An Efficient Service Function Chaining Placement Algorithm in Mobile Edge Computing

MENG WANG, BO CHENG, and JUNLIANG CHEN, Beijing University of Posts and Telcommunications, P.R. China

Mobile Edge Computing (MEC) is a promising network architecture that pushes network control and mobile computing to the network edge. Recent studies propose to deploy MEC applications in the Network Function Virtualization (NFV) environment. The mobile network service in NFV is deployed as a Service Function Chaining (SFC). In the dynamic and resource-limited mobile network, SFC placement aiming at optimizing resource utilization is a challenging problem. In this article, we solve the SFC placement problem in the MEC-NFV environment. We formulate the SFC placement problem as a weighted graph matching problem, including two sub-problems: a graph matching problem and an SFC mapping problem. To efficiently solve the graph matching problem, we propose a Linear Programming–(LP) based approach to calculate the similarity between VNFs and physical nodes. Based on the similarity, we design a Hungarian-based algorithm to solve the SFC mapping problem. Evaluation results show that our proposed LP-based solutions outperform the heuristic algorithms in terms of execution time and resource utilization.

CCS Concepts: • **Networks** → **Network resources allocation**; • **Human-centered computing** → **Mobile computing**;

Additional Key Words and Phrases: Mobile edge computing, network function virtualization, service function chaining, placement, linear programming

ACM Reference format:
Meng Wang, Bo Cheng, and Junliang Chen. 2020. An Efficient Service Function Chaining Placement Algorithm in Mobile Edge Computing. *ACM Trans. Internet Technol.* 20, 4, Article 32 (October 2020), 21 pages.
https://doi.org/10.1145/3388241

1 INTRODUCTION

With the advent of 5G and IoT [27], mobile applications with extreme requirements are increasing. Mobile Edge Computing (MEC) [18] and Network Function Virtualization (NFV) [11] are two emerging technologies to satisfy the mobile network service requests. Figure 1 shows the MEC-NFV environment. In this figure, the mobile edge network is considered as a three-layer architecture consisting of a core level, edge level, and user level. The main idea of MEC is to push computing, network, and storage functions to the network edges (e.g., base stations, access points,

This work was supported by the National Science and Technology Major Project (2018ZX030110004) and in part by the BUPT Excellent Ph.D. Students Foundation (No. CX2019214).
Authors' address: M. Wang, B. Cheng (corresponding author), and J. Chen, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telcommunications, No. 10, Xitucheng Road, Haidian District, Beijing City, P.R. China, 100876; emails: {mengwang, chengbo, chjl}@bupt.edu.cn.

1533-5399/2020/10-ART32 $15.00
https://doi.org/10.1145/3388241

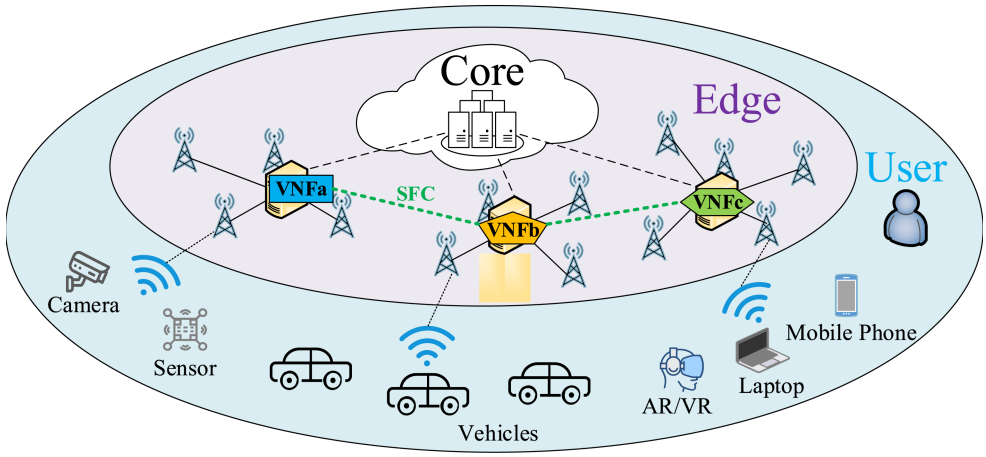


Fig. 1. SFC Placement in the MEC-NFV Environment.

and edge servers) to enable a good performance of latency-intensive and computation-critical network applications at the resource-limited edge devices [23, 24].

NFV is a new network architecture that decouples network functions from the hardware. Due to the convenient and flexible management of Virtual Network Functions (VNFs), NFV significantly reduces the Capital Expenditure (CAPEX) and Operating Expense (OPEX) [26] and plays an important role in communication networks, such as enterprise networks, data center networks, and mobile edge networks. ETSI defines the network service request graphs as VNF Forwarding Graphs (VNF-FGs) [1]. Based on the network service requests, NFV chaining is described as a Service Function Chaining (SFC) consisting of an ordered set of VNFs.

