Digital Twin-Empowered Network Planning for Multi-Tier Computing

Conghao Zhou, Jie Gao, Mushu Li, Xuemin (Sherman) Shen, Weihua Zhuang

Abstract—In this paper, we design a resource management scheme to support stateful applications, which will be prevalent in 6G networks. Different from stateless applications, stateful applications require context data while executing computing tasks from user terminals (UTs). Using a multi-tier computing paradigm with servers deployed at the core network, gateways, and base stations to support stateful applications, we aim to optimize long-term resource reservation by jointly minimizing the usage of computing, storage, and communication resources and the cost from reconfiguring resource reservation. The coupling among different resources and the impact of UT mobility create challenges in resource management. To address the challenges, we develop digital twin (DT) empowered network planning with two elements, i.e., multi-resource reservation and resource reservation reconfiguration. First, DTs are designed for collecting UT status data, based on which UTs are grouped according to their mobility patterns. Second, an algorithm is proposed to customize resource reservation for different groups to satisfy their different resource demands. Last, a Meta-learning-based approach is developed to reconfigure resource reservation for balancing the network resource usage and the reconfiguration cost. Simulation results demonstrate that the proposed DT-empowered network planning outperforms benchmark frameworks by using less resources and incurring lower reconfiguration costs.

Keywords—6G, digital twin, network planning, multi-tier computing, Meta learning.

I. INTRODUCTION

The sixth generation (6G) networks are expected to support a wide range of computing applications[1]. A large portion of these applications are *stateful*, meaning that context data is required to execute computing tasks[2-3]. For example, augmented reality applications require volumetric media objects or holograms, as the context data, to process video segments for user terminals (UTs). The prevalent mobile edge computing (MEC) paradigm provides a solution to supporting computing applications with low offloading delay but has limitations in supporting stateful applications[4]. Specifically, edge servers close to UTs generally have limited storage capacity to store all context data of stateful applications. Moreover, even if the context data could be fully stored at an edge server, limited communication coverage and a relatively small number of UTs served by the edge server would degrade storage resource utilization.

C. H. Zhou, X. M. (Sherman) Shen, and W. H. Zhuang. Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, N2L 3G1, Canada (e-mail: c89zhou@uwaterloo.ca; sshen@uwaterloo.ca; wzhuang@uwaterloo.ca).

J. Gao. School of Information Technology, Carleton University, Ottawa, ON, K1S 5B6, Canada (email: jie.gao6@carleton.ca).

M. S. Li. Department of Electrical, Computer, and Biomedical Engineering, Toronto Metropolitan University, Toronto, ON, M5B 2K3, Canada (e-mail: mushu1.li@ryerson.ca)

To address the above limitations, both the industry and the academia have started looking into the collaboration of servers[5-6]. Extending from MEC, *multi-tier computing* integrates multiple servers deployed at the core network, gateways, access points, and other locations in the network for executing computing tasks from UTs. Servers at different tiers have diverse features in terms of resource capacity and service coverage[7]. Specifically, servers deployed at the core network and gateways have larger service coverage and more abundant resources than servers deployed at access points. Through coordinating servers at different tiers, multi-tier computing can exploit different features of servers to support computing applications, especially the stateful ones, in the era of 6G.

Network planning, as an important part of network management, can facilitate the coordination of servers at different tiers. Network planning consists of resource reservation and resource reservation reconfiguration[1]. Resource reservation refers to proactively reserving network resources for satisfying the upcoming resource demands from UTs. Resource reservation reconfiguration refers to timely updating resource reservation decisions to adapt to time-varying resource demands and dynamic network environments. Network planning for supporting stateful applications faces four challenges. First, the reservation of computing, storage, and communication resources for stateful applications is tightly coupled, yielding existing resource reservation solutions for supporting stateless applications inapplicable. Second, the requests for context data may vary across a network, rendering both computing task assignment and storage resource reservation dependent on specific servers and UT mobility patterns. Third, information regarding individual UT status, e.g., UT mobility, is unavailable at the time of network planning, yet such UT-level information can be useful for accurately calculating the amount of network resources needed for supporting stateful applications[8]. Fourth, reconfiguring computing and storage resource reservation for stateful applications in a dynamic network environment yields additional costs due to computing service interruption, which complicates resource reservation reconfiguration[9]. Addressing the above challenges is important to accurate and adaptive network planning for supporting stateful applications in 6G.

Recently, the digital twin paradigm has started attracting attention as a potential solution to advancing network management for 6G[10]. The concept of digital twins (DTs) originates from product life-cycle management in industry, where a DT is a synchronized virtual replica of a physical object[11-12]. For 6G networks, DTs can be introduced to represent individual UTs. Each DT consists of a *UT data profile* that describes

the corresponding UT, including the UT's mobility, service demands, and quality of service (QoS) satisfaction, and *DT functions* for data acquisition, processing, and analysis[10]. The introduction of DTs brings three benefits to network planning. First, historical data contained in DTs can be used to predict UT status in the upcoming time interval, which can, in turn, facilitate customized resource reservation for highly diversified UTs and enable fine-grained network planning. Second, data indicating the performance of network planning can be collected based on DTs, which can provide a foundation for resource reservation reconfiguration in network planning to adapt to a highly dynamic network environment. Third, DTs should acquire extensive and well-organized data that can be used to explore and exploit hidden network characteristics, thereby facilitating effective data-driven network planning approaches to enhancing network performance. Due to the above benefits, DTs can be exploited and designed to improve the granularity, adaptivity, and intelligence of network planning in 6G.

In this paper, we design a network planning scheme for supporting a stateful application in the scenario of multi-tier computing. Our research objective is to find out the minimum amount of network resources (including computing, storage, and communication) needed for supporting the application and also balance the resource usage and the cost from resource reservation reconfiguration in a dynamic network environment. To achieve this objective, we propose a DT-empowered network planning framework with the following two elements: group-based multi-resource reservation and closed-loop resource reservation reconfiguration. First, we design DTs for individual UTs to characterize their status and group them based on their mobility patterns. We propose an algorithm based on matching theory and particle swarm optimization to address the coupling relation among computing, storage, and communication in resource reservation. The proposed method enables customized resource reservation for satisfying different resource demands of UT groups with different mobility patterns. Second, we develop a Meta-learning-based approach for resource reservation reconfiguration to cope with the dynamic network environment. The main contributions of this paper are the followings:

- We propose a novel network planning framework to facilitate fine-grained resource reservation based on UT data contained in DTs;
- We address a challenging multi-resource reservation problem for supporting stateful applications in multi-tier computing;
- We develop an automated closed-loop approach to reconfigure resource reservation in a dynamic network environment for balancing the network resource usage and the cost from reconfiguring resource reservation.

The remainder of this paper is organized as follows. Section II provides an overview of related studies. Section III describes the considered network scenario, the proposed DT-empowered framework, and the system model. Section IV formulates the network planning problem for multi-tier computing. Sections V and VI introduce the proposed solutions for resource reservation and resource reservation reconfiguration, respectively. Section VII presents the simulation results, followed by the conclusion and future work in Section VIII.

II. Related Work

Network resource management is conducted in both short-term and long-term[1]. Short-term resource allocation in the *operation stage* relies on real-time information on individual UTs, such as UT locations, and targets real-time UT satisfaction. By contrast, long-term resource reservation in the *planning stage* is based on aggregated information of UTs, such as the number of UTs covered by an access point, and focuses on network performance, such as resource utilization over a relatively long time period, ranging from several minutes to hours[13-14].

Most works on real-time resource allocation focus on computing task offloading and service placement, among which many consider one-tier computing such as cloud computing or MEC[15-19]. Based on the real-time computing task arrival of each UT, decentralized and centralized communication and computing resource allocation approaches are proposed to minimize the delay of computing task offloading and executing in cloud computing and MEC, respectively[15-16]. Service placement is studied in MEC based on the real-time UT location and the type of service required by each UT to maximize the number of UTs that can be severed under each edge server's resource capacity[17-18]. Some works focus on resource allocation for multi-tier computing[20-24]. Computing resources on fog nodes and the cloud server are allocated to UTs at different locations to satisfy the delay requirements of their computing tasks[20]. Li *et al.* investigate a service placement approach for cloud and edge computing to satisfy each UT's computing demands[21]. Given that the same type of computing tasks can share computing results, Yu *et al.* study joint computing task offloading and service placement in multi-tier computing to reduce the delay of executing computing tasks[22]. Considering space-ground integrated networks, the authors in[23-24] investigate how to allocate computing and communication resources available at satellites, terrestrial access points, and UTs to minimize the delay of executing computing tasks from UTs.

There are less studies on long-term resource reservation for computing task offloading. Based on the aggregated computing demands from all access points, a proactive computing resource reservation approach in MEC is designed to minimize the delay of executing computing tasks[25] and maximizing resource utilization in computing task execution[26], respectively. Yin *et al.* study edge server placement to minimize the network resource usage based on statistical computing demands[27]. There are limited works on long-term resource reservation in multi-tier computing[28,8]. Considering edge and cloud computing, Zhou *et al.* propose a computing resource reservation approach to minimize network resource usage while satisfying different delay requirements of two applications[28]. For servers located at different tiers of space-ground integrated networks, joint communication and computing resource reservation is studied to minimize the long-term cost of delay requirement violation and network reconfiguration[8].
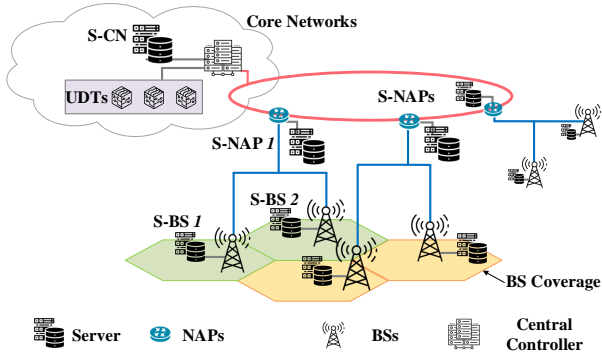
Fig. 1: The considered scenario of multi-tier computing.

While allocating resources to UTs to satisfy their real-time computing demands is important, proactively reserving resources on servers is also essential. In this work, we focus on network planning in the presence of the coupling relation among computing, storage, and communication resources and the impact of UT mobility in long-term resource reservation to support stateful applications. We leverage DTs to improve the granularity and effectiveness of network planning as compared to conventional resource reservation.

III. NETWORK SCENARIO AND SYSTEM MODEL

In this section, we first introduce the considered network scenario of multi-tier computing. Then, we propose a DT-empowered network planning framework to support stateful applications and present the corresponding system model.

A. Network Scenario

The considered scenario is shown in Fig. 1. Computing servers are deployed at different locations, including: (i) base stations (BSs); (ii) network aggregation points (NAPs), such as gateways; and (iii) the core network (CN)[4]. Each server can build one dedicated virtual machine (VM) to execute computing tasks from UTs for the stateful application. The service coverage area of a server at a BS (S-BS) is the BS's communication coverage. The service coverage area of a server at a NAP (S-NAP) is the union of the communication coverage of all the BSs connected to it. For example, S-NAP 1 connects to S-BS 1 and S-BS 2, and the communication coverage area of S-NAP 1 consists of the two green cells as shown in Fig. 1. The server at the CN (S-CN) can provide computing service to all UTs in the considered network. The service coverage areas of servers at the same tier of the network do not overlap. UTs generate computing tasks and offload their computing tasks to a server when located in its service coverage area. We assume that the computing task generation at all UTs corresponds to an identical statistical process due to the same stateful application. The VM at the server then executes the offloaded computing tasks and sends computing results back to the UTs. Denote the S-CN by e^{cn}, and let $\mathcal{E}^{\mathrm{bs}}$ and $\mathcal{E}^{\mathrm{nap}}$ denote the set of S-BSs and the set of S-NAPs, respectively. Let $\mathcal{E} = \mathcal{E}^{\mathrm{bs}} \cup \mathcal{E}^{\mathrm{nap}} \cup \{e^{\mathrm{cn}}\}$ represent the set of all servers in the network. Denote the set of BSs by $\mathcal{B} = \{1, 2, \cdots, B\}$, and let $\mathcal{E}_b \subset \mathcal{E}$ denote the set of servers that include BS $b \in \mathcal{B}$ in their service coverage areas.

We focus on resource reservation to support stateful applications, which requires context data for executing computing tasks. The application uses a fixed set of context data, and the popularity of different data chunks in this set is different. Moreover, the popularity of each data chunk can vary at different BSs across the network, i.e., the popularity of context data chunks is location-dependent. Therefore, if a UT moves into the coverage of a different BS, its requests for context data may also change, creating a dependence of its resource demand on its mobility. A UT that connects to more BSs within a time interval is considered to have higher mobility. UTs with similar mobility patterns can be grouped together in resource reservation due to their similar need for context data.

The time duration of interest is partitioned into K time intervals of length τ (e.g., 5 to 10 minutes per time interval). Denote the set of time intervals by $\mathcal{K} = \{1, 2, \cdots, K\}$. A central controller in the CN maintains the DTs for UTs and makes resource reservation decisions proactively for the stateful application at the beginning of each time interval. We define vector $\boldsymbol{\phi}_{d,k}$ to represent the mobility pattern of UT d in time interval $k \in \mathcal{K}$, where the bth element of $\boldsymbol{\phi}_{d,k}$ represents the time duration in which UT d is in the coverage of BS b during the time interval. Table I lists the important symbols for easy reference.

