the set of parameters that connect different components in the forward process and the gradients during the back-propagation as demonstrated in Fig. 11.

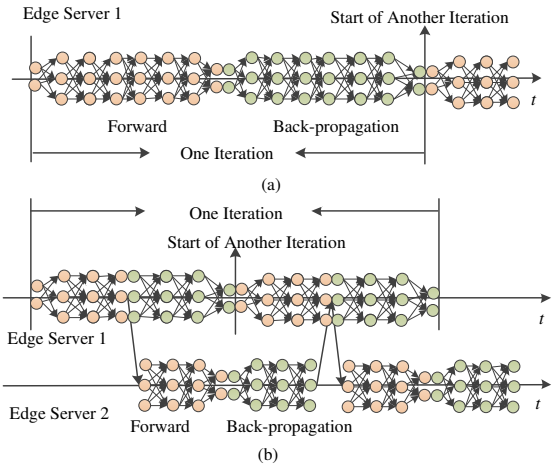

Fig. 13: Illustration of one iteration in multi-class classification; (a) The service is placed as in Fig. 12 (c); (b) The service is placed as in Fig. 12 (d)

Assume there are two classification services and three connected edge servers as shown in Fig. 12, the benefits of IA mechanism are twofolds:

1) Load balancing: Assume the computation capacity of each server is enough for holding two services, traditional approaches tend to assign both of the two services to the same server with larger GRC [55] or Page rank [58] since the server is connected to other two servers, as depicted in Fig. 12 (a). However, the IA mechanism prefers to assign the two services to different servers to avoid their interference when the communication delay is tolerable as shown in Fig. 12 (b);
2) Parallel computing: Traditional approaches incline to assign all components to the same server to reduce communication cost and delay as shown in Fig. 12 (c). However, the IA mechanism would like to distribute its components among servers to reduce computation delay leveraging additional communication resources as depicted in Fig. 12 (d). Surprisingly, the parallel computing resulting from distributed placement of a service can further accelerate service execution.

Illustration of the effectiveness of parallel computing is depicted in Fig. 13. Even though the time for executing one iteration of a multi-class classification service on two edge servers is longer than using only one server due to additional data transmission, the parallel computing allows iterations overlapping with each other, thus the neural network can be trained faster.

B. Experimental results

In this section, experiments are conducted to validate the effectiveness of IA mechanism to strengthen our motivation of IA service placement. The MNIST and CIFAR10 [62]

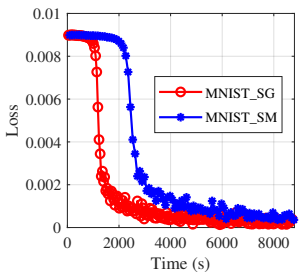

Fig. 14: Loss of MNIST for different assignment results

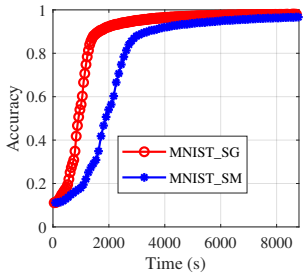

Fig. 15: Accuracy of MNIST for different assignment results

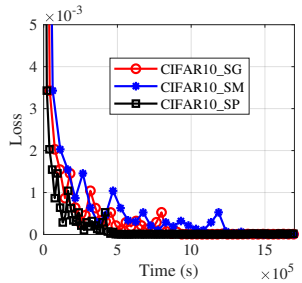

Fig. 16: Loss of CIFAR10 for different assignment results

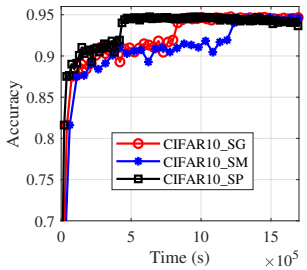

Fig. 17: Accuracy of CIFAR10 for different assignment results

are used as the datasets for image classification with 10 classes. Personal computers equipped with Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz core are used as the edge servers with 100 Mbp wireless bandwidth capacity. The classic neural networks of LeNet5 [63] and ResNet34 [64] are applied for MNIST and CIFAR10, respectively. Other configurations for SGD can refer to [65]. We divide ResNet34 into two components, each of which has three layers. The data size that is transmitted between the two components is 4,194,304 parameters, which is equivalent to 134,217,728 bits. Using (4), the communication time between two components is 0.2s.

