Parallelism-Aware Service Function Chain Placement for Delay-sensitive IoT Applications with VNF Reuse in Mobile Edge Computing

***Abstract*—With the rise of Network Function Virtualization technology, Service Function Chain (SFC), composed of Virtual Network Functions (VNFs), has become a mainstream service form for IoT applications in Mobile Edge Computing network. However, the delay-sensitive and resource-hungry nature of IoT applications poses great challenges for SFC placement. Besides, energy consumption has to be taken into consideration during SFC placement. To address these challenges, we integrate Network Function Parallelism and VNF Reuse into MEC network. Specifically, we formulate the Parallelism-aware Optimization for Delay-sensitive SFC Placement with VNF Reuse (PODPR) and design a heuristic Parallelism and VNF Reuse-based SFC Placement Solution (PRPS). The proposed PRPS transforms candidate routing paths into a path tree structure and utilizes greedy-based backtracking pruning algorithm to search for placement strategy with the lowest energy consumption. Simulation results demonstrate that PRPS outperforms baselines in terms of energy consumption, service delay, and admission rate.**

***Index Terms*—Mobile Edge Computing, Internet of Things, Service Function Chain, SFC Placement, Network Function Parallelism, VNF Reuse**

I. Introduction

The rise of 5G mobile communication technology has boomed the development of Internet of Things (IoT) applications, such as smart grid, Industry 4.0, and connected cars [1]. These applications typically need to process large volumes of data with demanding computing resources in real time. However, IoT devices are strictly constrained by limited resource capacities and remote cloud servers usually lead to extremely high transmission delay. Neither of them is suitable for serving delay-sensitive and resource-hungry IoT applications [2]. To address this issue, Mobile Edge Computing (MEC) has emerged as an enabling technique to support real-time IoT applications by bringing cloud services within the proximity of users [3]. Further, the adoption of Network Function Virtualization (NFV) offers new flexibility in providing with IoT services in MEC network [4]. The technology decouples network functions from proprietary hardware and implements them as Virtual Network Functions (VNFs) that run in virtual machines. In this context, an IoT application is represented by a set of ordered VNFs, i.e., Service Function Chain (SFC) and SFC placement deals with how to place the ordered VNFs on edge servers and how to route data packets among these VNFs [5].

Despite the advantages of applying NFV in MEC network, there are several challenges for SFC placement: (1) *Delay-sensitivity of IoT applications*: IoT applications have stringent requirements for low latency. The processing delay resulting from consecutively running VNFs in a SFC can be significant and comparable to transmission delay, which will become a bottleneck for latency optimization [6]. (2) *Limited resources of edge servers*: Compared with cloud servers, the resources of edge servers are very limited. When multiple IoT users request services to edge servers, their requests may not be satisfied because of resource hunger [7]. As a result, resource utilization needs to be improved during SFC placement. (3) *Energy consumption of SFC placement*: Energy consumption is crucial for the sustainable development of IoT. The processing and transmission of IoT data will consume system energy, making it a challenge to identify an energy-efficient SFC placement strategy in a short time [8].

We aim to address these issues. For challenge (1), Network Function Parallelism (NFP) will be introduced to mitigate the impact caused by processing delay. NFP enables multiple VNFs to run in a parallel way on the same server [9], thereby decreasing the total processing latency of SFC and alleviating the bottleneck of latency optimization. For challenge (2), we will employ Multi-Tenancy technology to realize the reusability of VNF instances, which allows the sharing of computing resources allocated to a VNF instance among different users [10]. VNF reuse can improve resource utilization and reduce initialization time. Based on the above two factors, we design a heuristic energy-efficient SFC placement solution to address challenge (3). To strengthen the potential benefits of NFP and VNF reuse, we propose a greedy-based backtracking pruning placement algorithm that deploys parallelizable VNFs on the same server as much as possible and maximizes the reuse of existing VNF instances.

Fig. 1 presents a detailed example of deploying two SFCs on MEC network, where each server is limited to creating only two VNF instances. Traditional SFC placement strategy, as illustrated in Fig. 1a, selects the shortest paths for both SFC1 and SFC2 to minimize total processing delay (TPD) and total transmission delay (TTD). In this case, if we place parallelizable VNFs on the same server, TPD can be further reduced. Additionally, if SFC2 reuses the VNF4 and VNF5 instances of SFC1, more server resources can be saved. As a result, our proposed solution, depicted in Fig. 1b, incorporates NFP and VNF reuse into MEC network. Specifically, we place parallelizable VNF1 and VNF2 on server *B*, which reduces TPD of SFC1 from 30ms to 24ms. However, the reuse of VNF4 and VNF5 instances increases TTD of SFC2 from

30ms to 35ms. Therefore, it is a great challenge to obtain an optimal SFC placement strategy for delay-sensitive IoT applications with NFP and VNF reuse, while considering energy consumption.

To the best of our knowledge, we are the first to investigate energy-efficient SFC placement for delay-sensitive IoT applications with NFP and VNF reuse in MEC network. Main contributions of this paper are as follows:

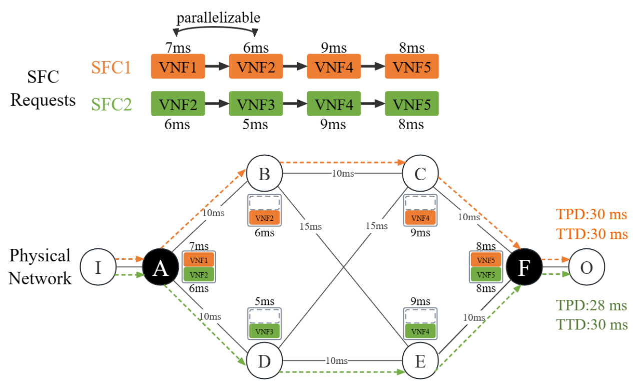
- We formulate the Parallelism-aware Optimization for Delay-sensitive SFC Placement with VNF Reuse in MEC (PODPR), aiming to minimize energy consumption.
- We design a heuristic Parallelism and VNF Reuse-based SFC Placement Solution (PRPS). The proposed PRPS consists of the Maximum Profit Path Tree Creation Algorithm (MPPT-CA) and Greedy-based Backtracking Pruning Placement Algorithm (GBP-PA).
- We evaluate the performance of the proposed PRPS through simulations. Experimental results demonstrate that PRPS outperforms baselines in terms of energy consumption, service delay, and admission rate.

The rest of this paper is organized as follows. Section 2 presents related works. Section 3 formulates the PODPR and proves the problem is NP-hard. In Section 4, we propose a heuristic solution PRPS to solve PODPR. Simulation results and performance analysis are presented in Section 5. Section 6 concludes this paper.