B. DT-empowered Network Planning

In this subsection, we present the idea of DT-empowered network planning with our specific design of DTs, which is a succession and development of the framework in[10]. A DT is created for an individual UT, called UDT, which consists of a UT data profile and UDT functions. As shown in Fig. 2, UDTs are located at the CN, and UT data profiles are maintained and updated by the central controller via UDT functions. Specifically, each UT data profile is a well-organized set of UT data. In this work, the data attributes of each UDT consist of the mobility of the corresponding UT, including the UT's location and velocity, as well as the information of each computing task from the UT, including context data requirement, input and output data size, computing workloads, and resource demands. There are three UDT functions used to manage and analyze UT data profiles for network planning, as follows:

- **Data collection and storage function** – UT data required for network planning is collected from individual UTs via BSs or offered by service providers. Data regarding UT mobility, e.g., UT locations, can be uploaded by individual UTs periodically, as specified in 5G. Data regarding services, e.g., computing workloads and required context data, can be obtained from service providers or computing servers[4]. The collected UT data is stored in the corresponding UDTs;
- **Data processing function** – At the beginning of time interval k, the mobility patterns of each UT in the past T time intervals, i.e., $\boldsymbol{\phi}_{d,k'}, \forall k' \in [k-T, k-1]$, are

TABLE I: List of Symbols

Symbols	Definition
a_k	The indicator on whether the resource reservation in time interval k should be reconfigured
$c_{e,k}^n$	The amount of computing resource reserved at server e for group n in time interval k
$f_{b,e,k}^n$	The load of computing tasks from group n covered by BS b and assigned to server e in time interval k
$f_{b,e,k}^{n,i}$	The load of computing tasks requiring chunk i from group n covered by BS b and assigned to server e in time interval k
$g_{e,k}$	The amount of storage resource reserved at server e in time interval k
$\mathcal{I}_{e,k}$	The set of chunks stored at server e in time interval k
L	The data volume of each chunk of context data
L^{re}	The data volume of remotely accessing context data for each computing task
$m_{e,k}^n$	The load of computing tasks from group n assigned to server e in time interval k
o_k^{v}	The cost from reconfiguring resource reservation in time interval k
$p_{b,k}^i$	The request ratio of chunk i in the coverage of BS b in time interval k
$v_{b,e,k}^{\mathrm{up}}, v_{b,e,k}^{\mathrm{down}}$	The communication resource usage of uploading and downloading, respectively, between server e and BS b in time interval k
$v_{n,e,k}^{\mathrm{re}}$	The communication resource usage of remote access for executing computing tasks at server e in time interval k
$w^{\mathrm{c}}, w^{\mathrm{s}}, w^{\mathrm{o}}$	The weights of communication, storage, and communication resource usage, respectively
$x_{b,k}^n, \tilde{x}_{b,k}^n$	The actual and predicted load of computing tasks, respectively, from group n in the coverage of BS b in time interval k
$\tilde{\boldsymbol{x}}_k^n$	The spatial task distribution of group n in time interval k
α, β, γ	The input data size, computing workload, size of computing result of each computing task, respectively
Δ_k	The network resource usage during time interval k
$\epsilon_{e,k}^{\mathrm{c}}, \epsilon_{e,k}^{\mathrm{s}}$	The computing and storage resource usage of server e, respectively, during time interval k
$\eta_{b,e}$	The communication resource usage of transmitting a bit data between BS b and server e
$\xi_e^{e'}$	The communication resource usage of accessing context data remotely from server e' to server e

Fig. 2: The designed UDTs used for network planning.

obtained based on the UT data in the corresponding UDT. Then, the historical mobility patterns are used to predict the mobility pattern of each UT, i.e., $\phi_{d,k}$, in the subsequent time interval. Other data prediction, e.g., spatial task distributions and requested context data, based on historical data in UDTs is also conducted via this function;

- **UT grouping function** – UTs are grouped based on their predicted mobility patterns, i.e., $\{\phi_{d,k}, \forall d\}$, at the
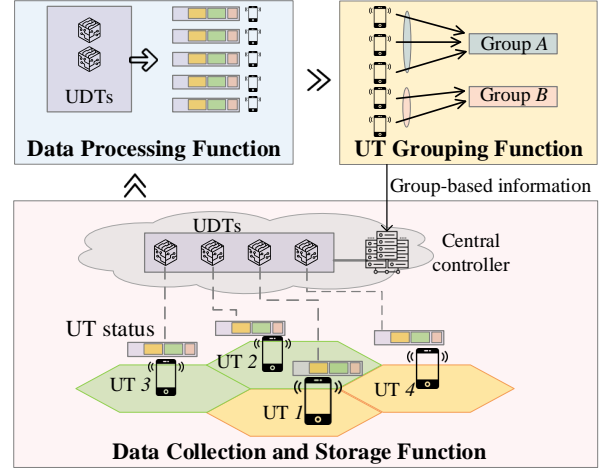
beginning of each time interval. UTs from the same group have similar network resource demands for executing computing tasks due to their similar mobility patterns, allowing for an accurate approximation of resource demands for UTs from each group. Based on UT grouping, network resources can be reserved accurately to achieve fine-grained network planning.[1]

The UDT functions are used to manage and analyze UT data profiles to empower network planning. However, the UDT functions do not make network planning decisions but only provide information for network planning.

Based on UDTs, we propose a novel network planning framework as shown in Fig. 3, which includes two core elements: group-based resource reservation and closed-loop resource reservation reconfiguration.

- **Group-based resource reservation:** Based on UT grouping, we reserve storage and computing resources accurately to satisfy the resource demands for UEs in each group in the upcoming time interval. Denote the set of groups by $\mathcal{N} = \{1, 2, \cdots, N\}$. Let $x_{b,k}^n$ denote the number of computing tasks generated by the UTs who are associated with group n and are in the coverage of BS b during time interval k. We refer to matrix $\boldsymbol{x}_k^n = [x_{b,k}^n]_{\forall b \in \mathcal{B}}$ as *spatial task distribution* of group n in time interval k. Since it is impossible to know the actual value of $\boldsymbol{x}_k^n$ at the beginning of time interval k, $\boldsymbol{x}_k^n$ is predicted based on historical data contained in UDTs. We use superscript "$\sim$", e.g., $\tilde{x}$, to represent the predicted values of x. Given spatial task distributions, we propose an algorithm to address the storage and computing resource reservation problem with highly coupled variables. The detail of group-based resource reservation is discussed in Section V.

---

[1]Note that the mechanism of UT grouping, as well as the data attributes used for grouping, can be customized based on the data contained in UDTs in different network scenarios. The number of groups depends on the trade-off between granularity and complexity.
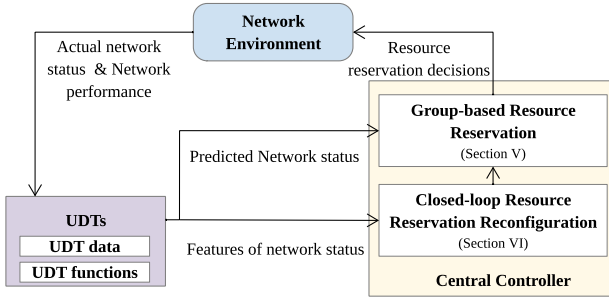
(a) Workflow.
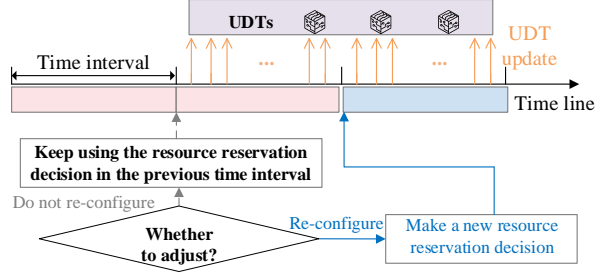
(b) Time line.

Fig. 3: An illustration of DT-empowered network planning.

- **Closed-loop resource reservation reconfiguration:** Since spatial task distributions may change across time intervals, reconfiguring the reserved resources on servers to adapt to dynamic computing demands is necessary but yields additional cost from reconfiguring resource reservation. We design a closed-loop approach to reconfigure resource reservation reconfiguration for balancing the network resource usage and the cost from reconfiguring resource reservation, and the time line is shown in Fig. 3(b). Specifically, at the beginning of each time interval, the central controller identifies whether to reconfigure resource reservation with a proposed algorithm. If the resource reservation needs to be reconfigured for the subsequent time interval, the controller will make a new resource reservation decision; Otherwise, the controller will keep using the resource reservation from the previous time interval. At the end of each time interval, the data regarding network performance is collected based on UDTs. The detail of closed-loop resource reservation reconfiguration is discussed in Section VI.

C. Computing Model

The overall load of computing tasks assigned to each server during a time interval affects network resource reservation on the server. Denote the load of computing tasks generated by UTs from group n in the coverage of BS b and assigned to server e during time interval k by $f_{b,e,k}^n$. The relation between computing task assignment and spatial task distribution is given by

$$\sum_{e \in \mathcal{E}_b} f_{b,e,k}^n = \tilde{x}_{b,k}^n, \ \forall b \in \mathcal{B}, n \in \mathcal{N}, k \in \mathcal{K}. \tag{1}$$

Let $m_{e,k}^n$ denote the overall load of computing tasks generated by UTs from group n and assigned to server e during time interval k,

$$m_{e,k}^n = \sum_{b \in \mathcal{B}_e} f_{b,e,k}^n, \ \forall e \in \mathcal{E}, n \in \mathcal{N}, k \in \mathcal{K}, \tag{2}$$

where $\mathcal{B}_e$ denotes the set of BSs within the service coverage of server e. In Eq. (2), the fine granularity of computing task assignment, reflected through $f_{b,e,k}^n$, helps determine the computing load at each server accurately.

Since we consider one specific stateful application, the computing tasks for this application are assumed to have (approximately) the same input data size (in bits), computing workload (in CPU cycles per bit), and result data size (in bits).[2] Let α, β, and γ denote the average input data size, the average computing workload, and the average result data size, respectively. Each server can reserve a certain amount of computing resource (in CPU cycles per second) for VMs to execute computing tasks. Let $c_{e,k}^n$ denote the amount of computing resource of server e reserved for group n during time interval k, and define $\boldsymbol{c}_k = [c_{e,k}^n]_{\forall e \in \mathcal{E}, n \in \mathcal{N}}$ as the computing resource reservation decision in time interval k. For time interval k, the computing resource reservation should satisfy the following constraint:

$$\sum_{n \in \mathcal{N}} c_{e,k}^n \leq C_e, \ \forall e \in \mathcal{E}, k \in \mathcal{K}, \tag{3}$$

where C_e denotes the maximum computing resource at server e that can be utilized for the stateful application. We assume that the computing resources on the S-CN, i.e., $e = e^{\text{cn}}$, is sufficient for executing all computing tasks. The time that server e takes for executing the computing tasks assigned to it from group n during time interval k should satisfy the following requirement:

$$\frac{m_{e,k}^n \alpha \beta}{c_{e,k}^n} \leq \tau^{\text{p}}, \ \forall e \in \mathcal{E}, n \in \mathcal{N}, k \in \mathcal{K}, \tag{4}$$

where τ^{p} denotes the maximum tolerable computation time.[3] Due to (4), computing resource reservation and computing task assignment for each server are mutually dependent.

Let $\epsilon_{e,k}^{\text{c}}$ represent the overall computing resource usage for executing all computing tasks assigned to server e in time interval k, which is computing load-dependent. Based

---

[2]The proposed framework can be straightforwardly extended to handle the case of computing tasks with different input data sizes, computing workloads, and result data sizes by leveraging DTs to collect such information for each computing task. The problem-solving approach (including the proposed algorithms) remains applicable in such a case.

[3]We do not consider queuing delay in the planning stage because we do not assume a particular computing task arrival pattern. For modeling the queueing delay, the task arrival pattern must be known *a priori*. However, such a pattern is usually unavailable in practice, while assuming a particular arrival pattern can oversimplify the scenario.

on [29-30], we adopt a linear model of computing resource usage as follows:

$$\epsilon_{e,k}^{\mathrm{c}} = \sum_{n \in \mathcal{N}} \varepsilon_e^{\mathrm{c}} m_{e,k}^n, \ \forall e \in \mathcal{E}, k \in \mathcal{K}, \tag{5}$$

where $\varepsilon_e^{\mathrm{c}}$ is the computing resource usage for executing each computing task at server e.

D. Storage Model & Remote Access

Different from stateless applications, the execution of a computing task for stateful applications requires the corresponding context data. If the context data is not in the storage of the server, the server should download the context data from a remote server, thereby yielding additional communication resource usage. We model the storage and the additional communication resource usage for the stateful application in this subsection.

Denote the amount of the reserved storage resource (in bits) of server e during time interval k and the storage capacity (in bits) of server e by $g_{e,k}$ and G_e, respectively. The value of $g_{e,k}$ should satisfy the following constraint:

$$g_{e,k} \le G_e, \ \forall e \in \mathcal{E} \backslash \{e^{\mathrm{cn}}\}, n \in \mathcal{N}, k \in \mathcal{K}. \tag{6}$$

We assume that the storage resources on the S-CN, i.e., $e = e^{\mathrm{cn}}$, is sufficient for storing all context data. Define $\boldsymbol{g}_k = [g_{e,k}]_{\forall e \in \mathcal{E} \backslash \{e^{\mathrm{cn}}\}}$ as the storage resource reservation decision for all S-NAPs and S-BSs in time interval k. Based on the model of storage resource usage in [31], the resource usage for reserving storage resource at server e in time interval k, denoted by $\epsilon_{e,k}^{\mathrm{s}}$, is given by

$$\epsilon_{e,k}^{\mathrm{s}} = \varepsilon_e^{\mathrm{s}} g_{e,k}, \ \forall e \in \mathcal{E}, k \in \mathcal{K}, \tag{7}$$

where $\varepsilon_e^{\mathrm{s}}$ represents the per bit resource usage for reserving storage resource at server e.