The assignment results of the two services are as in Fig. 12, where "SG" means each services is assigned to a single server, "SM" indicates two services are assigned to the same server simultaneously and "SP" represents the case when different components of a service (CIFAR10 in our experiment) is assigned to different servers.

The experimental results are shown in Figs. 14 - 17, where the training loss and the test accuracy (i.e., the ratio of successful prediction using the trained neural network on a separated test dataset) are shown. Comparing the cases in Fig. 12 (a) and (b), both MNIST_SG and CIFAR10_SG achieve better loss and accuracy than MNIST_SM and CIFAR10_SM. For the cases in Fig. 12 (c) and (d), it can be observed that CIFAR10_SP outperforms CIFAR10_SG, reflecting the effectiveness of parallel computing.

The IA mechanism tends to assign services or components in a service to different edge servers to reduce their interference, that is, the IA mechanism prefers the assignment results of (b) and (d) to (a) and (c) in Fig. 12. The induced load balancing and parallel computing is effective for improving the performance of services (i.e., loss and accuracy). Therefore, the investigation of IA mechanism is not trivial in practical.

VIII. Conclusion

In this paper, we have investigated the interference-aware online multi-component service placement in edge cloud networks. The services are modeled using trees and the total executing delay of services is formulated based on M/G/1 and M/M/1 queuing systems. By analyzing the delay formulation, the IA residual capacities of edge servers, wireless links, and physical paths are derived theoretically. We prove that the multi-component service placement problem is NP-hard, thus the ACO is used to obtain the near-optimal solution. The component ranking based on level traversal and the interference-aware dynamic pruning are proposed for ACO to achieve faster convergence and avoid the interference among services. The simulation results show that the proposed algorithm outperform comparative approaches in terms of the convergence time and the acceptance ratio of services. In addition, the experiments on AI application of multi-class classification is conducted to further validate the effectiveness of the interference-aware mechanism. In the future, more experiments on a larger range of AI applications will be conducted using our proposed algorithms.

Acknowledgment

This work was supported in part by National Key R&D Program of China (2018YFE0206800), National Natural Science Foundation of China (61775033), Chongqing Municipal Education Commission (KJQN201900647), Open Fund of IPOC (BUPT) under Grant IPOC2019A007, and Zhejiang Lab under Grant NO. 2019LE0AB01.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, pp. 50–58, 04 2010.

[2] S. Bhardwaj, L. Jain, and S. Jain, "Cloud computing : a study of infrastructure as a service (iaas)," *International Journal of Information Technology and Web Engineering*, vol. 2, no. 1, pp. 60–63, 2010.

[3] J. Wu, L. Ping, X. Ge, Y. Wang, and J. Fu, "Cloud storage as the infrastructure of cloud computing," in *2010 International Conference on Intelligent Computing and Cognitive Informatics*. IEEE, 2010, pp. 380–383.

[4] *AI studio*. [Online]. Available: https://aistudio.baidu.com/aistudio/index

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

[6] 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.

[7] 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.

[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] 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.

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

[12] 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.

[13] 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.

[14] P. Han, Y. Liu, and L. Guo, "Interference-aware task assignment in edge cloud-enhanced 5G fiber-wireless access networks," in *Asia Communications and Photonics Conference (ACP)*, 2019, pp. 1–3.

[15] F. Xu, F. Liu, and H. Jin, "Heterogeneity and interference-aware virtual machine provisioning for predictable performance in the cloud," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2470–2483, 2016.

[16] T. Stützle and H. H. Hoos, "Max–min ant system," *Future generation computer systems*, vol. 16, no. 8, pp. 889–914, 2000.

[17] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, "VNE-AC: Virtual network embedding algorithm based on ant colony metaheuristic," in *IEEE ICC*, 2011, pp. 1–6.

[18] 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.

[19] 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.

[20] S. Yang, F. L. iand Meng 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] X. Guo, R. Singh, T. Zhao, and Z. Niu, "Achieving optimal power-delay tradeoff in edge cloud systems," in *IEEE ICC*, 2016, pp. 1–6.