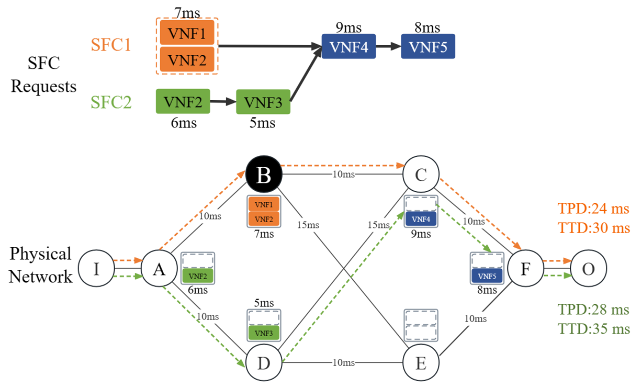
II. Related Works

A. General SFC Placement

Numerous studies have investigated SFC placement without NFP and VNF reuse, focusing on various performance objectives. [1] proposes a cooperative dual-agent deep reinforcement learning algorithm to optimize partial offloading and SFC mapping, aiming to minimize average cost. [11] proposes a constant approximation algorithm based on Next Fit (NF) strategy to address SFC deployment problem and utilizes a double spanning tree algorithm to handle the network topology and the corresponding flow routing problem. [12] studies SFC deployment in Multi-tier computing and proposes an online heuristic algorithm based on binary search to maximize system revenue. Dynamic programming is used in [13] to solve SFC placement with and without resiliency concerns. [14] presents a mobility-aware, priority-driven, and ant colony optimization-based service placement model. The model prioritises applications based on their criticality and aims to minimize their latency. [15] models SFC placement as a Markov decision process and leverages reinforcement learning with an actor-critic approach for MEC nodes to learn complex placement and chaining policies. Physical programming method is used in [16] to solve multi-objective SFC placement, which allows SFC partitioning and embedding over multiple domains. [17] employs mixed-integer linear programming(MILP) technique to solve delay-sensitive SFC placement with the aim of minimizing VNF migrations.



(a) Traditional SFC placement strategy



(b) Our SFC placement strategy with NFP and VNF reuse

Fig. 1: Two candidate SFC placement strategies

B. SFC Placement with NFP

Network Function Parallelism (NFP) technology has recently emerged as an effective approach for optimizing service delay [18]. For the first time, [19] introduces an Improved Service Function Graph to reflect the parallelization relationships between VNFs. [5] designs an improved cuckoo search algorithm for the deployment of delay-sensitive SFC based on parallelization. [20] offers an latency factor-based parallelism-aware SFC optimization algorithm to jointly optimize processing and propagation delay. Various techniques, including Network Function Distribution, NFP, and optimal resource allocation have been utilized in [21] to place requested VNFs with minimum cost and energy consumption. Viterbi dynamic programming algorithm is used in [22] to estimate bottleneck resource occupation and adjust SFC processing order for an optimal solution. [7] solves the parallelized SFC deployment problem as an Integer Linear Program that minimizes energy consumption while ensuring reliability and delay constraints. [23] only parallelizes functions if it can indeed effectively reduce latency after considering VNF placements and additional parallelization cost. To strengthen the potential benefits of NFP, [9] formulates the fairness-aware SFC placement problem with the aim of maximizing system throughput and proposes a relaxation-based generalized benders algorithm.

TABLE I: Key Notations

Notation	Description
$\boldsymbol{V}$	Set of edge servers
$\boldsymbol{E}$	Set of physical links
$\boldsymbol{F}$	Set of virtualized network functions
$\boldsymbol{U}$	Set of IoT users
SFC_u	Service function chain of $u \in \boldsymbol{U}$
$D_u^{service}$, E_u	End-to-end delay and energy consumption of SFC_u
$x_{u,i}^{v}$	The ith VNF of SFC_u is placed on $v \in \boldsymbol{V}$ for $x_{u,i}^{v}$=1
$y_{u,i}^{v,k}$	The ith VNF of SFC_u reuses the kth VNF instance of same type on $v \in \boldsymbol{V}$ for $y_{u,i}^{v,k}$=1
z_{s_u,d_u}^{i}	SFC_u adopts the ith connected path between $s_u \in \boldsymbol{V}$ and $d_u \in \boldsymbol{V}$ to transmit data for z_{s_u,d_u}^{i}=1
$q_{s_u,d_u}^{i,e}$	Link $e \in \boldsymbol{E}$ is included in the ith connected path between $s_u \in \boldsymbol{V}$ and $d_u \in \boldsymbol{V}$ for $q_{s_u,d_u}^{i,e}$=1

C. SFC Placement with VNF Reuse

In addition to NFP, extensive investigations have been conducted on SFC placement under the assumption that resources allocated to each VNF instance can be share between different requests. [4] formulates a novel subchain-aware NFV service placement optimization model and strives to reuse existing subchains of consecutive network functions. [24] designs a two-stage latency-aware SFC deployment scheme to optimize the resource utilization of edge servers and links by maximizing the reuse of VNF instances. [25] proposes two efficient heuristics, namely Greedy-based and Tabu search-based algorithms to solve SFC placement with VNF reuse. [10] proposes a prediction-based service placement scheme with VNF sharing at the edge, which utilizes the predicted required resources in a defined lookahead window to minimize the rejection rate of premium services. [26] leverages mixed-integer linear programming techniques to solve the multi-objective problem of joint user association, SFC placement, and resource allocation with VNF sharing. Steiner tree algorithm is used in [27], [28] to obtain the placement strategies that reuse existing VNF instances with low cost for multicast SFCs. [29] designs a centrality-based ranking method to map VNFs to existing instances and utilizes Lagrange relaxation based aggregated cost algorithm to find a shortest path with minimum cost.

Unlike the aforementioned works that utilize either NFP or VNF reuse to address SFC placement, we will take a comprehensive insight into SFC placement for delay-sensitive IoT applications with both technologies, aiming to minimize system energy consumption.

III. System Model and Problem Formulation

In this section, we introduce the MEC network model and formulate the optimization of SFC placement for IoT applications. Key notations are listed in Table I.

A. System Model

1) Network Function Parallelism: Different network functions perform diverse operations on data including reading, writing, dropping, and so on. When operations of multiple network functions will not change data content simultaneously (e.g., writing and dropping, writing and writing), Network Function Parallelism (NFP) can be introduced to decrease SFC processing delay by executing these functions in a parallel way on the same server [20]. In this paper, we employ NF parallelism Identification Algorithm proposed in [18] to decide whether two VNFs can be executed in parallel.