Let $\mathcal{I} = \{1, 2, \cdots, I\}$ represent the set of all chunks of context data in the library for the stateful application, where I denotes the number of chunks in the set $\mathcal{I}$. All chunks of context data for the stateful application have the same data structure and thus identical data size (in bits), denoted by L. Executing each computing task requires one chunk of context data in the set $\mathcal{I}$ [32]. Denote the request ratio of chunk i in the coverage of BS b in time interval k by $p_{b,k}^i$, i.e., the load of computing tasks requesting chunk $i \in \mathcal{I}$ over the load of all computing tasks generated in the coverage of BS b in time interval k. The value of $p_{b,k}^i$ may be different in the coverage of different BSs and may vary across different time intervals. Let $\boldsymbol{p}_k = [p_{b,k}^i]_{\forall b \in \mathcal{B}, i \in \mathcal{I}}$ denote the chunk request ratio profile in time interval k, which can be obtained via prediction based on historical data contained in UDTs [33].

The overall data volume of all chunks may be much larger than the storage capacities of S-BSs and S-NAPs. Each S-BS and S-NAP can only store some chunks of context data for executing computing tasks prior to the beginning of each time interval. The S-CN stores all chunks of context data. Let $\mathcal{I}_{e,k} \subseteq \mathcal{I}$ denote the set of chunks stored at server e in time interval k. Given the amount of reserved storage resource on server e, i.e., $\boldsymbol{g}_k$, the number of chunks in set $\mathcal{I}_{e,k}$ is $|\mathcal{I}_{e,k}| = \lfloor g_e/L \rfloor$ where $\lfloor \cdot \rfloor$ represents the floor function.

Since our focus is storage resource reservation, we follow the hierarchical storage policy in [34-35] to determine the set of stored chunks on each S-BS and S-NAP, i.e., $\mathcal{I}_{e,k}$, based on the *effective request ratio* of chunks. Define the effective request ratio of chunk i for executing the computing tasks assigned to server e in time interval k as $q_{e,k}^i$, i.e., the load of computing tasks requesting chunk $i \in \mathcal{I}$ over the load of all computing tasks generated in the service coverage of server e in time interval k. In multi-tier computing, the stored chunks on S-NAPs and the S-CN may not be requested for computing task executing when the chunks are stored on S-BSs for the sake of reducing communication resource usage. As a result, the effective request ratio of a chunk for an S-NAP depends on not only the chunk request ratio profile, i.e., $\boldsymbol{p}_k$, but also the set of chunks stored on S-BSs and computing task assignment, which is difficult to determine.

For simplicity, we make two following assumptions to estimate the effective request ratio for each S-NAP. First, S-BS e located at BS b stores $|\mathcal{I}_{e,k}|$ chunks with largest $p_{b,k}^i$. Second, given set $\mathcal{I}_{e,k}$ for any S-BS, the computing tasks requesting a chunk stored on the S-BS are assigned to the S-BS as much as possible when not violating the computing resource capacity of the S-BS [35]. Based on the two assumptions, the value of $q_{e,k}^i$ for an S-BS equals to the request ratio of chunk i in the coverage of the corresponding BS in time interval k, i.e., $p_{b,k}^i$; The value of $q_{e,k}^i(\boldsymbol{g}_k, \boldsymbol{p}_k)$ for an S-NAP can be estimated given $\boldsymbol{g}_k$ and $\boldsymbol{p}_k$, which is detailed in Appendix A. Each S-BS and S-NAP sorts the chunks in a non-increasing order based on the effective request ratio, i.e., $q_{e,k}^i(\boldsymbol{g}_k, \boldsymbol{p}_k)$, and stores the most requested $|\mathcal{I}_{e,k}|$ chunks.

When the chunk required for executing a computing task is not found in the storage of an S-BS, the S-BS will first attempt to download the chunk of context data from the S-NAP covering it and resort to the S-CN in the case that the S-NAP does not have the chunk of context data either. When the chunk required for executing a computing task is not found in the storage of an S-NAP, the server will download the context data from the S-CN. For any server, accessing another server remotely and downloading context data from the remote server yields additional communication resource usage, referred to as remote access of context data [36].

Denote the data volume (in bits) of remotely accessing context data for each computing task by L^{re}, the value of which may be different from the value of L due to headers used by transmission protocols. Let $f_{b,e,k}^{n,i}$ denote the number of computing tasks that require chunk i and are generated by group n in the coverage of BS b and assigned to server e during time interval k. The relation between $f_{b,e,k}^n$ defined in Eq. (1) and $f_{b,e,k}^{n,i}$ is $f_{b,e,k}^n = \sum_{\mathcal{I}} f_{b,e,k}^{n,i}$. Define the computing task assignment in time interval k as $\boldsymbol{f}_k = [f_{b,e,k}^{n,i}]_{\forall b \in \mathcal{B}, \forall e \in \mathcal{E}_b, n \in \mathcal{N}, i \in \mathcal{I}_{e,k}}$. For S-BS e, the communication resource usage for downloading context data from S-NAP e' covering S-BS e for executing computing tasks from group n during time interval k is given by:

$$v_{n,e,k}^{\text{re}} = L^{\text{re}} \xi_e^{e'} \sum_{i \in \overline{\mathcal{I}}_{e,k} \cap \mathcal{I}_{e',k}} \sum_{b \in \mathcal{B}_e} f_{b,e,k}^{n,i} +$$
$$L^{\text{re}} \xi_e^{e^{\text{cn}}} \sum_{i \in \overline{\mathcal{I}}_{e,k} \cup \mathcal{I}_{e',k}} \sum_{b \in \mathcal{B}_e} f_{b,e,k}^{n,i}, \ \forall e \in \mathcal{E}^{\text{bs}}, \quad (8)$$

where coefficients $\xi_e^{e'}$ and $\xi_e^{e^{\text{cn}}}$ represent the communication resource usage for downloading per bit context data to S-BS e from S-NAP e' and the S-CN, i.e., e^{cn}, respectively. In Eq. (8), the first term represents the communication resource usage for server e to access the context data remotely from server e' at the NAP, and the second term represents the communication resource usage for server e to access the context data remotely from S-CN e^{cn}. For S-NAP e, the communication resource usage for downloading context data from the S-CN during time interval k is as follows:

$$v_{n,e,k}^{\text{re}} = L^{\text{re}} \xi_e^{e^{\text{cn}}} \sum_{i \in \overline{\mathcal{I}}_{e,k}} \sum_{b \in \mathcal{B}_e} f_{b,e,k}^{n,i}, \ \forall e \in \mathcal{E}^{\text{nap}}, \quad (9)$$

where $\xi_e^{e^{\text{cn}}}$ denotes the communication resource usage for downloading per bit context data to S-NAP e from the S-CN, i.e., e^{cn}.

E. Communication Model

Generally, communication resource usage for uploading the input data and downloading the result of a computing task involves two parts: (i) the resource usage for the wireless communication between the UT and the connected BS; and (ii) the resource usage for the wired communication between the BS and a server. In the considered scenario, each UT is associated with only one BS. Regardless of which computer server deployed in the multi-tier network is selected for executing the computing task, the resource consumption for the wireless communication between the associated BS and the UT for uploading input data and downloading computing results is a constant. As a result, the wireless communication does not affect the solution of the resource reservation problem. Since S-BSs and BSs are co-located, there is no additional communication resource usage if any computing task is processed at an S-BS.

Denote the maximum communication resource usage of server $e \in \mathcal{E}^{\text{nap}} \cup \{e^{\text{cn}}\}$ for uploading input data and downloading result data by V_e^{up} and V_e^{down}, respectively. Let $\eta_{b,e}$ denote the coefficient representing the communication resource usage for uploading and downloading per bit data between BS b and server e. The communication resource usage for uploading input data and downloading computing results between server e and BS b during time interval k are given by:

$$v_{b,e,k}^{\text{up}} = \alpha \eta_{b,e} \sum_{n \in \mathcal{N}} \sum_{i \in \mathcal{I}} f_{b,e,k}^{n,i}, \ \forall e \in \mathcal{E} \backslash \{e^{\text{bs}}\}, b \in \mathcal{B}_e, k \in \mathcal{K},$$
$$(10)$$

and

$$v_{b,e,k}^{\text{down}} = \gamma \eta_{b,e} \sum_{n \in \mathcal{N}} \sum_{i \in \mathcal{I}} f_{b,e,k}^{n,i}, \ \forall e \in \mathcal{E} \backslash \{e^{\text{bs}}\}, b \in \mathcal{B}_e, k \in \mathcal{K},$$
$$(11)$$

respectively.

IV. PROBLEM FORMULATION

We formulate the problem of planning-stage resource reservation for multi-tier computing to support stateful applications in this section. We aim to find out the minimum amount of network resources needed for supporting the application while balancing the resource usage and the cost from reconfiguring resource reservation in the presence of network dynamics.

Let $\boldsymbol{r}_k = [\boldsymbol{c}_k, \boldsymbol{g}_k, \boldsymbol{f}_k]$ denote the resource reservation decision, including computing resource reservation, storage resource reservation, and computing task assignment, in time interval k. Variable $\boldsymbol{r}_k$ is determined at the beginning of time interval k. Given $\boldsymbol{r}_k$, the overall network resource usage in time interval k, denoted by $\Delta_k(\boldsymbol{r}_k)$, can be obtained based on Eqs. (5), (7), (10), and (11), as follows:

$$\Delta_k(\boldsymbol{r}_k) = \sum_{e \in \mathcal{E}} w^{\text{c}} \epsilon_{e,k}^{\text{c}} + w^{\text{s}} \epsilon_{e,k}^{\text{s}} + w^{\text{o}}(v_{e,k}^{\text{re}} + v_{e,k}^{\text{co}}), \quad (12)$$

where $v_{e,k}^{\text{co}} = \sum_{b \in \mathcal{B}} (v_{b,e,k}^{\text{up}} + v_{b,e,k}^{\text{down}})$, and w^{c}, w^{s}, and w^{o} are the weights of the computing, storage, and communication resource usage, respectively. Since the spatial task distribution may vary across time intervals, even if a resource reservation decision can minimize instantaneous resource usage in Eq. (12) in time interval k, the same decision may not minimize the overall network resource usage in the subsequent time intervals. Reconfiguration is required for the resource reservation to adapt to the dynamic spatial task distribution, while reconfiguring resource reservation yields additional cost, e.g., the cost from vertical scaling of VM[9]. Denote by O^{v} the cost from reconfiguring resource reservation.

Let $a_k \in \{0, 1\}$ indicate whether the resource reservation in time interval k should be reconfigured or not, which is determined at the beginning of time interval k. If $a_k = 0$, the controller should make new resource reservation decision for time interval k; Otherwise, the controller should keep using the resource resource from time interval $k - 1$. Define $\tilde{\boldsymbol{x}}_k = [\tilde{x}_{b,k}^n]_{\forall b \in \mathcal{B}, n \in \mathcal{N}}$ as spatial task distributions of groups, referred to as group-based spatial task distribution. Let function $\boldsymbol{r}_k = \psi(\tilde{\boldsymbol{x}}_k, \boldsymbol{p}_k)$, representing that the resource reservation decision is made according to the group-based spatial task distribution and the chunk request profile in time interval k. The value of $\boldsymbol{r}_k$, resource reservation in time interval k, evolves as follows:

$$\boldsymbol{r}_k = (1 - a_k)\psi(\tilde{\boldsymbol{x}}_k, \boldsymbol{p}_k) + a_k \boldsymbol{r}_{k-1}, \ \forall k \in \mathcal{K}. \quad (13)$$

Given the value of a_k, the cost from reconfiguring resource reservation in time interval k, denoted by o_k^{v}, is as follows:

$$o_k^{\text{v}} = (1 - a_k)O^{\text{v}}, \ \forall k \in \mathcal{K}. \quad (14)$$

The problem of minimizing the long-term network resource usage and the cost from reconfiguring resource reservation

over K time intervals, is formulated as follows:

$$\text{P0:} \min_{\boldsymbol{a},\boldsymbol{R}} \sum_{k \in \mathcal{K}} \frac{\Delta_k(\boldsymbol{r}_k) + \lambda o_k^{\text{v}}}{\sum_{b \in \mathcal{B}} \sum_{n \in \mathcal{N}} \tilde{x}_{b,k}^n} \tag{15a}$$

$$\text{s.t. } (3), (4), (6) \tag{15b}$$

$$\sum_{b \in \mathcal{B}_e} v_{b,e,k}^{\text{up}} \leq V_e^{\text{up}}, \ \forall e \in \mathcal{E} \setminus \mathcal{E}^{\text{bs}}, k \in \mathcal{K}, \tag{15c}$$

$$\sum_{b \in \mathcal{B}_e} v_{b,e,k}^{\text{down}} \leq V_e^{\text{down}}, \ \forall e \in \mathcal{E} \setminus \mathcal{E}^{\text{bs}}, k \in \mathcal{K}, \tag{15d}$$

$$\sum_{e \in \mathcal{E}} v_{n,e,k}^{\text{re}} \leq V_n^{\text{re}}, n \in \mathcal{N}, k \in \mathcal{K}, \tag{15e}$$

$$c_{e,k}^n \in \mathbb{R}^+, \ \forall e \in \mathcal{E}, n \in \mathcal{N}, k \in \mathcal{K}, \tag{15f}$$

$$g_{e,k} \in \mathbb{R}^+, \ \forall e \in \mathcal{E} \setminus \{e^{\text{cn}}\}, n \in \mathcal{N}, k \in \mathcal{K}, \tag{15g}$$

$$f_{b,e,k}^{n,i} \in \mathbb{N}, \ \forall b \in \mathcal{B}, e \in \mathcal{E}, n \in \mathcal{N}, k \in \mathcal{K}, \tag{15h}$$

$$a_k \in \{0,1\}, \ \forall k \in \mathcal{K}, \tag{15i}$$

where V^{re} denotes the maximum communication resource usage of remote access for one computing task, $\boldsymbol{a} = [a_k]_{\forall k \in \mathcal{K}}$, $\boldsymbol{R} = [\boldsymbol{r}_k]_{\forall k \in \mathcal{K}}$, and λ is the weight balancing the network resource usage and the cost from reconfiguring resource reservation. $\mathbb{R}^+$ represents the set of positive real numbers, and $\mathbb{N}$ represents the set of natural numbers. Constraint (15e) limits the communication resource usage of remote access averaged over all computing tasks from each group to be less than V_n^{re}. The solution of (15) provide a lower bound on the resources needed to support the stateful application, taking into account the resource reservation reconfiguration cost in network planning. If the network resources are sufficient, more resources can be reserved for the application for better service quality. In addition, in a practical network supporting multiple applications, statistical multiplexing among resources reserved for different applications can be implemented for high resource utilization.