[26] Z. Zhang, F. R. Yu, F. Fu, Q. Yanz, and Z. Wang, "Joint offloading and resource allocation in mobile edge computing systems: An actor-critic approach," in *IEEE global communications conference (GLOBECOM)*, 2018, pp. 1–6.

[27] S. Ko, K. Han, and K. Huang, "Wireless networks for mobile edge computing: Spatial modeling and latency analysis," *IEEE Transactions on Wireless Communications*, vol. 17, no. 8, pp. 5225–5240, 2018.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] S. Guo, D. Wu, H. Zhang, and D. Yuan, "Queueing network model and average delay analysis for mobile edge computing," in *2018 Workshop on Computing, Networking and Communications (CNC)*, 2018, pp. 172–176.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *Performance Evaluation Review*, vol. 40, no. 4, pp. 23–32, 2013.

[40] W. Zhang, Y. Wen, and D. Wu, "Collaborative task execution in mobile cloud computing under a stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 14, pp. 81–93, 01 2015.

[41] S. Bi, L. Huang, and Y. A. Zhang, "Joint optimization of service caching placement and computation offloading in mobile edge computing systems," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2020.

[42] Y. Zhang, H. Liu, L. Jiao, and X. Fu, "To offload or not to offload: An efficient code partition algorithm for mobile cloud computing," in *2012 IEEE 1st International Conference on Cloud Networking (CLOUDNET)*, 2012, pp. 80–86.

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

[44] Y. Wu, J. Shi, K. Ni, L. Qian, W. Zhu, Z. Shi, and L. Meng, "Secrecy-based delay-aware computation offloading via mobile edge computing for internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4201–4213, 2019.

[45] P. Zhao, H. Tian, and B. Fan, "Partial critical path based greedy offloading in small cell cloud," in *IEEE 84th Vehicular Technology Conference (VTC-Fall)*, 2016, pp. 1–5.

[46] M. A. Aguilar, R. Leupers, G. Ascheid, and L. G. Murillo, "Automatic parallelization and accelerator offloading for embedded applications on heterogeneous mpsocs," in *ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2016, pp. 1–6.

[47] 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.

[48] 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.

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

[50] 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.

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

[52] M. Diligenti, M. Gori, and M. Maggini, "A unified probabilistic framework for web page scoring systems," *IEEE Transactions on knowledge and data engineering*, vol. 16, no. 1, pp. 4–16, 2014.

[53] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: bringing order to the web," *technical report, Stanford digital library technologies project*, 1998.

[54] L. Page, "The pagerank citation ranking: Bringing order to the web," *Stanford InfoLab*, pp. 1–14, 01 1999.

[55] 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.

[56] A. Al-Shuwaili and A. Lawey, "Achieving low-latency mobile edge computing by uplink and downlink decoupled access in hetnets," *ArXiv preprint*, vol. abs/1809.04717, 2018.

[57] 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 *2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall)*, 2015, pp. 1–5.

[58] 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.

[59] M. Lapin, M. Hein, and B. Schiele, "Analysis and optimization of loss functions for multiclass, top-k, and multilabel classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 7, pp. 1533–1554, 2018.

[60] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," *Journal of Network and Computer Applications*, vol. 116, pp. 1–8, 08 2018.

[61] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," *ArXiv preprint*, vol. abs/1812.00564, 2018.

[62] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10," *(canadian institute for advanced research)*, 2009. [Online]. Available: http://www.cs.toronto.edu/ kriz/cifar.html.

[63] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, p. 2278–2324, 1998.

[64] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[65] H. Chen, Y. Wang, C. Xu, Z. Yang, C. Liu, B. Shi, C. Xu, and Q. Tian, "Data-free learning of student networks," in *International Conference on Computer Vision (ICCV)*, 10 2019, pp. 3513–3521.

[66] S. Shimizu, K. Yamaguchi, and S. Masuda, "A maximum edge-weight clique extraction algorithm based on branch-and-bound," *ArXiv preprint*, 2018.

[67] E. M. Macambira and C. C. de Souza, "The edge-weighted clique problem: Valid inequalities, facets and polyhedral computations," *European Journal of Operational Research*, vol. 123, no. 2, pp. 346–371, 2000.