2) Physical Network with VNF Reuse: We represent the MEC network as an undirected graph $\boldsymbol{G}$ = ($\boldsymbol{V}$, $\boldsymbol{E}$), where $\boldsymbol{V}$ is the set of edge servers and $\boldsymbol{E}$ is the set of physical links connecting servers. Each server $v \in \boldsymbol{V}$ has a computing resource capacity denoted by C_v, which is utilized to create VNF instances. The creation of a new VNF instance on v will generate initialization delay D_v. Similarly, each link $e \in \boldsymbol{E}$ has a bandwidth resource capacity denoted by B_e. The delay of transmitting data along e is denoted as D_e. The set of connected paths between two servers $v_i \in \boldsymbol{V}$ and $v_j \in \boldsymbol{V}$ is denoted as P_{v_i,v_j}=\{$P^1_{v_i,v_j}$, $P^2_{v_i,v_j}$, ...\}.

The set of different types of VNFs is denoted as $\boldsymbol{F}$. Each function $f \in \boldsymbol{F}$ generates processing delay D_f and requires C_f server resources to create a type-f VNF instance. We assume all VNF instances can be shared between different users, so that their processing capabilities can be maximally reused. We use μ_f to denote the maximum data processing capacity of a type-f VNF instance, which means IoT users can resue the VNF instance if the residual processing capacity is sufficient to process their data flow. $Ins_{v,f}$=\{$ins^1_{v,f}$, $ins^2_{v,f}$, ...\} represents all type-f VNF instances on server $v \in \boldsymbol{V}$.

3) SFC Request of an IoT user: For each IoT user $u \in \boldsymbol{U}$, the service request is represented by a set of VNFs with certain sequences SFC_u=\{$f_{u,1}$, $f_{u,2}$, ...\}. I_u represents the length of SFC_u. We use $s_u \in \boldsymbol{V}$ to denote the source node, which is an IoT gateway that forwards data collected from sensors. $d_u \in \boldsymbol{V}$ denotes the destination node that often connects to a remote site for specific control tasks, such as opening/closing doors or switching lights. r_u represents the size of data flow and D_u represents the maximum tolerable end-to-end delay of SFC_u.

End-to-end delay has a significant impact on service quality of IoT applications, which consists of three parts: the processing delay, the transmission delay, and the initialization delay, defined as follows:

The Processing Delay: NFP enables some network functions to run in a parallel way on the same server. For convenience of calculation, we split SFC into a set of consecutively ordered parallel blocks, each of which consists of either a set of parallelizable VNFs or one VNF. We use PB_u to denote the set of parallel blocks of SFC_u, and pb^i_u to denote the ith block. Specifically, SFC1 in Fig. 1b contains three parallel blocks, where pb^1_u=\{VNF1,VNF2\}, pb^2_u=\{VNF4\}, pb^3_u=\{VNF5\}. The processing delay of each parallel block is determined by the maximum processing delay of VNF in the block, i.e., $D_{pb^i_u} = max_{f\in pb^i_u} D_f$. Therefore, the total processing delay of a SFC is the sum of processing delay of its parallel blocks:

$$D_u^{pro} = \sum_{pb^i_u \in PB_u} D_{pb^i_u}, \forall u \in \boldsymbol{U} \tag{1}$$

The Transmission Delay: We use a binary variable $z^i_{s_u,d_u}$ to denote whether the ith connected path $p^i_{s_u,d_u}$ between source node s_u and destination node d_u is selected to transmit user data (=1) or not (=0). The total transmission delay of a SFC is equal to the transmission delay of the selected path:

$$D_u^{tran} = \sum_{i=1}^{|P_{s_u,d_u}|} z^i_{s_u,d_u} \cdot D(p^i_{s_u,d_u}), \forall u \in \boldsymbol{U} \quad (2)$$

The Initialization Delay: The creation of a new VNF instance on server $v \in \boldsymbol{V}$ generates initialization delay D_v. We use a binary variable $x^v_{u,i}$ to indicate whether the ith VNF of SFC_u is placed on server v (=1) or not (=0), and $y^{v,k}_{u,i}$ to indicate whether $f_{u,i}$ reuses the kth existing type-$f_{u,i}$ VNF instance on v (=1) or not (=0). $\sum_{k=1}^{|Ins_{v,f_{u,i}}|} y^{v,k}_{u,i}$=0 indicates the absence of reusable type-$f_{u,i}$ VNF instances on v. Therefore, the total initialization delay of a SFC can be defined as:

$$D_u^{init} = \sum_{i=1}^{I_u} \sum_{v \in \boldsymbol{V}} x^v_{u,i} \cdot D_v \cdot (1 - \sum_{k=1}^{|Ins_{v,f_{u,i}}|} y^{v,k}_{u,i}), \forall u \in \boldsymbol{U} \quad (3)$$

In summary, for each IoT request SFC_u, end-to-end service delay can be defined as:

$$D_u^{service} = D_u^{pro} + D_u^{tran} + D_u^{init}, \forall u \in \boldsymbol{U} \quad (4)$$

Similarly, the total energy consumption of placing a SFC in MEC network comes from data processing, data transmission, and instance initialization, defined as follows:

The Processing Energy Consumption: The energy consumption generated from data processing by a type-f VNF instance is proportional to processing delay D_f. We assume that the processing power of $v \in \boldsymbol{V}$ is Pow_v, and the total processing energy consumption can be defined as:

$$E_u^{pro} = \sum_{i=1}^{I_u} \sum_{v \in \boldsymbol{V}} x^v_{u,i} \cdot Pow_v \cdot D_{f_{u,i}}, \forall u \in \boldsymbol{U} \quad (5)$$

The Transmission Energy Consumption: The energy consumption for transmitting IoT data by link $e \in \boldsymbol{E}$ is related to the transmission delay D_e. We assume that the transmission power of e is denoted as Pow_e, and the total transmission energy consumption can be defined as:

$$E_u^{tran} = \sum_{i=1}^{|P_{s_u,d_u}|} z^i_{s_u,d_u} \cdot (\sum_{e \in p^i_{s_u,d_u}} Pow_e \cdot D_e), \forall u \in \boldsymbol{U} \quad (6)$$

The Initialization Energy Consumption: The creation of a new type-f VNF instance on $v \in \boldsymbol{V}$ will consume initialization energy, which is related to the initialization delay D_v:

$$E_u^{init} = \sum_{i=1}^{I_u} \sum_{v \in V} x^v_{u,i} \cdot Pow_v \cdot D_v \cdot (1 - \sum_{k=1}^{|Ins_{v,f_{u,i}}|} y^{v,k}_{u,i}), \forall u \in \boldsymbol{U} \quad (7)$$

In summary, for each IoT request SFC_u, the total energy consumption can be defined as:

$$E_u = E_u^{pro} + E_u^{tran} + E_u^{init}, \forall u \in \boldsymbol{U} \quad (8)$$