Solving Problem P0 is challenging due to the following two reasons: (i) For each time interval, determining the value of $\boldsymbol{r}_k$ is a mix-integer optimization problem, and the variables of $\boldsymbol{f}_k$ and $\boldsymbol{g}_k$ are mutually dependent; (ii) determining the value of a_k is a sequential decision making problem, and the decision at any time interval affects the subsequent decisions. To solve Problem P0, we decouple it into two problems. We propose algorithms for resource reservation in each time interval, which are presented in Section V, and a learning-based approach for reconfiguring resource reservation over multiple time intervals, which is presented in Section VI.

V. GROUP-BASED RESOURCE RESERVATION

In this section, we design an algorithm to enable group-based resource reservation in each time interval.

When $a_k = 1$, the controller keeps using the resource reservation decision from time interval $k - 1$. For when $a_k = 0$, we formulate the group-based multi-resource reservation problem in time interval k, given the predicted spatial task distributions $\tilde{x}_k$, as follows:

$$\text{P1:} \min_{\boldsymbol{r}_k} \frac{\Delta_k(\boldsymbol{r}_k)}{\sum_{b \in \mathcal{B}} \sum_{n \in \mathcal{N}} \tilde{x}_{b,k}^n} \tag{16}$$

$$\text{s.t. } (3), (4), (6), (15\text{c-h}).$$

Problem P1 is a combinatorial optimization problem, and variables $\boldsymbol{f}_k$ and $\boldsymbol{g}_k$ are still coupled. We first address computing task assignment, i.e., $\boldsymbol{f}_k$, and computing resource reservation, i.e., $\boldsymbol{c}_k$, given a storage resource reservation decision, i.e., $\boldsymbol{g}_k$. Then, we leverage particle swarm optimization to find the solution of storage resource reservation. The solution of Problem P1 corresponds to group-specific resource reservation, and the total amount resources to be reserved for all groups can be calculated accordingly. The reserved resources can be multiplexed among different groups.

A. Computing Task Assignment and Computing Resource Reservation

Since multiple computing tasks can be assigned to the same server, assigning computing tasks to servers is many-to-one matching. We first transform the many-to-one matching into a one-to-one matching. Specifically, several *virtual servers* are created to represent each physical server. Denote the number of virtual servers for S-NAP or S-BS e by N_e. The value of N_e is the maximum number of computing tasks that can be assigned to server e while not violating the constraints of resource usage, i.e., constraints (3), (15c), and (15d). The number of virtual servers for physical server e is given by:

$$N_e = \\ \begin{cases} \min\{\lfloor \frac{\tau^{\text{p}} C_e}{\alpha\beta} \rfloor, \lfloor \frac{V_e^{\text{up}}}{\sum_{\mathcal{B}_e} v_{b,e,k}^{\text{up}}} \rfloor, \lfloor \frac{V_e^{\text{down}}}{\sum_{\mathcal{B}_e} v_{b,e,k}^{\text{down}}} \rfloor\}, \text{ if } e \in \mathcal{E}^{\text{nap}}; \\ \lfloor \frac{\tau^{\text{p}} C_e}{\alpha\beta} \rfloor, \text{ if } e \in \mathcal{E}^{\text{bs}}, \end{cases} \tag{17}$$

where $\lfloor \cdot \rfloor$ represents the floor function. For an S-NAP, i.e., $e \in \mathcal{E}^{\text{nap}}$, the value of N_e in Eq. (17) is the minimum value among the maximum number of computing tasks that can be executed, i.e., $\lfloor \frac{\tau^{\text{p}} C_e}{\alpha\beta} \rfloor$, the maximum number of computing tasks that can be uploaded, i.e., $\lfloor \frac{V_e^{\text{up}}}{\sum_{\mathcal{B}_e} v_{b,e,k}^{\text{up}}} \rfloor$, and the maximum number of computing tasks that can be downloaded, i.e., $\lfloor \frac{V_e^{\text{down}}}{\sum_{\mathcal{B}_e} v_{b,e,k}^{\text{down}}} \rfloor$; and for an S-BS, i.e., $e \in \mathcal{E}^{\text{bs}}$, the value of N_e in Eq. (17) is the maximum number of computing tasks that can be executed, i.e., $\lfloor \frac{\tau^{\text{p}} C_e}{\alpha\beta} \rfloor$. We let the number of the corresponding virtual servers for the S-CN in time interval k be $\sum_{b \in \mathcal{B}} \sum_{n \in \mathcal{N}} \tilde{x}_{b,k}^n$, i.e., the load of all computing tasks in time interval k, to guarantee that all computing tasks can be processed. Each virtual server is assigned at most one computing task, and assigning computing tasks to virtual servers becomes one-to-one matching.

Given the amount of storage resource reserved on S-BSs and S-NAPs, the sets of stored context data chunks, i.e., $\mathcal{I}_{e,k}$, on all S-BSs and S-NAPs are determined based on the hierarchal storage policy described in Section III.D. Denote by $D_{b,e,k}^{n,i}$ the network resource usage, including resource usage from computing, uplink communication, downlink communication, and remote access, for executing a computing task that requests chunk i and is generated by a UT from group n in the coverage of BS b during time interval k at server e. The calculation of $D_{b,e,k}^{n,i}$ consists of two parts. The first part is the resource usage from computing, uplink communication, and downlink communication, which is not related to the sets of stored chunks, while the second part is the communication resource

---

Algorithm 1: MCLA Algorithm

---

1 Input: $\mathcal{L}_{u,k}^{\text{server}}, \forall u \in \mathcal{U}$

2 Initialization: $\mathcal{U}, \mathcal{U}^{\text{not}} = \mathcal{U}, \mathcal{T}, j = 0$;

3 while $|\mathcal{T}| = 0$ **do**

4 $j = j + 1$;

5   **for** $u \in \mathcal{U}^{not}$ **do**

6   Select the first computing task in preference list $\mathcal{L}_{u,k}^{\text{server}}$ as the proposal from virtual server u;

7   Remove the selected computing task from preference list $\mathcal{L}_{u,k}^{\text{server}}$;

8   **end**

9   Adopt the dynamic programming in [37] to select proposals from the new proposals in iteration j and adjust the matched proposals in iteration $j - 1$ for minimizing the objective function in (19) while satisfying constraint (15e).

10   Remove the matched virtual servers from $\mathcal{U}^{\text{not}}$;

11   Add the unmatched virtual servers to $\mathcal{U}^{\text{not}}$;

12   Remove the matched computing tasks from $\mathcal{T}$ based on the matching result in iteration j;

13 end

14 $\boldsymbol{f}_k \leftarrow \boldsymbol{z}_k$;

15 Output: $\boldsymbol{f}_k$

---

usage from remote access, which depends on the sets of stored chunks. Denote by $W_{b,e} = w^{\text{c}}\varepsilon_e^{\text{c}} + w^{\text{o}}(\alpha + \gamma)\eta_{b,e}$ the sum of resource usage from computing, uplink communication, and downlink communication used to execute a computing task generated in the coverage of BS b at server e. According to the communication resource usage for remote access, the calculation of $D_{b,e,k}^{n,i}$ is categorized into the following four cases:

$$D_{b,e,k}^{n,i} =$$
$$\begin{cases} W_{b,e} + w^{\text{o}}L^{\text{re}}\xi_e^{e'}, & \text{if } e \in \mathcal{E}^{\text{bs}}, e' \in \mathcal{E}^{\text{nap}}, i \in \overline{\mathcal{I}_{e,k}} \cap \mathcal{I}_{e',k}; \\ W_{b,e} + w^{\text{o}}L^{\text{re}}\xi_e^{e^{\text{cn}}}, & \text{if } e \in \mathcal{E}^{\text{bs}}, e' \in \mathcal{E}^{\text{nap}}, i \in \overline{\mathcal{I}_{e,k}} \cup \overline{\mathcal{I}_{e',k}}; \\ W_{b,e} + w^{\text{o}}L^{\text{re}}\xi_e^{e^{\text{cn}}}, & \text{if } e \in \mathcal{E}^{\text{nap}}, i \notin \mathcal{I}_{e,k}; \\ W_{b,e}, & \text{otherwise.} \end{cases}$$
$$(18)$$

In Eq. (18), if chunk i is not stored on S-BS e but stored on S-NAP e', i.e., $i \in \overline{\mathcal{I}_{e,k}} \cap \mathcal{I}_{e',k}$, the communication resource usage for S-BS e to remotely access S-NAP e' for one computing task is $w^{\text{o}}L^{\text{re}}\xi_e^{e'}$; If chunk i is not stored on S-BS e or S-NAP e', i.e., $i \in \overline{\mathcal{I}_{e,k}} \cup \overline{\mathcal{I}_{e',k}}$, the communication resource usage for S-BS e to remotely access the S-CN for one computing task is $w^{\text{o}}L^{\text{re}}\xi_e^{e^{\text{cn}}}$; If chunk i is not stored on S-NAP e, i.e., $i \notin \mathcal{I}_{e,k}$, the communication resource usage for S-NAP e to remotely access the S-CN for one computing task is $w^{\text{o}}L^{\text{re}}\xi_e^{e^{\text{cn}}}$; Otherwise, chunk i is stored on server e, and no communication resource is used for remote access.

Denote the set of virtual servers and the set of computing tasks by $\mathcal{U}$ and $\mathcal{T}$, respectively. Given computing task $t \in \mathcal{T}$ and virtual server $u \in \mathcal{U}$, we can determine the corresponding values of (n, i, b) of the computing task and physical server e. Let $D_k^{(u,t)}$ represent the sum of computing and communication

resource usage for executing computing task t at virtual server u in time interval k, which can be calculated via Eq. (18) based on the corresponding values of (n, i, b) and physical server e. We introduce variable $z_k^{(u,t)} \in \{0, 1\}$ to indicate whether to assign computing task t to virtual server u in time interval k or not and define $\boldsymbol{z}_k = [z_k^{(u,t)}]_{\forall u \in \mathcal{U}, t \in \mathcal{T}}$. If computing task t is assigned to virtual server u, $z_k^{(u,t)} = 1$; Otherwise, $z_k^{(u,t)} = 0$. Finding the solution of $\boldsymbol{f}_k$ in Problem P1 can be transformed into finding the solution of $\boldsymbol{z}_k$, as follows:

$$\text{P2:} \min_{\boldsymbol{z}_k} \sum_{u \in \mathcal{U}} \sum_{t \in \mathcal{T}} D_k^{(u,t)} z_k^{(u,t)}$$
$$\text{s.t. (15e)},$$
$$z_k^{(u,t)} \in \{0, 1\}, \ \forall k \in \mathcal{K}. \tag{19}$$

We propose a matching-based computing task assignment (MCLA) algorithm to select a virtual server for each computing task, as shown in Algorithm 1.