APPENDIX

A. Proof of ***Lemma 1***

Proof. We use a special case where all services in $\mathcal{K}$ are homogeneous and each has one component with computation resource requirement of $\overline{c}$ for the proof. The arrival rate and delay requirement of every service are $\overline{\eta}$ and $\overline{d}$, respectively. Thus, P1 becomes

$P2:$

$$\min\ U(\boldsymbol{a})$$

$s.t.$

$$(7),(11),(13),(14)$$

Assume each edge server can hold at most κ services in order to satisfy the resource capacity (11) and delay constraint (13), that is, $\frac{1}{C_n-\kappa\overline{\eta}\overline{c}}+\frac{\kappa\overline{c}}{C_n}<\overline{d}<\frac{1}{C_n-(\kappa+1)\overline{\eta}\overline{c}}+\frac{(\kappa+1)\overline{c}}{C_n},\forall n\in\mathcal{N}$. The problem $P2$ is equivalent to a SAT problem with $|\mathcal{N}|\begin{pmatrix}\kappa+1\\|\mathcal{K}|\end{pmatrix}+2|\mathcal{K}|$ Boolean formulas as (29) - (31), where $|\cdot|$ denote the number of elements in the set.

$$\underbrace{\overline{a_{k,1,n}}\vee\cdots\vee\overline{a_{k',1,n}}}_{\kappa+1\text{ items}},\forall n\in\mathcal{N},\ \text{all different } k,\cdots,k'\in\mathcal{K}, \tag{29}$$

$$a_{k,1,1}\vee a_{k,1,2}\vee\cdots\vee a_{k,1,|\mathcal{N}|},\forall k\in\mathcal{K}, \tag{30}$$

$$\wedge_{n\neq n'}\left(\overline{a_{k,1,n}}\vee\overline{a_{k,1,n'}}\right),\forall k\in\mathcal{K},n,n'\in\mathcal{N}. \tag{31}$$

Formula (29) taking 1 ensures that any $\kappa+1$ services/components should not be placed onto the same edge node to satisfy the delay requirements (constraint (13)) for all binary $a_{k,1,n}$ (constraint (14)) where $\overline{a_{k,i,n}}$ flips the value of $a_{k,i,n}$. Formula (30) taking 1 specifies each component should be placed to at least one edge node and (31) guarantees any pair of edge nodes can not hold a component at the same time. Thus, (30) and (31) guarantee the unsplittable component placement, that is (7) in $P2$. Consequently, endowing the outputs of Boolean formulas (29) - (31) with all 1 is equivalent to the constraints in $P2$. Thus, Lemma 1 is proved. □

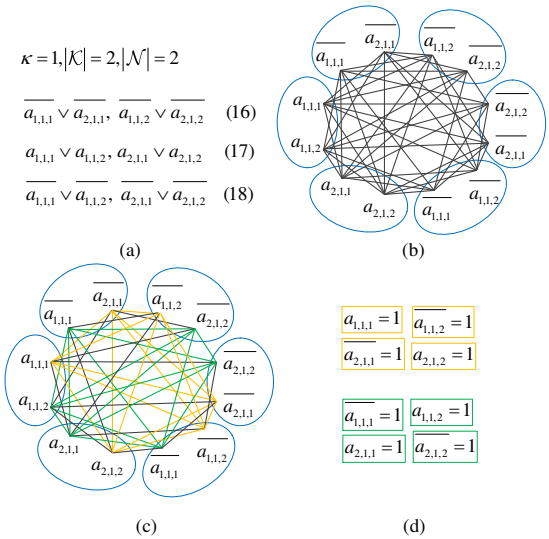

Fig. 18: Diagram of transforming SAT into MEWC. (a) Boolean formulas of placing two services on two physical nodes with $\kappa=1$; (b) Auxiliary graph $\mathcal{A}$; (c) Two MEWCs of $\mathcal{A}$, each clique is a fully-connected graph; (d) Two placement results extracted form MEWCs

B. Proof of ***Lemma 2***

Proof. Any SAT problem can be transformed into MEWC problem, as exampled in Fig. 18, through constructing an auxiliary graph $\mathcal{A}$ as follows:

1) Transform each unit (i.e., x and $\overline{x}$) in SAT into a vertex in $\mathcal{A}$. The units with OR relation in SAT are set into a subset in $\mathcal{A}$;
2) Connect vertices that locate in different subsets without conflict, i.e., x and $\overline{x}$ should not be connected;
3) Set the weights of edges that connect different units (units with and without the NOT operation are the same) with the total utility of the end nodes.
4) Find the MEWC of $\mathcal{A}$.