B. Problem Formulation

Based on the model described above, we formulate the Parallelism-aware Optimization for Delay-sensitive SFC Placement with VNF Reuse in MEC, i.e., PODPR, which aims to minimize system energy consumption and optimize service delay of IoT applications:

$$Minimize \sum_{u \in \boldsymbol{U}} E_u \quad (9)$$

$$s.t.\ D_u^{service} \le D_u, \forall u \in \boldsymbol{U} \quad (10)$$

$$\sum_{v \in \boldsymbol{V}} x^v_{u,i} = 1, \forall u \in \boldsymbol{U}, \forall i \in [1, I_u] \quad (11)$$

$$\sum_{k=1}^{|Ins_{v,f_{u,i}}|} y^{v,k}_{u,i} \le 1, \forall u \in \boldsymbol{U}, \forall i \in [1, I_u], \forall v \in \boldsymbol{V} \quad (12)$$

$$\sum_{i=1}^{|P_{s_u,d_u}|} z^i_{s_u,d_u} = 1, \forall u \in \boldsymbol{U} \quad (13)$$

$$\sum_{f \in \boldsymbol{F}} |Ins_{v,f}| \cdot C_f \le C_v, \forall v \in \boldsymbol{V} \quad (14)$$

$$\sum_{u \in \boldsymbol{U}} \sum_{i=1}^{I_u} y^{v,k}_{u,i} \cdot r_u \le \mu_{ins^k_{v,f_{u,i}}}, \forall v \in \boldsymbol{V}, \forall k \in [1, |Ins_{v,f_{u,i}}|] \quad (15)$$

$$\sum_{u \in \boldsymbol{U}} \sum_{i=1}^{|P_{s_u,d_u}|} z^i_{s_u,d_u} \cdot q^{i,e}_{s_u,d_u} \cdot r_u \le B_e, \forall e \in \boldsymbol{E} \quad (16)$$

$$x^v_{u,i}, y^{v,k}_{u,i}, z^j_{s_u,d_u} \in \{0,1\}, \forall u \in \boldsymbol{U}, \forall i \in [1, I_u], \forall v \in \boldsymbol{V}, \forall k \in [1, |Ins_{v,f_{u,i}}|], \forall j \in [1, |P_{s_u,d_u}|] \quad (17)$$

The objective function is aimed to minimize the energy consumption of all IoT users. Equation (10) ensures the fulfillment of low-latency requirements for IoT users. Equation (11) ensures that each VNF can only be placed on one server, and (12) indicates that if $f_{u,i}$ is placed on server v, it either reuses an existing instance or creates a new instance. Equation (13) ensures that only one connected path between s_u and d_u will be selected to transmit data. Equation (14)-(16) define the resource constraints of servers, VNF instances, and links.

C. NP-hardness of PODPR

In this section, we prove the NP-hardness of PODPR by reducing it to the Knapsack problem, a well-known NP-hard problem. In the Knapsack problem, given a fixed-capacity knapsack and a set of items with weights and values, the goal is to select items that maximize the total value without exceeding the knapsack's capacity.

We consider a special form of PODPR in MEC, where there is a single edge server with a fixed computing resource capacity. Each SFC request consists of a single VNF and VNF instances are not reusable. By transforming PODPR into the Knapsack problem, the edge server represents the knapsack, and each user's service request becomes an item. The objective is to minimize system energy consumption while respecting resource constraints, which can be transferred into maximizing the total value within the knapsack's limited capacity. Since the Knapsack problem is NP-hard, we can conclude that PODPR is also NP-hard.

IV. Algorithm Design

A. Parallelism and VNF Reuse-based SFC Placement Solution

Since PODPR is proved to be NP-hard, it is computational expensive to obtain the optimal strategy. By assuming that IoT requests arrive into MEC system one by one without the knowledge of future arrivals, our problem focuses on how to place and route a single SFC in MEC network. However, it is still tricky to solve the problem. Therefore, we design a heuristic Parallelism and VNF Reuse-based SFC Placement Solution (PRPS), which can obtain near-optimal strategies in a short time. PRPS consists of two parts: the Maximum Profit Path Tree Creation Algorithm (MPPT-CA), which selects candidate transmission paths based on profit values and transforms them into a path tree structure; and the Greedy-based Backtracking Pruning Placement Algorithm (GBP-PA), which is used to search for a placement strategy with the lowest energy consumption on the path tree.

1) Maximum Profit Path Tree Creation Algorithm: Due to the integration of NFP and VNF reuse, it is expected to place parallelizable VNFs on one server as much as possible and maximize the reuse of existing instances. Therefore, to strengthen the potential benefits of NFP and VNF reuse, we define the profit of a server $v \in \boldsymbol{V}$ as:

$$profit_v = \frac{|SFC_{sub}|}{I_u} \cdot \left(\frac{D_{sub} - D_{sub}^{'}}{D_{sub}}\right) \tag{18}$$

where SFC_{sub} represents the maximum set of consecutive VNFs on SFC_u that can reuse instances on server v. I_u is the length of SFC_u. $D_{sub}^{'}$ and D_{sub} represent the total processing delay of SFC_u with and without any parallelisms, respectively. D_{sub}-$D_{sub}^{'}$ defines the amount of processing delay saved by NFP. A higher profit value indicates that more VNF instances can be reused to serve SFC_u and a greater reduction in processing delay can be achieved with NFP.

Based on the profit of a server, we further define the profit of a connected path. For a given path, its quality depends on the profit values of all servers included in the path and the transmission delay. Therefore, the profit of a connected path p is defined as:

$$profit_p = \frac{\sum_{v \in p} profit_v}{D(p) \cdot |p|} \tag{19}$$

where $D(p)$ is the total transmission delay of path p, $\sum_{v \in p} profit_v / |p|$ equals to the average profit of all servers on the path. The higher the profit of a path, the more likely the placement strategy with this path to transmit data will be close to the optimal solution.

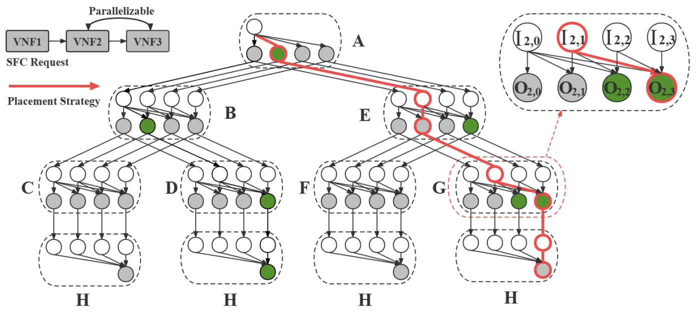


Fig. 2: The path tree with a candidate placement strategy.