We construct the preference list $\mathcal{L}_{u,k}^{\text{server}}$ for virtual server u in time interval k, which is a vector containing the indexes of all computing tasks that can be assigned to virtual server u in time interval k, sorted by the value of $D_k^{(u,t)}$ in a non-decreasing order. In each iteration, the controller checks the current preference list for each virtual server and selects the first computing task in the preference list as the proposal from the virtual server. After that, the selected computing task is removed from the virtual server's preference list. The controller selects a proposal for each computing task to minimize the objective function in Problem P2 while satisfying constraint (15e), which is a 0-1 Knapsack problem. Let $\mathcal{U}^{\text{not}} \subset \mathcal{U}$ denote the set of virtual servers that are not yet matched. A dynamic programming approach in [37] is adopted to adjust the matching result, i.e., selecting proposals for virtual servers in the set $\mathcal{U}^{\text{not}}$ and adjusting the matched proposals for virtual servers in the set $\mathcal{U} \backslash \mathcal{U}^{\text{not}}$ for minimizing the objective function in Problem P2 while satisfying constraint (15e). Then, set $\mathcal{U}^{\text{not}}$ is updated accordingly after each iteration. A computing task should be re-proposed for each virtual server in the set $\mathcal{U}^{\text{not}}$ in the next iteration. The matching process terminates when all computing tasks in the set $\mathcal{T}$ are successfully matched. Based on the matching result, i.e., $\boldsymbol{z}_k$, we can determine the computing task assignment decision, i.e., $\boldsymbol{f}_k$.

Given $\boldsymbol{f}_k$, the numbers of computing tasks assigned to the servers, i.e., $\boldsymbol{m}_k$, are determined based on Eq. (2). Then, computing resource reservation in time interval k is determined according to Eq. (3), which is:

$$\boldsymbol{c}_k = \frac{\alpha\beta}{\tau^{\text{p}}} \boldsymbol{m}_k, \ \forall k \in \mathcal{K}. \tag{20}$$

The time complexity of Algorithm 1 depends on the number of iterations of the outer loop for matching (Lines 5 - 13) and the time complexity of solving the knapsack problems in each iteration (Line 9). For the outer loop, the time complexity of solving the matching problem is $O(ZX_k)$, where $Z = \sum_{e \in \mathcal{E}} N_e$ and $X_k = \sum_{n \in \mathcal{N}} \sum_{b \in \mathcal{B}} x_{b,k}^n$ denote the number of all virtual servers and the number of all computing tasks in time interval k, respectively [38]. In each iteration, we

---

Algorithm 2: RR Algorithm

---

1 **Input:** ς, ς_1, ς_2, φ_1, φ_2, $l^{\max}$

2 **Initialization:** $l = 0$, $\boldsymbol{s}_y^{(0)} = \boldsymbol{0}$, $\boldsymbol{g}_y^{(0)} \in \mathbb{F}$, $\forall y \in \mathcal{Y}$;

3 $\boldsymbol{f}_y^{(0)}, \forall y \in \mathcal{Y} \leftarrow$ Calculate by Algorithm 1 given $\boldsymbol{g}_y^{(0)}$;

4 $\hat{\Delta}_y, \forall y \in \mathcal{Y} \leftarrow$ Calculate by (12) given $\boldsymbol{c}_y^{(0)}$, $\boldsymbol{g}_y^{(0)}$, and $\boldsymbol{f}_y^{(0)}$;

5 $\hat{\boldsymbol{g}}_y, \forall y \in \mathcal{Y} \leftarrow \boldsymbol{g}_y^{(0)}, \forall y \in \mathcal{Y}$;

6 $\Delta^* \leftarrow \min \left\{ \hat{\Delta}_y, \forall y \in \mathcal{Y} \right\}$;

7 $\boldsymbol{g}^* \leftarrow \hat{\boldsymbol{g}}_{y'}$ where $y' = \arg\min_y \left\{ \hat{\Delta}_y, \forall y \in \mathcal{Y} \right\}$;

8 **while** $l \leq l^{max}$ **do**

9   **for** $y \in \mathcal{Y}$ **do**

10   **if** *constraint* (6) *is not satisfied* **then**

11       $\boldsymbol{g}_y^{(l)} \leftarrow$ Select a position in $\mathbb{F}$ randomly;

12   **end**

13     $\boldsymbol{f}_y^{(l)} \leftarrow$ Calculate by Algorithm 1 given $\boldsymbol{g}_y^{(l)}$;

14     $\Delta_y^{(l)} \leftarrow$ Calculate by Eq. (12) given $\boldsymbol{c}_y^{(l)}$, $\boldsymbol{g}_y^{(l)}$, and $\boldsymbol{f}_y^{(l)}$;

15   **if** $\Delta_y^{(l)} < \hat{\Delta}_y$ **then**

16       $\hat{\boldsymbol{g}}, \hat{\Delta}_y \leftarrow \boldsymbol{g}_y^{(l)}, \Delta_y^{(l)}$;

17   **end**

18   **if** $\Delta_y^{(l)} < \Delta^*$ **then**

19       $\boldsymbol{g}^*, \Delta^* \leftarrow \boldsymbol{g}_y^{(l)}, \Delta_y^{(l)}$;

20   **end**

21     $\boldsymbol{s}_y^{(l+1)}, \boldsymbol{g}_y^{(l+1)} \leftarrow$ Update by Eqs. (21) and (22), respectively;

22   **end**

23 **end**

24 $\boldsymbol{f}^*, \boldsymbol{c}^* \leftarrow$ Calculate by Algorithm 1 and Eq. (20) given $\boldsymbol{g}^*$, respectively;

25 **Output:** $\boldsymbol{g}^*$, $\boldsymbol{f}^*$, $\boldsymbol{c}^*$, Δ^*

---

should solve a knapsack problem to satisfy constraint (15e) for every group. The time complexity of the adopted dynamic programming approach for group n is $O(V_n^{\text{re}} X_k^n)$, where $X_k^n = \sum_{b \in \mathcal{B}} x_{b,k}^n$ denotes the number of all computing tasks from group n in time interval k[37]. Therefore, the time complexity of Algorithm 1 is $O(Z X_k \sum_{n \in \mathcal{N}} V_n^{\text{re}} X_k^n)$.

B. Storage Resource Reservation

Determining the amount of storage resources reserved on S-NAPs and S-BSs is a combinatorial optimization problem. Therefore, finding the globally optimal solution is challenging[39]. Evolutionary heuristics, specifically particle swarm optimization (PSO), is leveraged to achieve the local optima of the problem. Based on PSO, we propose a resource reservation (RR) algorithm to solve Problem P1. In the proposed RR algorithm, we leverage a number of particles, referred to as the particle swarm, where the position of each particle corresponds to the solution of storage resource reservation, i.e., $\boldsymbol{g}_k$. In each iteration, each particle moves within the solution space while adjusting its position and speed dynamically based on[39]. After repeating such an iteration multiple times, the positions of all particles can converge to the same position, which is the

found solution of storage resource reservation[40]. Given the storage resource reservation, the values of $\boldsymbol{f}_k$ and $\boldsymbol{c}_k$ can be determined based on Section IV.A.

The detailed procedures of the RR algorithm are introduced in Algorithm 2. Since the RR algorithm applies to any time interval, we omit subscript "k" in the rest of this subsection. We define the solution space of the storage resource reservation problem as $\mathbb{F}$, where the possible solution of $\boldsymbol{f}_k$, i.e., particles' positions, should satisfy constraint (6). Denote the set of particles and the position of particle y in the l th iteration by $\mathcal{Y}$ and $\boldsymbol{g}_y^{(l)}$, respectively. Let $\hat{\boldsymbol{g}}_y$ and $\boldsymbol{g}^*$ denote the best position of particle y and the best position among all particles' positions, i.e., the particle swarm's best position, up to the l th iteration, respectively. Accordingly, let $\Delta_y^{(l)}$, $\hat{\Delta}_y$, and Δ^* denote the value of Δ in Problem P1 given position $\boldsymbol{g}_y^{(l)}$, $\hat{\boldsymbol{g}}_y$, and $\boldsymbol{g}^*$, respectively. Based on[40], the speed of particle $y \in \mathcal{Y}$ in the l th iteration, denoted by $\boldsymbol{s}_y^{(l)}$, evolves as follows:

$$
\begin{aligned}
\boldsymbol{s}_y^{(l)} = {} & \varsigma \boldsymbol{s}_y^{(l-1)} + \varsigma_1 \varphi_1 \left(\hat{\boldsymbol{g}}_y - \boldsymbol{g}_y^{(l-1)} \right) \\
& + \varsigma_2 \varphi_2 \left(\boldsymbol{g}^* - \boldsymbol{g}_y^{(l-1)} \right),
\end{aligned} \tag{21}
$$

where parameter ς is the weight for each particle to keep its speed from the previous iteration. Parameters ς_1 and ς_2 are cognitive and social coefficients for learning from each particle's own best position and the particle swarm's best position up to the current iteration, respectively, and both are positive random variables for exploring the solution space[40]. The position of particle y, i.e., $\boldsymbol{g}_y^{(l)}$, in the l th iteration is given by:

$$
\boldsymbol{g}_y^{(l)} = \boldsymbol{g}_y^{(l-1)} + \boldsymbol{s}_y^{(l)}. \tag{22}
$$

If a particle moves out of the solution space, the particle is replaced by a new particle with a random position in the solution space $\mathbb{F}$. In this way, the positions of all particles are guaranteed to satisfy constraint (6).

Algorithm 2 shows the detail of the proposed RR algorithm. Denote the maximum number of iteration by $l^{\max}$. Line 2 initializes all particles with the positions in the solution space $\mathbb{F}$. Line 3 to Line 4 obtain the value of Δ in Problem P1, i.e., $\hat{\Delta}_y$, given the position of particle y, i.e., $\boldsymbol{g}_y^{(l)}$. Line 5 to Line 7 find the best solution among all particles based on the value of $\hat{\Delta}_y$. Line 10 to Line 21 update the position of each particle based on Eqs. (21) and (22) in each iteration. The outputs of the RR algorithm include the solution of Problem P1, i.e., $\boldsymbol{g}^*$, $\boldsymbol{f}^*$, and $\boldsymbol{c}^*$, and the value of Δ^*.

VI. META LEARNING BASED RESOURCE RESERVATION RECONFIGURATION

In the preceding section, we solve the network resource reservation problem for one time interval. In this section, we determine the value of $\boldsymbol{a} = [a_k]_{\forall k \in \mathcal{K}}$ to reconfigure resource reservation decisions among K time intervals.

We formulate the sub-problem of resource reservation reconfiguration based on Problem P0, as follows:

$$
\text{P3:} \min_{\boldsymbol{a}} \quad \sum_{k \in \mathcal{K}} \frac{\Delta_k(\boldsymbol{r}_k) + \lambda o_k^{\text{v}}}{\sum_{b \in \mathcal{B}} \sum_{n \in \mathcal{N}} \tilde{x}_{b,k}^n} \tag{23a}
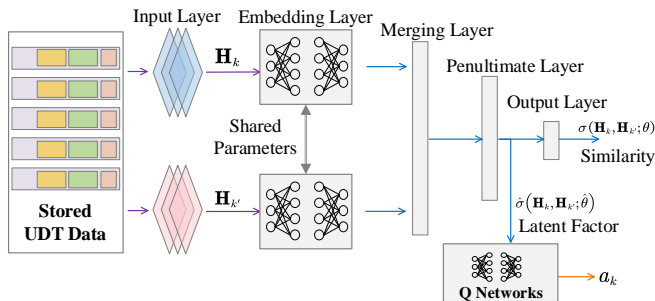$$

$$
\text{s.t. (15i).} \tag{23b}
$$

Fig. 4: The proposed MetaR^3 approach.

We define $k^\star$ as the time interval when the latest resource reservation was reconfigured up to time interval k. In time interval $k^\star$, $a_{k^\star} = 0$, $a_j = 1$ for $j \in [k^\star + 1, \cdots, k - 1]$ and $k^\star < k$. The relation between k and $k^\star$ is given by:

$$k^\star = \begin{cases} k, & \text{if } a_k = 0; \\ k^\star, & \text{otherwise.} \end{cases} \tag{24}$$

Problem P3 is a sequential decision making problem, which can be solved by reinforcement learning (RL) based methods[26,23]. However, RL-based methods cannot be applied directly to solve Problem P3. Resource reservation reconfiguration is owing to the difference of *network status*, i.e., spatial task distributions and chunk request ratio profiles in this work, in different time intervals. Identifying the difference between network status can potentially improve the learning efficiency of RL-based methods.

We propose a <u>Meta</u>-learning-based <u>r</u>esource <u>r</u>eservation <u>r</u>econfiguration (MetaR^3) approach to solve Problem P3, as given in Algorithm 3. At the end of each time interval, data regarding spatial task distributions, chunk request ratio profiles, indicator a_k and the network performance is collected and stored based on UDTs. Meta learning is adopted to capture the *similarity* between network status in two time intervals k and $k^\star$, and the policy of resource reservation decision reconfiguration is learned based on the captured similarity by using RL. Based on the learned policy of resource reservation decision reconfiguration, the value of a_k can be determined. If $a_k = 0$, resource reservation decision is reconfigured at the beginning of time interval k.

Two components are underlying the proposed MetaR^3 approach: (i) capturing the similarity between network status during two different time intervals, and (ii) reconfiguring resource reservation in a closed-loop manner, which are presented in the following two subsections, respectively.

A. Similarity Capture

Denote the network status in time interval k by $\boldsymbol{h}_k = [\boldsymbol{x}_k, \boldsymbol{p}_k]$. The value of $\boldsymbol{h}_k$ is unavailable at the beginning of time interval k. Therefore, we use the data regarding spatial task distributions and chunk request ratio profiles in past T' time intervals contained in UDTs as the features of the network status in time interval k, denoted by $\boldsymbol{H}_k = [\boldsymbol{h}_{k-T'}, \cdots, \boldsymbol{h}_{k-1}]$. Based on $\boldsymbol{H}_k$, the value of a_k is

determined. If $a_k = 0$, $\boldsymbol{H}_k$ can be used to predict the network status, i.e., $\boldsymbol{h}_k$, for making resource reservation decision. Define the similarity between the network status in different time intervals k and k' as $\sigma(\boldsymbol{H}_k, \boldsymbol{H}_{k'})$.