□

C. Proof of ***Theorem 1***

Proof. Based on Lemma 2 and the fact that the MEWC problem is NP-hard [66], [67], the problem P1 is also NP-hard. □

D. Proof of ***Theorem 2***

Proof. From (20), we have

$$\left(\varepsilon^L_{n,n'}+w_l+\frac{\overline{X^2}R_l}{2F}-\frac{\nu^L_{\eta,n,n'}}{R_l}\right)$$
$$\left(R_l-\lambda_l F-\nu^L_{\eta,n,n'}\cdot\eta\right)\geq\frac{\overline{X^2}R_l^2}{2F},$$

it follows

$$\frac{\eta}{R_l}\left(\nu^L_{\eta,n,n'}\right)^2-\left[\left(\varepsilon^L_{n,n'}+w_l+\frac{\overline{X^2}R_l}{2F}\right)\eta+\frac{R_l-\lambda_l F}{R_l}\right]\nu^L_{\eta,n,n'}+\left(\varepsilon^L_{n,n'}+w_l+\frac{\overline{X^2}R_l}{2F}\right)(R_l-\lambda_l F)\geq\frac{\overline{X^2}R_l^2}{2F},$$

which gives

$$\left(\nu^L_{\eta,n,n'}-\frac{\left(\varepsilon^L_{n,n'}+w_l\right)R_l}{2}-\frac{\overline{X^2}R_l^2}{4F}-\frac{R_l-\lambda_l F}{2\eta}\right)^2\geq\frac{\overline{X^2}R_l^3}{2F\eta}-\left(\varepsilon^L_{n,n'}+w_l+\frac{\overline{X^2}R_l}{2F}\right)\frac{R_l(R_l-\lambda_l F)}{\eta}+\left(\frac{\left(\varepsilon^L_{n,n'}+w_l\right)R_l}{2}+\frac{\overline{X^2}R_l^2}{4F}+\frac{R_l-\lambda_l F}{2\eta}\right)^2.$$

Finally, we have

$$\left(\nu^L_{\eta,n,n'}-\frac{\left(\varepsilon^L_{n,n'}+w_l\right)R_l}{2}-\frac{\overline{X^2}R_l^2}{4F}-\frac{R_l-\lambda_l R}{2\eta}\right)^2\geq\frac{\overline{X^2}R_l^3}{2F\eta}+\left(\frac{\left(\varepsilon^L_{n,n'}+w_l\right)R_l}{2}+\frac{\overline{X^2}R_l^2}{4F}-\frac{R_l-\lambda_l F}{2\eta}\right)^2.$$

With the definitions of $\Phi_l\geq 0$ and $\Omega_l\geq 0$, we obtain the quadratic inequality,

$$\left(\nu^L_{\eta,n,n'}-\Phi_l\right)^2\geq\Omega_l. \tag{32}$$

It has a positive discriminant when the equality holds. Therefore,

$$\begin{cases}\nu^L_{\eta,n,n'}\geq\Phi_l+\sqrt{\Omega_l}, \text{ if } \nu^L_{\eta,n,n'}>\Phi_l,\\ \nu^L_{\eta,n,n'}\leq\Phi_l-\sqrt{\Omega_l}, \text{ if } \nu^L_{\eta,n,n'}\leq\Phi_l.\end{cases} \tag{33}$$

From (21), we have

$$\nu^L_{\eta,n,n'}<\frac{R_l-\lambda_l F}{\eta}. \tag{34}$$

Thus, (20) and (21) are transformed into (33) and (34).