For a given SFC request SFC_u, we select the top k paths with the highest profit values from the set of connected paths P_{s_u,d_u} as candidate transmission paths $\boldsymbol{P}$. Subsequently, a path tree PT_u is created based on these paths, the structure of which is illustrated in Fig. 2. Each branch from the root to a leaf node on PT_u corresponds to a connected path from source node s_u to destination node d_u. Each tree node is represented by a subgraph G_v, which consists of two sets of nodes, each containing I_u+1 nodes. Node $I_{i,j}$, where $j \in \{0, 1, 2, \cdots I_u\}$, represents an *Input* node, indicating that the first j VNFs on SFC_u are placed on the first i servers of current path. Node $O_{i,j}$, where $j \in \{0, 1, 2, \cdots I_u\}$, represents an *Output* node. Each *Input* node $I_{i,j}$ is connected to *Output* nodes $O_{i,j'}$, where $j^{'} \in \{j, j+1, \cdots, I_u\}$. The path from $I_{i,j}$ to $O_{i,j'}$ indicates that the j+1th, j+2th,..., $j^{'}$th VNFs are placed on current server. The green *Output* nodes $O_{i,j}$ indicate the presence of reusable type-$f_{u,j}$ VNF instances. The *Input* nodes of the root only contain $I_{0,0}$, while the *Output* nodes of leaf nodes only contain O_{i,I_u}. This is because we must deploy all VNFs on servers included in one specific connected path. The path tree structure merges overlapping subpaths from multiple connected paths, which can avoid duplicate searches and speeds up the search for feasible SFC placement strategies. Based on the path tree, we can decide how to place VNFs and how to route data simultaneously. Detailed steps for creating a path tree are shown in Algorithm 1.

Algorithm 1 selects k candidate paths from P_{s_u,d_u} based on bandwidth resources, total transmission delay, and profit values (line 1-2). Based on these paths, it creates each branch of PT_u with function ***createPathTree*** (line 3-6). It should be noted that if current subpath already exists on the path tree, we proceed to the next level of recursion (line 16). Otherwise, we create a new tree node before recursion (line 12-15).

2) Greedy-based Backtracking Pruning Placement Algorithm: The backtracking pruning algorithm is used to explore feasible SFC placement strategies based on the obtained path tree PT_u. To fully take advantage of the benefits of NFP and VNF reuse, we improve the traditional algorithm and propose a Greedy-based Backtracking Pruning Placement Algorithm (GBP-PA), which tries to place parallelizable VNFs on one server as much as possible and maximize the reuse of existing instances. Detailed steps are shown in Algorithm 2: S records the current placement strategy, *bestStrategy* represents current

Algorithm 1 MPPT-CA

```
Input: $P_{s_u,d_u}$: Set of connected paths between $s_u$ and $d_u$
Output: $PT_u$: The path tree
 1: Select paths $p \in P_{s_u,d_u}$ with enough bandwidth resources
    and low transmission delay, and add them into $P'_{s_u,d_u}$
 2: Select the first $k$ paths in $P'_{s_u,d_u}$ with the maximum profit
    values and add them into $\boldsymbol{P}$
 3: Construct the subgraph $G_{s_u}$ as root
 4: for each $p \in \boldsymbol{P}$ do
 5:     call createPathTree(p, 1, root)
 6: end for
 7: return root
 8: function createPathTree(p, curIndex, root)
 9:     if $curIndex \geq len(p)$ then
10:         return
11:     end if
12:     if the subgraph of p[curIndex] nextTreeNode does not
          exist in root.childs then
13:         Construct subgraph of p[curIndex] as nextTree-
            Node and connect each Output node of root with
            corresponding Input node of nextTreeNode
14:         add nextTreeNode into root.childs
15:     end if
16:     call createPathTree(p, curIndex+1, nextTreeNode)
```

Algorithm 2 GBP-PA

```
Input: $PT_u$: The path tree from $s_u$ to $d_u$
Output: bestStrategy: SFC placement strategy
 1: S = $\emptyset$, bestStrategy = $\emptyset$
 2: call greedyPlace($PT_u.I_{0,0}$, $PT_u$, 0, 0, S, bestStrategy)
 3: return bestStrategy
 4: function greedyPlace(IN, root, i, j, S, bestStrategy)
 5:     if $Delay(S) \geq D_u$ then
 6:         return
 7:     end if
 8:     if a leaf node is reached then
 9:         record path from IN to $O_{i,I_u}$
10:         if $EnergyCon(S) < EnergyCon(bestStrategy)$ then
11:             bestStrategy=S
12:         end if
13:         return
14:     end if
15:     for k in range(j+1,$I_u$) do
16:         if there exists a reusable instance for $f_{u,k}$ then
17:             use the reusable instance on root to serve $f_{u,k}$
18:             continue
19:         end if
20:         if $f_{u,k}$ is parallelizable with $f_{u,k-1}$ then
21:             create a new instance on root to serve $f_{u,k}$
22:             continue
23:         end if
24:         record path from IN to $O_{i,k-1}$
25:         for each nextN $\in$ root.childs do
26:             record path from $O_{i,k-1}$ to nextN.$I_{i+1,k-1}$
27:             call greedyPlace(nextN.$I_{i+1,k-1}$, nextN, i+1,
                                 k-1, S, bestStrategy)
28:         end for
29:     end for
```

best strategy, and *IN* is the *Input* node of current tree node. Algorithm 2 traverses all *Output* nodes connected to *IN*. If there are reusable VNF instances or current VNF can run in parallel with previous VNF, then we deploy it on current server (line 16-23). Otherwise, we traverse downwards (line 25-28). When reaching a leaf node, we place all remaining undeployed VNFs on it and update the best placement strategy based on energy consumption (line 8-14). Finally, we will obtain the SFC placement strategy with the lowest energy consumption while satisfying IoT requirements for low latency.

The SFC placement strategy shown in Fig. 2 with red lines selects A-E-G-H as routing path and places VNF1 on server *A*, VNF2 and VNF3 on server *G*. When the path goes into the *Input* node $I_{2,1}$ of *G*, we greedily deploy VNF2 and VNF3 on *G*. This decision is motivated by the parallelizability of VNF2 and VNF3, as well as the existence of reusable instances for both on *G*. Consequently, the path directly exits from the *Output* node $O_{2,3}$, skipping over $O_{2,1}$ and $O_{2,2}$.