We leverage Meta learning with siamese neural networks to approximate the value of $\sigma(\boldsymbol{H}_k, \boldsymbol{H}_{k'})$[41], as illustrated in Fig. 4. Let $\boldsymbol{\theta}$ and $\hat{\boldsymbol{\theta}}$ denote the parameters of the whole siamese neural networks and the parameters of the siamese neural networks without the output layer, respectively. The inputs of the siamese neural networks are the features of network status, i.e., $\boldsymbol{H}_k$ and $\boldsymbol{H}_{k'}$. The siamese neural networks have two outputs, i.e., the output of the whole siamese neural network and the output of the penultimate layer. The output of the whole siamese neural network represents the value of similarity, denoted by $\sigma(\boldsymbol{H}_k, \boldsymbol{H}_{k'}; \boldsymbol{\theta})$, and the output of the penultimate layer represents the *latent factors* of similarity, denoted by $\hat{\sigma}(\boldsymbol{H}_k, \boldsymbol{H}_{k'}; \hat{\boldsymbol{\theta}})$.

Training siamese neural networks should be based on labeled data. Define $\varrho(\boldsymbol{H}_k, \boldsymbol{H}_{k'}) \in \{0, 1\}$ as a *label* of the features of network status in two time intervals k and k', given by:

$$\varrho(\boldsymbol{H}_k, \boldsymbol{H}_{k'}) = \begin{cases} 1, & \text{if } \Delta_k(\boldsymbol{r}_k) - \Delta_k(\boldsymbol{r}_{k'}) > \lambda O^\text{v}; \\ 0, & \text{otherwise.} \end{cases} \tag{25}$$

In Eq. (25), $\Delta_k(\boldsymbol{r}_k)$ denotes the network resource usage in time interval k if the resource reservation is reconfigured, i.e., $a_k = 0$. $\Delta_k(\boldsymbol{r}_{k'})$ denotes the network resource usage in time interval k if the resource reservation is not reconfigured, i.e., $a_k = 1$, and the resource reservation decision from time interval k' is used, i.e., $\boldsymbol{r}_{k'}$. If $\Delta_k(\boldsymbol{r}_k) - \Delta_k(\boldsymbol{r}_{k'}) > \lambda O^\text{v}$, the network status in two time interval k and k' are considered to be "similar"; Otherwise, the network status in the two time intervals are considered to be "not similar". The features of the network status in any two time intervals and the corresponding value of $\varrho(\boldsymbol{H}_k, \boldsymbol{H}_{k'})$ are referred to as a labeled data entry. The goal of training the siamese neural networks is to let $\sigma(\boldsymbol{H}_k, \boldsymbol{H}_{k'}; \boldsymbol{\theta})$ approximate label $\varrho(\boldsymbol{H}_k, \boldsymbol{H}_{k'})$ by using extensive labeled data entries. The parameters of the siamese neural networks, i.e., $\boldsymbol{\theta}$, are obtained by minimizing the following loss function via gradient descent[41]:

$$\begin{aligned} \boldsymbol{\theta}^* = \arg\min_{\{\boldsymbol{\theta}\}} & \varrho(\boldsymbol{H}_k, \boldsymbol{H}_{k'}) \log\left(\sigma(\boldsymbol{H}_k, \boldsymbol{H}_{k'}; \boldsymbol{\theta})\right) + \\ & (1 - \varrho(\boldsymbol{H}_k, \boldsymbol{H}_{k'})) \log\left(1 - \sigma(\boldsymbol{H}_k, \boldsymbol{H}_{k'}; \boldsymbol{\theta})\right). \end{aligned} \tag{26}$$

The approximated value of similarity, i.e., $\sigma(\boldsymbol{H}_k, \boldsymbol{H}_{k'}; \boldsymbol{\theta})$, can indicate whether the features of network status in two time intervals are similar or not. However, its information on how much a difference between network status is insufficient for reconfiguring resource reservation.[4] Therefore, we use the latent factors of similarity, i.e., $\hat{\sigma}(\boldsymbol{H}_k, \boldsymbol{H}_{k'}; \hat{\boldsymbol{\theta}})$, to determine the value of a_k.

B. Closed-loop Resource Reservation Reconfiguration

We leverage deep Q learning with deep neural networks, named Q networks, to determine the value of a_k given the la-

---

[4]The output layer in the siamese neural networks is used for training the siamese neural networks.

Algorithm 3: MetaR³ Approach

---

1 Input: ρ

2 Initialization: $\boldsymbol{\theta}$, $\hat{\boldsymbol{\theta}}$, $\boldsymbol{\vartheta}$, $k^{\star}$, $\boldsymbol{H}_1$, $\hat{\sigma}_1$

3 for $k = 1, \cdots, K$ **do**

4 $\quad \sigma_k \leftarrow$ Obtain $\hat{\sigma}(\boldsymbol{H}_k, \boldsymbol{H}_{k^{\star}}; \hat{\boldsymbol{\theta}})$;

5 $\quad a_k \leftarrow$ Determine by Eq. (28);

6 $\quad \boldsymbol{r}_k \leftarrow$ Determine by Eq. (13);

7 $\quad \Delta_k(\boldsymbol{r}_k), \hat{\sigma}_{k+1} \leftarrow$ Implement $\boldsymbol{r}_k$ for time interval k;

8 $\quad \boldsymbol{\theta}, \hat{\boldsymbol{\theta}}, \boldsymbol{\vartheta} \leftarrow$ Train parameters via Eqs. (26) and (29);

9 $\quad k^{\star} \leftarrow$ Update by Eq. (24);

10 end

11 Output: a

---

tent factors of similarity. The state and action in time interval k are $\hat{\sigma}(\boldsymbol{H}_k, \boldsymbol{H}_{k^{\star}}; \hat{\boldsymbol{\theta}})$ and a_k, respectively. For simplicity, let $\hat{\sigma}_k$ denote $\hat{\sigma}(\boldsymbol{H}_k, \boldsymbol{H}_{k^{\star}}; \hat{\boldsymbol{\theta}})$ in the rest of this section. Define a Q-value function to represent the discounted long-term resource usage and cost of making decision a_k in state $\hat{\sigma}_k$, given by:

$$Q(\hat{\sigma}_k, a_k) = \sum_{k=1}^{K} \rho^k \frac{\Delta_k + \lambda o_k^{\mathrm{v}}}{\sum_{b \in \mathcal{B}} \sum_{n \in \mathcal{N}} \tilde{x}_{b,k}^n}. \tag{27}$$

where $\rho \in (0, 1)$ is the discount factor. In state $\hat{\sigma}_k$, a_k can be determined based on the Q-values as follows:

$$a_k = \arg \max_{a \in \{0,1\}} Q(\hat{\sigma}_k, a), \ \forall k \in \mathcal{K}. \tag{28}$$

The Q network with parameter $\boldsymbol{\vartheta}$ is used to approximate the Q-value function for learning the policy of resource reservation reconfiguration. The parameters $\boldsymbol{\vartheta}$ of the Q networks are obtained by minimizing the following loss function via gradient descent[23]:

$$\boldsymbol{\vartheta}^* = \arg \min_{\{\boldsymbol{\vartheta}\}} \Big| \frac{\Delta_k + \lambda o_k^{\mathrm{v}}}{\sum_{b \in \mathcal{B}} \sum_{n \in \mathcal{N}} \tilde{x}_{b,k}^n} + \rho \max_a Q(\hat{\sigma}_{k+1}, a; \boldsymbol{\vartheta})$$
$$- Q(\hat{\sigma}_k, a_k; \boldsymbol{\vartheta}) \Big|^2. \tag{29}$$

We summarize the workflow of the MetaR³ approach in Algorithm 3. Line 4 to Line 5 determine a_k at the beginning of time interval k based on the predicted network status obtained from UDTs. Given a_k, Line 6 to Line 7 determine and implement the resource reservation decision. At the end of the time interval, data regarding network performance, actual spatial task distributions, and chunk request ratio profiles is collected and stored in UDTs. Given the stored historical data, the siamese neural networks and Q networks can be trained to adapt to dynamic spatial task distribution and chunk request ratio profile in a closed-loop manner.

VII. PERFORMANCE EVALUATION

A. Simulation Settings

The simulated multi-tier network consists of 1 S-CN, 2 S-NAPs, and 4 to 10 S-BSs. In the network, there are 600 UTs with different trajectories within the networks. Based on the average time within the coverage of each BS for each UT, these UTs are grouped into 2 to 4 groups. The input data size,

TABLE II: Simulation Parameters

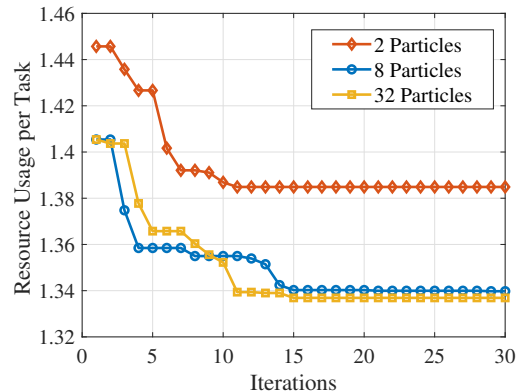| Parameter	Value	Parameter	Value
G^{bs}, G^{nap}	0.9, 2 Gigacycles/s	C^{bs}, C^{nap}	0.75, 1.5 GB
$\xi_{\mathrm{bs}}^{\mathrm{nap}}$, $\xi_{\mathrm{bs}}^{\mathrm{cn}}$	$3.5 * 10^{-9}$, $2.5 * 10^{-9}$	$\xi_{\mathrm{nap}}^{\mathrm{cn}}$	$6 * 10^{-9}$
$\varepsilon_{\mathrm{bs}}^{\mathrm{c}}$, $\varepsilon_{\mathrm{nap}}^{\mathrm{c}}$, $\varepsilon_{\mathrm{cn}}^{\mathrm{c}}$	1, 1, 1	I	20
$\varepsilon_{\mathrm{bs}}^{\mathrm{s}}$, $\varepsilon_{\mathrm{nap}}^{\mathrm{s}}$, $\varepsilon_{\mathrm{cn}}^{\mathrm{s}}$	0.8, 0.5, 1	L	0.15 GB
η^{nap}, η^{cn}	$5 * 10^{-9}$, $9 * 10^{-9}$	τ^{p}	0.5 s
w^{s}, w^{c}, w^{o}	$0.5 * 10^{-7}$, 1, 1	λ	12

Fig. 5: Convergence performance of the proposed RR algorithm.

computing workload, and the size of computing results of each computing task are set to 2 MB, 4 Megacycles/s, and 15 MB, respectively[24]. The network resource usage and the resource capacity of servers at the same tier can be different. The average network resource usage, average resource capacity of servers, and other parameters are listed in Table II.

For the siamese neural networks illustrated in Fig. 4, we use 3 fully connected layers with (64, 64, 32) neurons as a embedding layer. The features of network status $\boldsymbol{H}_k$ and $\boldsymbol{H}_{k'}$ are fed to two embedding layers separately, each with the same structure. The merging layer merges the outputs of the two embedding layers based on Euclidean distance, followed by the penultimate layer with 16 neurons and the output layer with 1 neuron. For the Q networks, we adopt 4 fully connected layers with 128, 512, 128, 32 neurons, respectively. We adopt the RMSprop optimizer and the Adam optimizer for training the siamese neural networks and the Q networks, respectively.

B. Performance of Group-based Resource Reservation

The convergence performance of the proposed RR algorithm for group-based resource reservation is shown in Fig. 5. Given the same spatial task distribution and configuration of all servers, we conduct the simulation with 2, 8, and 32 particles for 30 iterations. The proposed algorithm converges after 10, 14, and 16 iterations, respectively. With more particles, the algorithm achieves better performance at the cost of computation complexity.

Figures 6(a) and 6(b) compare the network resource usage per computing task, versus the load of computing tasks, of the proposed RR algorithm and the benchmark algorithms. We adopt four benchmark algorithms: (1) BS-first, which assigns
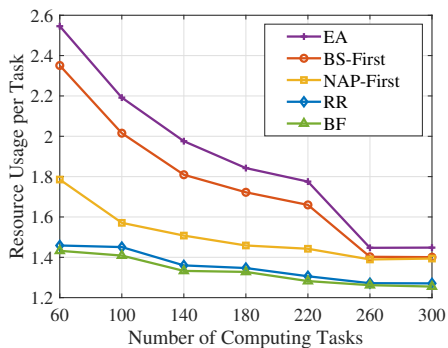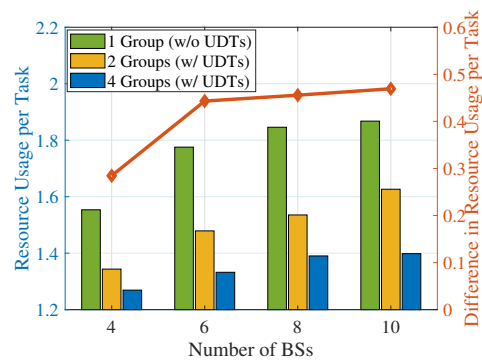
(a) Uneven spatial task distribution.
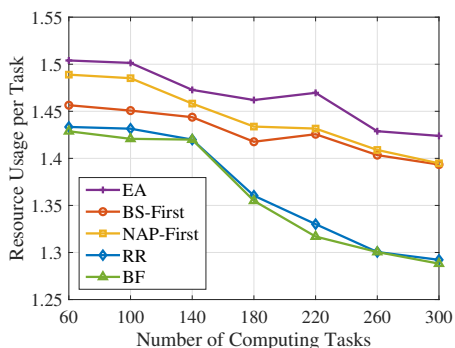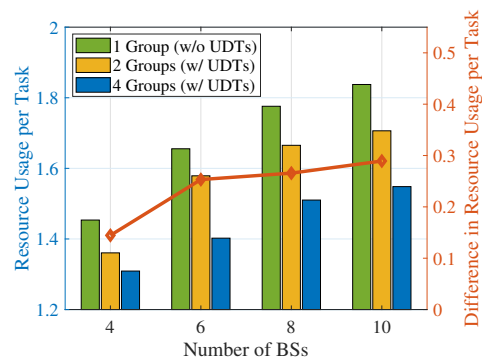

(b) Even spatial task distribution.