Taking insights into Φ_l, Ω_l and $(R_l-\lambda_l F)/\eta$, we have

$$\Phi_l+\sqrt{\Omega_l}-\frac{R_l-\lambda_l F}{\eta}=\Gamma_l+\sqrt{\frac{\overline{X^2}R_l^3}{2F\eta}+\Gamma_l^2}\geq 0 \tag{35}$$

and

$$\frac{R_l-\lambda_l F}{\eta}-\left(\Phi_l-\sqrt{\Omega_l}\right)=-\Gamma_l+\sqrt{\frac{\overline{X^2}R_l^3}{2F\eta}+\Gamma_l^2}\geq 0, \tag{36}$$

where $\Gamma_l:=\Phi_l-\frac{R_l-\lambda_l F}{\eta}=\frac{\left(\varepsilon^L_{n,n'}+w_l\right)R_l}{2}+\frac{\overline{X^2}R_l^2}{4F}-\frac{R_l-\lambda_l F}{2\eta}$. Combining (35) and (36) gives

$$\Phi_l-\sqrt{\Omega_l}\leq\frac{R_l-\lambda_l F}{\eta}\leq\Phi_l+\sqrt{\Omega_l}. \tag{37}$$

Therefore, we can depict the picture of $\nu^L_{\eta,n,n'}$ as in Fig. 19. Finally, we have $\nu^L_{\eta,n,n'}=\max\left\{\Phi_l-\sqrt{\Omega_l},0\right\}$. □

E. Proof of **Theorem 3**

Proof. From (24) we have

$$\left(\varepsilon^N_n+w_n-\frac{\nu^N_{\eta,n}}{C_n}\right)\left(C_n-\lambda_n-\nu^N_{\eta,n}\cdot\eta\right)\geq 1,$$

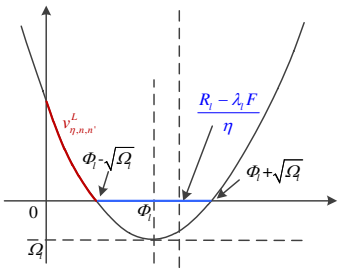

Fig. 19: The diagram of $\nu^L_{\eta,n,n'}$

which gives

$$\left(\nu^N_{\eta,n}\right)^2-\left(\left(\varepsilon^N_n+w_n\right)C_n+\frac{C_n-\lambda_n}{\eta}\right)\nu^N_{\eta,n}+\frac{C_n\left(\varepsilon^N_n+w_n\right)(C_n-\lambda_n)}{\eta}\geq\frac{C_n}{\eta}.$$

Through algebra operations, we have

$$\left(\nu^N_{\eta,n}-\frac{\left(\varepsilon^N_n+w_n\right)C_n}{2}-\frac{C_n-\lambda_n}{2\eta}\right)^2\geq\frac{C_n}{\eta}-\frac{C_n\left(\varepsilon^N_n+w_n\right)(C_n-\lambda_n)}{\eta}+\left(\frac{\left(\varepsilon^N_n+w_n\right)C_n}{2}+\frac{C_n-\lambda_n}{2\eta}\right)^2.$$

It follows

$$\left(\nu^N_{\eta,n}-\frac{\left(\varepsilon^N_n+w_n\right)C_n}{2}-\frac{C_n-\lambda_n}{2\eta}\right)^2\geq\frac{C_n}{\eta}+\left(\frac{\left(\varepsilon^N_n+w_n\right)C_n}{2}-\frac{C_n-\lambda_n}{2\eta}\right)^2.$$

The above is equivalent to

$$\left(\nu^N_{\eta,n}-\Phi_n\right)^2\geq\Omega_n$$

for $\Phi_n\geq 0$ and $\Omega_n\geq 0$. The solution of the quadratic inequality is ,

$$\begin{cases}\nu^N_{\eta,n}\geq\Phi_n+\sqrt{\Omega_n}, \text{ if } \nu^N_{\eta,n}>\Phi_n,\\ \nu^N_{\eta,n}\geq\Phi_n-\sqrt{\Omega_n}, \text{ if } \nu^N_{\eta,n}\leq\Phi_n.\end{cases} \tag{38}$$

From (11), we have

$$\nu^N_{\eta,n}<\frac{C_n-\lambda_n}{\eta}. \tag{39}$$

Furthermore, for (38) and (39), we can also obtain

$$\Phi_n-\sqrt{\Omega_n}\leq\frac{C_n-\lambda_n}{\eta}\leq\Phi_n+\sqrt{\Omega_n}, \tag{40}$$

using a similar way as (37).

Based on above, the theorem is proved. □