B. Algorithm Analysis

This section gives a brief complexity analysis of our proposed PRPS. In Algorithm 1, creating a path tree based on the selected *k* paths requires a computational complexity of O(*k*·*PM*), where *PM* is the average length of these paths. In the worst-case scenario, the path tree is composed of *k*(*PM*-1)+1 subgraphs, indicating that the selected *k* paths have no overlapping subpaths. Each subgraph contains a strategy space of 2(2+I_u), where I_u denotes the length of SFC. Based on above analysis, Algorithm 2 has a time complexity of O(*k*·*PM* · $2I_u$). However, due to the application of greedy algorithm and pruning operation into Algorithm 2, exhaustive exploration of every potential strategy on the path tree is unnecessary. Consequently, the actual time complexity of Algorithm 2 is significantly lower than O(*k*·*PM* · $2I_u$).

In conclusion, the worst-case performance bound of PRPS, composed of Algorithm 1 and 2, is O(*k*·*PM*(1+2·I_u)).

V. Performance Evaluation

This section describes the parameter settings for the simulation experiments and compares the performance of PRPS with other SFC placement solutions.

A. Simulation Settings

In simulation experiments, we randomly generate five MEC networks of different sizes, with the number of servers in each network being [10, 20, 30, 40, 50]. The computing resource capacity of each edge server is randomly set within the range of [2000, 5000] MIPS and the bandwidth resource capacity of each link is randomly set within the range of [2000, 5000] Mbps. 11 kinds of common network functions are considered in this paper. The required computing resources for creating a new VNF instance is randomly set within the range of [10, 70] MIPS and the maximum data processing capacity of each instance is randomly set within the range of [500, 1000] Mbps. For each SFC request, the length of SFC is randomly set within the range of [3,7] and the size of data flow is randomly set within the range of [20,50] Mbps.

Baselines used in the experiments are as follows:

- **VDA**: designs an Improved Service Function Graph (I-SFG) to represent various SFC parallel strategies and selects the deployment position for VNFs based on the I-SFG with lowest server delay [19].
- **SAP**: maximizes the reuse of subchains composed of multiple VNF instances instead of individual instances and utilizes Tabu search algorithm to search for near-optimal SFC placement strategies [4].
- **OLBS**: uses a binary search approach to divide a SFC into two sub-SFCs based on delay constraints. The placement strategies for the sub-SFCs are then determined separately and merged to obtain the final strategy [12].

B. Simulation Results

1) Ablation Study: In this section, we investigate the impact of integrateing NFP and VNF reuse on the performance of PRPS. We introduce PRPS without NFP (PRPS_withoutNFP), without VNF reuse (PRPS_withoutReuse), and without both (PRPS_withoutAll) to compare them with our proposed PRPS in dealing with 500 IoT user requests in simulated networks of different sizes.

Fig. 3-Fig. 5 show the obtained average energy consumption, average service delay, and admission rate by four algorithms respectively. Fig. 3 shows that the average energy consumption of PRPS and PRPS_withoutNFP is about 20% lower than that of the other two algorithms. This reduction can be attributed to the introduction of VNF reuse in these two algorithms, which can save energy from the creation of new VNF instances. Fig. 4 demonstrates that the average delay of PRPS and PRPS_withoutReuse is at least 10% lower than that of PRPS_withoutNFP and PRPS_withoutAll. This improvement can be attributed to the introduction of NFP, which effectively reduces the total processing delay of SFCs and consequently diminishes end-to-end delay. It should be noted that the greedy-based placement algorithm employed in PRPS contributes to the lowest energy consumption and service delay by placing parallelizable VNFs on the same server as much as possible and maximizing the reuse of existing VNF instances. Fig. 5 shows that the admission rates of PRPS and PRPS_withoutNFP are also the highest, ranging from 80% to 100%. It indicates that algorithms with VNF reuse can accommodate more users in networks of different sizes. PRPS_withoutAll performs the worst in all aspects.

These results indicate that the application of NFP and VNF reuse into SFC placement can effectively reduce service delay and energy consumption, while fully utilizing system resources to serve more users.

2) Performance Comparison with Baselines: This section compares PRPS with three other SFC placement solutions, namely VDA, SAP, and OLBS, from four perspectives: average energy consumption, average service delay, admission rate, and running time.

Fig. 6-Fig. 9 present the performance of four algorithms in dealing with 500 IoT users in networks of different sizes. Fig. 6 shows that PRPS achieves the lowest energy consumption, averaging 10% lower than that of SAP. Additionally, Fig. 8 demonstrates that PRPS achieves the highest admission rate, averaging 5% higher than that of SAP. This is because PRPS reuses existing VNF instances as much as possible, which can reduce energy consumption associated with creating new instances and also save more system resources to serve IoT users. Fig. 7 demonstrates that PRPS achieves the lowest service delay, averaging 10% lower than that of VDA. This can be attributed to the introduction of NFP, which reduces processing delay by enabling VNFs to run in a parallel way.

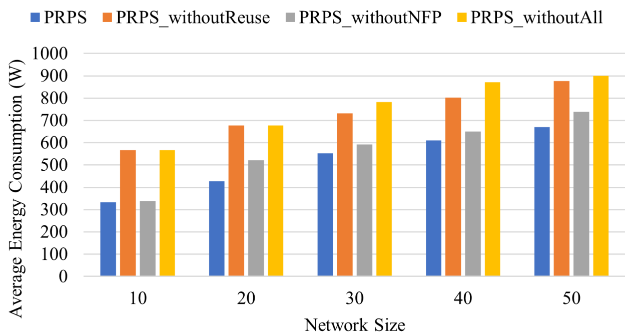


Fig. 3: Average energy consumption.

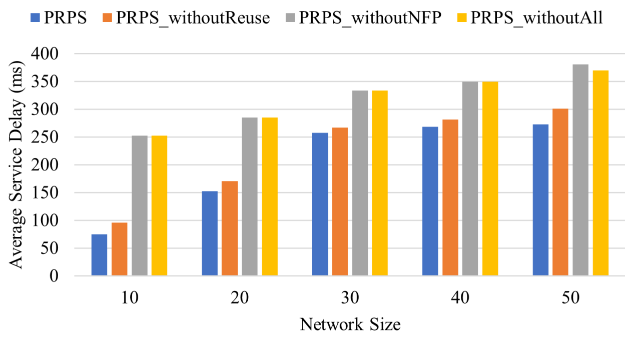


Fig. 4: Average service delay.

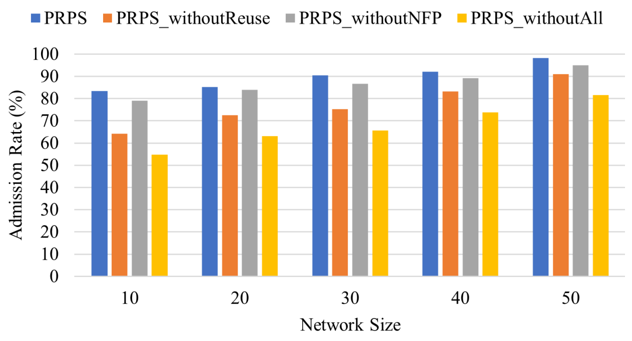


Fig. 5: Admission rate.