Fig. 6: Network resource usage per computing task under even and uneven spatial task distributions.

(a) Heterogeneous settings.

(b) Homogeneous settings.

Fig. 7: Network resource usage per computing task in heterogeneous and homogeneous scenarios.

computing tasks to S-BSs and reserves storage and computing resources on S-BSs first as much as possible, then on S-NAPs, and last on the S-CN. (2) NAP-first, which assigns computing tasks to S-NAPs and reserves storage and computing resources on S-NAPs first as much as possible, then on S-BSs, and last on the S-CN. (3) EA, which assigns computing tasks and reserves storage and computing resources on S-BS, S-NAP, and S-CN with equal priority. (4) BF, which is a brute force algorithm to find the global optimum. We have the three following observations. First, for both even and uneven task distribution, the network resource usage per computing task of the RR algorithm is close to the global optimum and much lower compared to the benchmark algorithms (except the BF algorithm). Second, the network resource usage per computing task decreases as the load of computing tasks generated in the network increases. This is because the storage resource usage per task decreases as the load of computing tasks requesting the same stored chunk increases. Third, the performance gap between the RR algorithm and the benchmark algorithms (except the BF algorithm) under uneven spatial task distribution is larger than under even spatial task distribution. With uneven spatial task distribution, the optimal resource reservation may be different for servers at the same tier. The RR algorithm can differentiate resource reservation decisions for different servers at the same tier using group-based spatial task distribution.

Figures 7(a) and 7(b) show the network resource usage per computing task versus the number of BSs and the number of groups in different scenarios. Specifically, the heterogeneous scenario means that the network resource usage for executing a computing task and resource capacity of servers at the same tier are different. The homogeneous scenario means that the network resource usage for executing a computing task and resource capacity of servers at the same tier are identical. The scheme labeled as "1 Group (w/o UDTs)" represents resource reservation without UDTs, and the schemes labeled as "n Groups (w/ UDTs)" represent the proposed RR algorithm with n groups. We have the following two observations. First, group-based resource reservation with UDTs outperforms resource reservation without UDTs in both homogeneous and heterogeneous scenarios. Without UDTs, group-based spatial task distribution are unknown for resource reservation and computing task assignment. As a result, schemes without UDTs should reserve more resources to satisfy constraint (15e), i.e., each group's communication resource usage for remote access. With more groups, the group-based resource reservation can achieve better performance at the cost of computation complexity and data management for UDTs. Second, the curve with diamond markers represents the performance gap between 4-group based resource reservation with UDTs ("4 Group (w/ UDTs)") and resource reservation without UDTs ("1 Group (w/o UDTs)"). In both homogeneous and heterogeneous scenarios, the effectiveness of group-based resource reservation increases with the number of BSs.
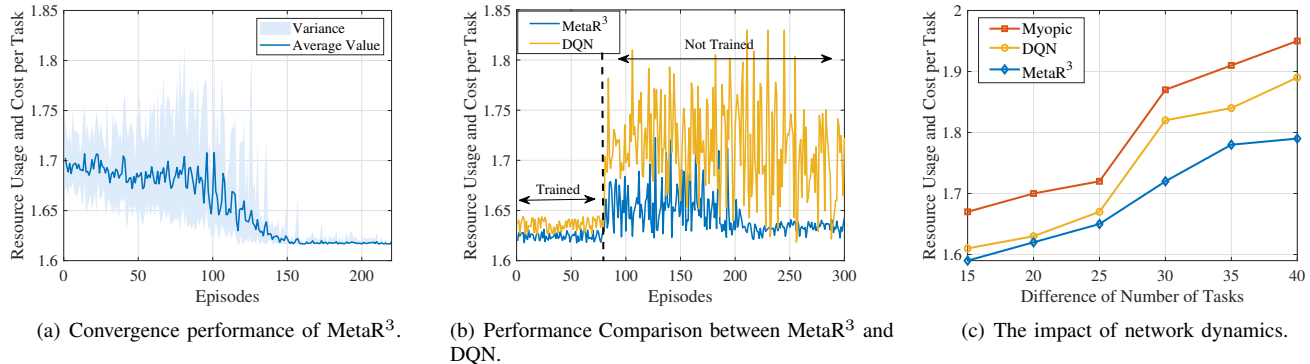
(a) Convergence performance of MetaR^3.

(b) Performance Comparison between MetaR^3 and DQN.

(c) The impact of network dynamics.

Fig. 8: Performance of MetaR^3 in the weighted sum of network resource usage and cost from reconfiguring resource reservation per computing task.

C. Performance of Closed-loop Resource Reservation Reconfiguration

A training dataset that includes spatial task distributions in 80 time intervals is created. Conducting the simulation for all 80 spatial task distributions in the training dataset is referred to as one episode. We conduct 10 simulations on the dataset, and each simulation includes 220 episodes. In Fig. 8(a), the smooth solid line is the average result over 10 simulations, while the spikes in the background represent the corresponding variance. Fig. 8(a) shows that the proposed MetaR^3 algorithm can converge and find a policy of resource reservation reconfiguration given a fixed network environment.

In Fig. 8(b), we compare the convergence performance of the MetaR^3 approach with that of a deep Q Learning (DQN) based algorithm, labeled as "DQN". In DQN, the group-based spatial task distribution is used as the state to determine the value of a_k. We create two datasets with different spatial task distributions. One training dataset is used to train the neural networks in advance, and one evaluation dataset is used to evaluate the convergence performance of MetaR^3. The evaluation dataset reveals network status from unknown network environments. Note that MetaR^3 keeps training the siamese neural networks and the Q networks in unknown network environment due to the closed-loop reconfiguration of resource reservation. We observe that MetaR^3 achieves lower network resource usage and lower cost from reconfiguring resource reservation per computing task, and also converges in fewer episodes in unknown network environment compared with the DQN algorithm. This is because MetaR^3 captures the similarity of network status, instead of learning the variation of network status, to determine a_k even though the current network status is unknown.

Figure 8(c) shows the performance in the weighted sum of the network resource usage and the cost from reconfiguring resource reservation per computing task versus the average difference in the load of computing tasks in adjacent two time intervals. When the average difference in the load of computing tasks in adjacent two time intervals increases, spatial task distribution changes faster. The benchmark algorithm, labeled as "Myopic", determines whether to reconfigure resource
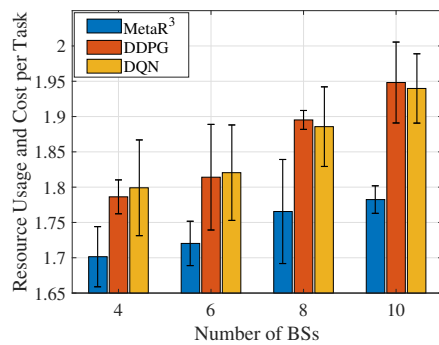

Fig. 9: Resource usage and cost per computing task of the MetaR^3, DDPG, and DQN-based algorithms.

reservation or not in each time interval without considering the long-term impact. We observe that the performance gaps between MetaR^3 and "DQN" and "Myopic" algorithms increase with the average difference in the load of computing tasks in adjacent two time intervals since the similarity capture features of MetaR^3 can reduce the state space for finding a good policy of resource reservation reconfiguration in dynamic network environments, which improves learning efficiency.

In Fig. 9, we compare the performance of the proposed MetaR^3 algorithm with that of two popular RL algorithms, i.e., deep deterministic policy gradient (DDPG)-based (labeled as "DDPG") and DQN-based algorithms in different network environments[26,42]. Specifically, we conduct simulations of the three algorithms with different numbers of BSs and average the resource usage and cost per computing task over three independent simulations. We can observe that the proposed MetaR^3 algorithm outperforms the DDPG- and DQN-based algorithms. This is because both DDPG- and DQN-based algorithms use network status as states. When the network status has a large dimensionality and network environments are highly dynamic, finding the optimal resource reservation reconfiguration policy is challenging for the DDPG and DQN-based algorithms. In contrast, the proposed MetaR^3 algorithm can capture the similarity of network status in consecutive time intervals. Since the similarity is low-dimensional, using

similarities as states has advantage on finding a proper policy of resource reservation reconfiguration, particularly in complicated network environments. Therefore, the proposed MetaR^3 algorithm achieves better network performance than DDPG- and DQN-based algorithms.

VIII. Conclusion and Future Work

In this paper, we have designed DT-empowered network planning for supporting stateful applications in multi-tier computing and proposed two approaches to enable group-based multi-resource reservation and closed-loop resource reservation reconfiguration. Our study focuses on minimizing the long-term network resource usage and the cost from reconfiguring resource reservation. The results have demonstrates that DT-empowered network planning can support UTs with diverse characteristics and adapt to dynamic network environments. In addition, the Meta-learning-based approach can exploit data contained in DTs to facilitate closed-loop network planning. Overall, we have demonstrated the essential role that DTs can play in network planning for 6G. In the future, we will improve the flexibility of DTs by differentiating and optimizing DTs for various applications or UTs.

Appendix

A. Effective Request Ratio for S-NAPs

For S-BS e located at BS b, S-BS e sorts $|\mathcal{I}_{e,k}|$ chunks with largest values of $p_{b,k}^i$ in time interval k. Let $J_{e,k}^i$ denote the order of chunk i among the chunks with largest values of $p_{b,k}^i$ in set $\mathcal{I}_{e,k}$, and denote by $\mathcal{I}_{e,k}^{(J_{e,k}^i)} \subseteq \mathcal{I}_{e,k}$ the set of $J_{e,k}^i$ chunks with largest values of $p_{b,k}^i$. We assume that the computing tasks requesting any chunk in $\mathcal{I}_{e,k}$ are assigned to the S-BS as much as possible while not violating the communication and computing resource capacities. Given different values of $g_{e,k}$ for S-BS e, the load of computing tasks assigned to S-BS e may be different. For S-BS e co-located with BS b, the overall load of computing tasks requiring any chunk in set $\mathcal{I}_{e,k}^{(J_{e,k}^i)}$ is given by $\sum_{i \in \mathcal{I}_{e,k}^{(J_{e,k}^i)}} \tilde{x}_{b,k}^{(i)}$ where $\tilde{x}_{b,k}^{(i)} = p_{b,k}^i \sum_{n \in \mathcal{N}} x_{b,k}^n$ is the load of computing tasks requiring chunk i in the coverage of BS b in time interval k. The load of computing tasks that request chunk $i \in \mathcal{I}_{e,k}$ in the coverage of BS b and are not assigned to S-BS e in time interval k, denoted by $P_{b,e,k}^i$, is as follows:

$$P_{b,e,k}^i =$$
$$\begin{cases} 0, \text{ if } \sum_{i \in \mathcal{I}_{e,k}^{(J_{e,k}^i)}} \tilde{x}_{b,k}^{(i)} \leq M_e; \\ \min\left\{ \tilde{x}_{b,k}^{(i)}, \sum_{i \in \mathcal{I}_{e,k}^{(J_{i,e,k})}} \tilde{x}_{b,k}^{(i)} - M_e \right\}, \text{ otherwise.} \end{cases} \quad (30)$$

where $M_e = \lfloor \frac{\tau^p C_e}{\alpha \beta} \rfloor$ is the maximum load of computing tasks that can be assigned to S-BS e with satisfying the computing capacity. In Eq. (30), if S-BS e has sufficient computing resource for executing all computing tasks requiring chunk i, i.e., $\sum_{\mathcal{I}_{e,k}^{(J_{e,k}^i)}} \tilde{x}_{b,k}^{(i)} \leq M_e$, no computing task requiring chunk i needs to be assigned to an S-NAP or the S-CN; Otherwise, a certain load of computing tasks requiring chunk i cannot be processed at S-BS e, which should be assigned to an S-NAP or the S-CN.

The overall load of computing tasks that request chunk i and are not assigned to any S-BS e within the service coverage of S-NAP e' in time interval k is given by $\sum_{b \in \mathcal{B}_{e'}} P_{b,e,k}^i$. The effective request ratio of chunk i for S-NAP e' in time interval k is as follows:

$$q_{e',k}^i = \frac{\sum_{b \in \mathcal{B}_{e'}} P_{b,e,k}^i}{\sum_{b \in \mathcal{B}_{e'}} \sum_{\mathcal{N}} \tilde{x}_{b,k}^n}, \ \forall i \in \mathcal{I}, e' \in \mathcal{E}^{\text{nap}}, k \in \mathcal{K}. \quad (31)$$

S-NAP e' stores $|\mathcal{I}_{e',k}|$ chunks with the largest values of $q_{e',k}^i$.