Additionally, VNF reuse can also decrease initialization delay by avoiding the creation of new VNF instances, thereby reducing end-to-end delay. Although the integration of NFP and VNF reuse may increase algorithm complexity, PRPS has a similar running time to that of heuristic VDA and SAP, as shown in Fig. 9, due to the application of greedy algorithm and pruning operation. However, PRPS achieves lower energy consumption and service delay than VDA and SAP. OLBS requires the most time to find the optimal placement strategy because it exhaustively explores all feasible strategies.

These results indicate that PRPS can effectively reduce service delay and energy consumption, while accommodating more IoT users in MEC network after a short period of time.

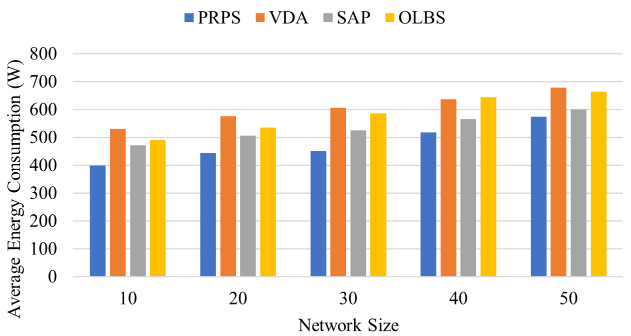


Fig. 6: Average energy consumption.

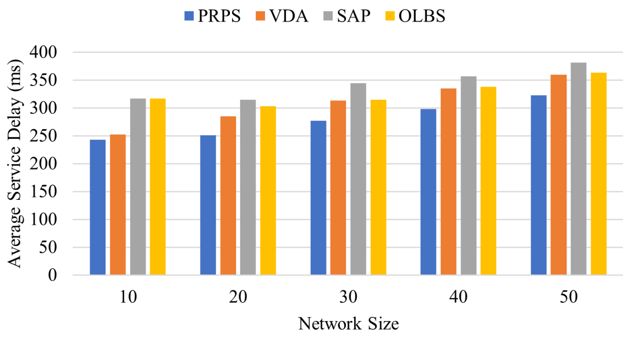


Fig. 7: Average service delay.

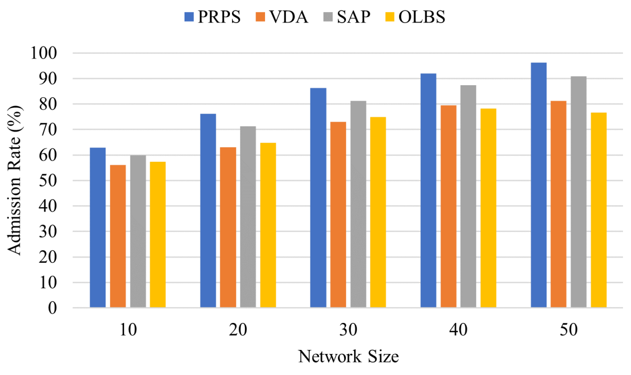


Fig. 8: Admission rate.

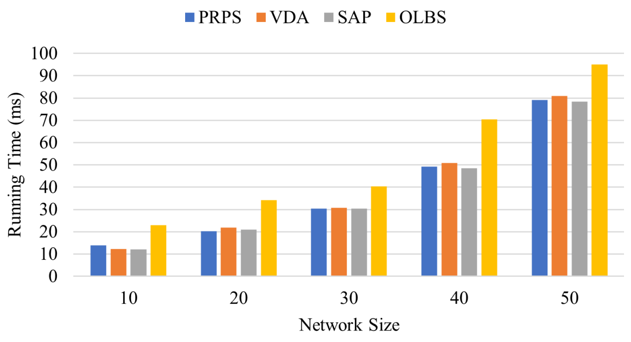


Fig. 9: Running time.

VI. Conclusion

In this paper, we apply NFP and VNF reuse into SFC placement for delay-sensitive IoT applications in MEC network. Based on the two technologies, we articulate the Parallelism-aware Optimization for Delay-sensitive SFC Placement with VNF Reuse (PODPR), which aims to minimize the system energy consumption and meet IoT requirements for low latency. To solve PODPR, we design a heuristic Parallelism and VNF Reuse-based SFC Placement Solution (PRPS) based on path tree structure and greedy-based backtracking pruning algorithm. Simulation results demonstrate that integrating NFP and VNF reuse into SFC placement for delay-sensitive IoT applications can effectively reduce service delay, minimize energy consumption, and offer services to more IoT users. Future research will explore SFC placement in the scenario when parallelizable VNFs can be placed on different servers, and the related issue of energy consumption for IoT applications.

References

[1] X. Wang, H. Xing, F. Song, S. Luo, P. Dai and B. Zhao, ”On Jointly Optimizing Partial Offloading and SFC Mapping: A Cooperative Dual-Agent Deep Reinforcement Learning Approach,” in IEEE Transactions on Parallel and Distributed Systems, vol. 34, no. 8, pp. 2479-2497, Aug. 2023.

[2] X. Gao, R. Liu, A. Kaushik and H. Zhang, ”Dynamic Resource Allocation for Virtual Network Function Placement in Satellite Edge Clouds,” in IEEE Transactions on Network Science and Engineering, vol. 9, no. 4, pp. 2252-2265, 1 July-Aug. 2022.

[3] F. Kamrani, F. Kaheni, S. S. Etehadi, M. Hosseinpour and M. H. Yaghmaee, ”A Double Auction-based MEC-Enabled IoT Task Offloading: A Green Framework,” 2023 7th International Conference on Internet of Things and Applications (IoT), Isfahan, Iran, Islamic Republic of, 2023, pp. 1-5.

[4] T. V. Doan, G. T. Nguyen, M. Reisslein and F. H. P. Fitzek, ”SAP: Subchain-Aware NFV Service Placement in Mobile Edge Cloud,” in IEEE Transactions on Network and Service Management, vol. 20, no. 1, pp. 319-341, March 2023.

[5] S. Zhao, Q. Kang, J. Wang, H. Hu, Y. Fu and others, ”Intelligent Deployment of Delay-Sensitive Service Function Chain Based on Parallelization and Improved Cuckoo Search Algorithm” in Wireless Communications and Mobile Computing, 12 Sept. 2023.

[6] S. Agarwal, V. R. Chintapalli and B. R. Tamma, ”FlexSFC: Flexible Resource Allocation and VNF Parallelism for Improved SFC Placement,” 2022 IEEE 8th International Conference on Network Softwarization (NetSoft), Milan, Italy, pp. 302-306, 2022.

[7] V. R. Chintapalli, B. R. Killi, R. Partani, B. R. Tamma and C. S. R. Murthy, ”Energy- and Reliability-Aware Provisioning of Parallelized Service Function Chains With Delay Guarantees,” in IEEE Transactions on Green Communications and Networking, vol. 8, no. 1, pp. 205-223, March 2024.

[8] N. H. Thanh, N. Trung Kien, N. V. Hoa, T. T. Huong, F. Wamser and T. Hossfeld, ”Energy-Aware Service Function Chain Embedding in Edge–Cloud Environments for IoT Applications,” in IEEE Internet of Things Journal, vol. 8, no. 17, pp. 13465-13486, 1 Sept.1, 2021.

[9] M. Liu, G. Feng, Y. Sun, N. Chen and W. Tan, ”A Network Function Parallelism-Enabled MEC Framework for Supporting Low-Latency Services,” in IEEE Transactions on Services Computing, vol. 16, no. 1, pp. 40-52, 1 Jan.-Feb. 2023.

[10] A. Mohamad and H. S. Hassanein, ”Prediction-based SFC Placement with VNF Sharing at the Edge,” 2022 IEEE 47th Conference on Local Computer Networks (LCN), Edmonton, AB, Canada, 2022, pp. 26-33.

[11] Y. Mao, X. Shang, Y. Liu and Y. Yang, ”Joint Virtual Network Function Placement and Flow Routing in Edge-Cloud Continuum,” in IEEE Transactions on Computers, vol. 73, no. 3, pp. 872-886, March 2024.

[12] H. Liu, S. Long, Z. Li, Y. Fu, Y. Zuo and X. Zhang, ”Revenue Maximizing Online Service Function Chain Deployment in Multi-Tier Computing Network,” in IEEE Transactions on Parallel and Distributed Systems, vol. 34, no. 3, pp. 781-796, 1 March 2023.

[13] S. Yang, F. Li, R. Yahyapour and X. Fu, ”Delay-Sensitive and Availability-Aware Virtual Network Function Scheduling for NFV,” in IEEE Transactions on Services Computing, vol. 15, no. 1, pp. 188-201, 1 Jan.-Feb. 2022.

[14] C. Cabrera, S. Svorobej, A. Palade, A. Kazmi and S. Clarke, ”MAACO: A Dynamic Service Placement Model for Smart Cities,” in IEEE Transactions on Services Computing, vol. 16, no. 1, pp. 424-437, 1 Jan.-Feb. 2023.

[15] A. Abouaomar, S. Cherkaoui, Z. Mlika and A. Kobbane, ”Mean-Field Game and Reinforcement Learning MEC Resource Provisioning for SFC,” 2021 IEEE Global Communications Conference (GLOBECOM), Madrid, Spain, 2021, pp. 1-6.

[16] N. Toumi, O. Bernier, D. -E. Meddour and A. Ksentini, ”On Using Physical Programming for Multi-Domain SFC Placement With Limited Visibility,” in IEEE Transactions on Cloud Computing, vol. 10, no. 4, pp. 2787-2803, 1 Oct.-Dec. 2022.

[17] D. Harutyunyan, N. Shahriar, R. Boutaba and R. Riggio, ”Latency and Mobility–Aware Service Function Chain Placement in 5G Networks”, vol. 21, no. 5, pp. 1697-1709, 2022.

[18] C. Sun, J. Bi, Z. Zheng, H. Yu and H. Hu, ”NFP: Enabling Network Function Parallelism in NFV” in Proceedings of the Conference of the ACM Special Interest Group on Data Communication, 2017.

[19] F. Tian, J. Liang and J. Liu, ”Joint VNF Parallelization and Deployment in Mobile Edge Networks,” in IEEE Transactions on Wireless Communications, vol. 22, no. 11, pp. 8185-8199, Nov. 2023.

[20] D. Zheng, C. Peng, X. Liao and X. Cao, ”Parallelism-aware Service Function Chaining and Embedding for 5G Networks” in 2021 International Conference on Computer Communications and Networks (ICCCN), pp. 1-9, 2021.

[21] F. Tashtarian, M. F. Zhani, B. Fatemipour and D. Yazdani, ”CoDeC: A Cost-Effective and Delay-Aware SFC Deployment,” in IEEE Transactions on Network and Service Management, vol. 17, no. 2, pp. 793-806, June 2020.

[22] J. Luo, J. Li, L. Jiao and J. Cai, ”On the Effective Parallelization and Near-Optimal Deployment of Service Function Chains,” in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 5, pp. 1238-1255, 1 May 2021.

[23] I. -C. Lin, Y. -H. Yeh and K. C. -J. Lin, ”Toward Optimal Partial Parallelization for Service Function Chaining,” in IEEE/ACM Transactions on Networking, vol. 29, no. 5, pp. 2033-2044, Oct. 2021.

[24] P. Jin, X. Fei, Q. Zhang, F. Liu and B. Li, ”Latency-aware VNF Chain Deployment with Efficient Resource Reuse at Network Edge,” IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, Toronto, ON, Canada, 2020, pp. 267-276.

[25] N. Promwongsa, A. Ebrahimzadeh, R. H. Glitho and N. Crespi, ”Joint VNF Placement and Scheduling for Latency-Sensitive Services,” in IEEE Transactions on Network Science and Engineering, vol. 9, no. 4, pp. 2432-2449, 1 July-Aug. 2022.

[26] R. Behravesh, D. Harutyunyan, E. Coronado and R. Riggio, ”Time-Sensitive Mobile User Association and SFC Placement in MEC-Enabled 5G Networks,” in IEEE Transactions on Network and Service Management, vol. 18, no. 3, pp. 3006-3020, Sept. 2021.

[27] H. Ren et al., ”Efficient Algorithms for Delay-Aware NFV-Enabled Multicasting in Mobile Edge Clouds With Resource Sharing,” in IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 9, pp. 2050-2066, 1 Sept. 2020.

[28] Y. Ma, W. Liang, J. Wu and Z. Xu, ”Throughput Maximization of NFV-Enabled Multicasting in Mobile Edge Cloud Networks,” in IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 2, pp. 393-407, 1 Feb. 2020.

[29] A. Varasteh, B. Madiwalar, A. Van Bemten, W. Kellerer and C. Mas-Machuca, ”Holu: Power-Aware and Delay-Constrained VNF Placement and Chaining,” in IEEE Transactions on Network and Service Management, vol. 18, no. 2, pp. 1524-1539, June 2021.