References

[1] SHEN X, GAO J, WU W, et al. AI-assisted network-slicing based next-generation wireless networks[J]. IEEE Open Journal of Vehicular Technology, 2020, 1(1): 45-66.

[2] SHILLAKER S, PIETZUCH P. Faasm: Lightweight isolation for efficient stateful serverless computing[C]// USENIX Annual Technical Conference. Berkeley: USENIX Association, 2020.

[3] BARCELONA-PONS D, SÁNCHEZ-ARTIGAS M, PARÍS G, et al. On the FaaS track: Building stateful distributed applications with serverless architectures[C]// Proceedings of International Middleware Conference. New York: ACM, 2019.

[4] European Telecommunications Standards Institute. White paper: MEC in 5G networks[R]. Sophia Antipolis: ETSI, 2018.

[5] KRÓL M, MASTORAKIS S, ORAN D, et al. Compute first networking: Distributed computing meets ICN[C]// Proceedings of ACM Conference on Information-Centric Networking. New York: ACM, 2019.

[6] Internet Engineering Task Force. Framework of Compute First Networking (CFN)[R]. Fremont: IETF, 2019.

[7] WANG K, CHEN W, LI J, et al. Joint task offloading and caching for massive MIMO-aided multi-tier computing networks[J]. IEEE Transactions on Communications, 2022, 70(3): 1820-1833.

[8] WU H, CHEN J, ZHOU C, et al. Learning-based joint resource slicing and scheduling in space-terrestrial integrated vehicular networks[J]. Journal of Communications and Information Networks, 2021, 6(3): 208-223.

[9] HWANG K, SHI Y, BAI X. Scale-out vs. scale-up techniques for cloud performance and productivity[C]// IEEE International Conference on Cloud Computing Technology and Science. Piscataway: IEEE Press, 2014.

[10] SHEN X, GAO J, WU W, et al. Holistic network virtualization and pervasive network intelligence for 6G [J]. IEEE Communications Surveys & Tutorials, 2022, 24(1): 1-30.

[11] YU Q, REN J, ZHOU H, et al. A cybertwin based network architecture for 6G[C]//6G Wireless Summit. Piscataway: IEEE Press, 2020.

[12] TAO F, ZHANG H, LIU A, et al. Digital twin in industry: State-of-the-art[J]. IEEE Transactions on Industrial Informatics, 2018, 15(4): 2405-2415.

[13] WANG S, RAN C. Rethinking cellular network planning and optimization[J]. IEEE Wireless Communications, 2016, 23(2): 118-125.

[14] LI M, GAO J, ZHOU C, et al. Slicing-based artificial intelligence service provisioning on the network edge: Balancing AI service performance and resource consumption of data management[J]. IEEE Vehicular Technology Magazine, 2021, 16(4): 16-26.

[15] CHEN X. Decentralized computation offloading game for mobile cloud computing[J]. IEEE Transactions on Parallel and Distributed Systems, 2015, 26(4): 974-983.

[16] HE H, SHAN H, HUANG A, et al. Edge-aided computing and transmission scheduling for LTE-U-enabled IoT[J]. IEEE Transactions on Wireless Communications, 2020, 19(12): 7881-7896.

[17] PASTERIS S, WANG S, HERBSTER M, et al. Service placement with provable guarantees in heterogeneous edge computing systems[C]//IEEE International Conference on Computer Communications. Piscataway: IEEE Press, 2019.

[18] OUYANG T, ZHOU Z, CHEN X. Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing[J]. IEEE Journal on Selected Areas in Communications, 2018, 36(10): 2333-2345.

[19] KUANG Z, LI L, GAO J, et al. Partial offloading scheduling and power allocation for mobile edge computing systems[J]. IEEE Internet of Things Journal, 2019, 6(4): 6774-6785.

[20] LERA I, GUERRERO C, JUIZ C. Availability-aware service placement policy in fog computing based on graph partitions[J]. IEEE Internet of Things Journal, 2018, 6(2): 3641-3651.

[21] CHEN L, XU J. Budget-constrained edge service provisioning with demand estimation via bandit learning [J]. IEEE Journal on Selected Areas in Communications, 2019, 37(10): 2364-2376.

[22] YU S, LANGAR R, FU X, et al. Computation offloading with data caching enhancement for mobile edge computing[J]. IEEE Transactions on Vehicular Technology, 2018, 67(11): 11098-11112.

[23] ZHOU C, WU W, HE H, et al. Deep reinforcement learning for delay-oriented IoT task scheduling in SAGIN [J]. IEEE Transactions on Wireless Communications, 2021, 20(2): 911-925.

[24] CHENG N, LYU F, QUAN W, et al. Space/aerial-assisted computing offloading for IoT applications: A learning-based approach[J]. IEEE Journal on Selected Areas in Communications, 2019, 37(5): 1117-1129.

[25] HU S, SHI W, LI G. CEC: A containerized edge computing framework for dynamic resource provisioning [J]. IEEE Transactions on Mobile Computing, 2022, Early Access.

[26] ZHANG J, CHEN S, WANG X, et al. Dynamic reservation of edge servers via deep reinforcement learning for connected vehicles[J]. IEEE Transactions on Mobile Computing, 2022, Early Access.

[27] YIN H, ZHANG X, LIU H, et al. Edge provisioning with flexible server placement[J]. IEEE Transactions on Parallel and Distributed Systems, 2016, 28(4): 1031-1045.

[28] ZHOU Z, YU S, CHEN W, et al. CE-IoT: Cost-effective cloud-edge resource provisioning for heterogeneous IoT applications[J]. IEEE Internet of Things Journal, 2020, 7(9): 8600-8614.

[29] DAYARATHNA M, WEN Y, FAN R. Data center energy consumption modeling: A survey[J]. IEEE Communications Surveys & Tutorials, 2015, 18(1): 732-794.

[30] CHEN L, ZHOU S, XU J. Computation peer offloading for energy-constrained mobile edge computing in small-cell networks[J]. IEEE/ACM Transactions on Networking, 2018, 26(4): 1619-1632.

[31] RUIZ-ALVAREZ A, HUMPHREY M. A model and decision procedure for data storage in cloud computing [C]//IEEE International Symposium on Cluster, Cloud and Internet Computing. Piscataway: IEEE Press, 2012.

[32] PARK J, CHOU P, HWANG J. Rate-utility optimized streaming of volumetric media for augmented reality [J]. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 2019, 9(1): 149-162.

[33] PASCHOS G S, IOSIFIDIS G, TAO M, et al. The role of caching in future communication systems and networks [J]. IEEE Journal on Selected Areas in Communications, 2018, 36(6): 1111-1125.

[34] DAI J, HU Z, LI B, et al. Collaborative hierarchical caching with dynamic request routing for massive content distribution[C]//IEEE International Conference on Computer Communications. Piscataway: IEEE Press, 2012.

[35] POULARAKIS K, TASSIULAS L. On the complexity of optimal content placement in hierarchical caching networks[J]. IEEE Transactions on Communications, 2016, 64(5): 2092-2103.

[36] ZHANG F, LIU G, FU X, et al. A survey on virtual machine migration: Challenges, techniques, and open issues[J]. IEEE Communications Surveys & Tutorials, 2018, 20(2): 1206-1243.

[37] KELLERER H, PFERSCHY U, PISINGER D. Multidimensional knapsack problems[M]. Berlin: Springer, 2004.

[38] LOVÁSZ L, PLUMMER M D. Matching theory[M]. Rhode Island: American Mathematical Society, 2009.

[39] CYGAN M, KOWALIK Ł, WYKURZ M. Exponential-time approximation of weighted set cover[J]. Information Processing Letters, 2009, 109(16): 957-961.

[40] CLERC M, KENNEDY J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(1): 58-73.

[41] KOCH G, ZEMEL R, SALAKHUTDINOV R, et al. Siamese neural networks for one-shot image recognition [C]//International Conference on Machine Learning. New York: ACM, 2015.

[42] NOURUZI A, ZAKERI A, JAVAN M R, et al. Online service provisioning in NFV-enabled networks using deep reinforcement learning[J]. IEEE Transactions on Network and Service Management, 2022, Early Access.

Conghao Zhou (S'19) received the B.Eng. degree from Northeastern University, Shenyang, China, in 2017 and the M.Sc. degree from University of Illinois at Chicago, Chicago, IL, USA, in 2018. He is currently working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research interests include space-air-ground integration networks and machine learning in wireless networks.
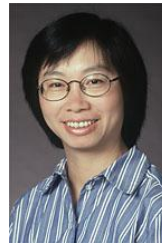
Jie Gao [corresponding author] (S'13-M'17-SM'21) received the B.Eng. degree in electronics and information engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2007, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of Alberta, Edmonton, AB, Canada, in 2009 and 2014, respectively. He was a postdoctoral fellow with Toronto Metropolitan University, Toronto, ON, Canada, from 2017 to 2019, a research associate with the University of Waterloo, Waterloo, ON, Canada, from 2019 to 2020, and an assistant professor with the Department of Electrical and Computer Engineering, Marquette University, Milwaukee, WI, USA, from 2020 to 2022. Dr. Gao is currently an assistant professor in the School of Information Technology, Carleton University. His research interests include machine learning for communications and networking, network virtualization and digital twins, Internet of Things (IoT) and industrial IoT solutions, and next-generation wireless networks in general. He is a senior member of IEEE, the lead associate editor for the Vehicular Society Section within IEEE Access, and an editor for Peer-to-Peer Networking and Applications (Springer).

Mushu Li (S'18-M'21) received the Ph.D. degree in Electrical and Computer Engineering from the University of Waterloo, Canada, in 2021. She received the B.Eng. degree from the University of Ontario Institute of Technology (UOIT), Canada, in 2015, the M.A.Sc. degree from Ryerson University, Canada, in 2017. She is currently a Postdoctoral Fellow at Toronto Metropolitan University, ON, Canada. She was a Postdoctoral Fellow with the University of Waterloo, ON, Canada, from 2021 to 2022. Her research interests include mobile edge computing, the system optimization in wireless networks, and machine learning-assisted network management. She was the recipient of Natural Science and Engineering Research Council of Canada (NSERC) Postdoctoral Fellowship (2022), NSERC Canada Graduate Scholarship (2018), and Ontario Graduate Scholarship (2015 and 2016).

Xuemin (Sherman) Shen (M'97-SM'02-F'09) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990. He is a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular networks. Dr. Shen is a registered Professional Engineer of Ontario, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, a Chinese Academy of Engineering Foreign Member, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society.

Dr. Shen received the Canadian Award for Telecommunications Research from the Canadian Society of Information Theory (CSIT) in 2021, the R.A. Fessenden Award in 2019 from IEEE, Canada, Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) in 2019, James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, Joseph LoCicero Award in 2015 and Education Award in 2017 from the IEEE Communications Society, and Technical Recognition Award from Wireless Communications Technical Committee (2019) and AHSN Technical Committee (2013). He has also received the Excellent Graduate Supervision Award in 2006 from the University of Waterloo and the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. He served as the Technical Program Committee Chair/Co-Chair for IEEE Globecom' 16, IEEE Infocom'14, IEEE VTC'10 Fall, IEEE Globecom'07, and the Chair for the IEEE Communications Society Technical Committee on Wireless Communications. Dr. Shen is the President of the IEEE Communications Society. He was the Vice President for Technical & Educational Activities, Vice President for Publications, Member-at-Large on the Board of Governors, Chair of the Distinguished Lecturer Selection Committee, Member of IEEE Fellow Selection Committee of the ComSoc. Dr. Shen served as the Editor-inChief of the IEEE IoT JOURNAL, IEEE Network, and IET Communications.
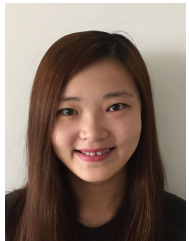
Weihua Zhuang (M'93-SM'01-F'08) received the B.Sc. and M.Sc. degrees from Dalian Marine University, China, and the Ph.D. degree from the University of New Brunswick, Canada, all in electrical engineering. She is a University Professor and a Tier I Canada Research Chair in Wireless Communication Networks at University of Waterloo, Canada. Her research focuses on network architecture, algorithms and protocols, and service provisioning in future communication systems. She is the recipient of 2021 Women's Distinguished Career Award from IEEE Vehicular Technology Society, 2021 Technical Contribution Award in Cognitive Networks from IEEE Communications Society, 2021 R.A. Fessenden Award from IEEE Canada, and 2021 Award of Merit from the Federation of Chinese Canadian Professionals in Ontario. She was the Editor-in-Chief of the IEEE Transactions on Vehicular Technology from 2007 to 2013, General Co-Chair of 2021 IEEE/CIC International Conference on Communications in China (ICCC), Technical Program Chair/Co-Chair of 2017/2016 IEEE VTC Fall, Technical Program Symposia Chair of 2011 IEEE Globecom, and an IEEE Communications Society Distinguished Lecturer from 2008 to 2011. She is an elected member of the Board of Governors and the Executive Vice President of the IEEE Vehicular Technology Society. Dr. Zhuang is a Fellow of the IEEE, Royal Society of Canada, Canadian Academy of Engineering, and Engineering Institute of Canada.