Figure 1 shows the SFC placement in the MEC-NFV environment. At the edge level, an SFC consisting of three VNFs (connected with green dotted lines) runs on the edge servers, as shown in Figure 1. Different from the SFC placement problem in traditional networks, mobile network services run on the resource-limited edge devices. Besides, the network status in the mobile edge network is dynamic and complex. The SFC placement problem in the MEC-NFV environment, which aims at optimizing resource utilization, becomes a critical research issue. Recently, the SFC placement problem in the MEC-NFV environment has received increasing attention [10, 13, 19, 21, 22, 29, 37]. To efficiently solve the SFC placement problem in the MEC-NFV environment, most of the existing works propose heuristic solutions. However, there are still some problems remaining to be solved:

- The mobile edge applications run on resource-limited edge devices. Traditional solutions cannot effectively improve resource utilization.
- Different from traditional networks, mobile edge networks are dynamic and complex. Traditional solutions are unable to effectively sense network state changes during the SFC placement.
- The heuristic algorithms usually iterate multiple times and therefore require long execution time. Besides, heuristic solutions can only find a suboptimal result.

Given these facts, we propose Linear Programming–(LP) based solutions, which aim to optimize resource utilization. We update the network status during the placement process. Besides, LP-based solutions run faster than heuristic solutions. Therefore, LP-based solutions can solve the above problems. In this article, we formulate SFC request and physical network as two weighted graphs.

Therefore, the SFC placement problem can be reformulated as an optimal matching problem that aims at minimizing the distance between the adjacency matrices of two weighted graphs. Then we extend the Weighted Graph Matching Problem (WGMP) [3] to solve the optimal matching problem. The problem is divided into two sub-problems: a graph matching problem and an SFC mapping problem. To efficiently solve the two sub-problems, we propose an LP-based matching algorithm and a Hungarian-based mapping algorithm.

In summary, the main contributions of this article are summarized as follows:

- We formulate SFC request and physical network as two weighted graphs and formulate the SFC placement problem in the MEC-NFV environment as the WGMP consisting of graph matching and SFC mapping.
- We propose an LP-based algorithm and a Hungarian-based algorithm to solve graph matching and SFC mapping in the WGMP. Our proposed solutions can run in polynomial time and optimize resource utilization.
- We evaluate the performance of our proposed solutions. Evaluation results show that our proposed LP-based solutions outperform heuristic algorithms in terms of execution time and resource utilization.

The rest of this article is organized as follows. Section 2 discusses the related works. Section 3 describes the SFC placement problem in the MEC-NFV environment. Section 4 formulates the SFC placement problem aiming at optimizing resource utilization. Section 5 details the proposed solutions. Section 6 evaluates the performance. Finally, Section 7 concludes this article.

2 RELATED WORKS

In general, the SFC placement problem focuses on how to deploy VNF chaining in the physical networks. Traditionally, the SFC is deployed in communication networks, such as enterprise networks and data center networks. In this article, we deploy SFC in the mobile edge network, which is dynamic and resource limited. Therefore, we first discuss the SFC placement in traditional networks, since traditional placement solutions can bring many benefits. Then, we describe the SFC placement in the MEC-NFV environment.

2.1 Traditional SFC Placement

Recently, multiple works deploy SFC in the traditional physical network. Herrera et al. [16] define the NFV resource allocation problem as three phases, including SFC composition, VNF-FG embedding, and VNF scheduling. Beck et al. [7] propose a heuristic algorithm in a random physical network to map SFC in a scalable and coordinated way. Ye et al. [39] propose placement solutions to optimize the network configuration and guarantee the availability of SFC. Sallam et al. [28] propose a joint SFC placement and resource allocation algorithm that can maximize the network flow and meet resource constraints. Ye et al. [38] propose an end-to-end packet delay model to embed the VNF chaining into 5G core networks. Benkacem et al. [8] formulate the SFC placement problem as two linear integer problems aiming at maximizing the quality of experience and minimizing the cost. Zheng et al. [41] formulate the SFC placement problem as a non-linear integer programming model and propose an efficient heuristic algorithm to quickly find near-optimal placement solutions. Bari et al. [6] formulate the VNF orchestration problem as a Multi-Stage directed graph with associated costs. Then they can find a near-optimal VNF placement result form the directed graph.

In summary, most SFC placement works propose heuristic algorithms and solve the problem in NFV-enabled networks. In this article, we focus on the SFC placement problem in the MEC-NFV environment. In the mobile edge network, the resource is limited and network status is dynamic.

Therefore, we propose LP-based solutions to reduce execution time and optimize resource utilization in MEC.

2.2 SFC Placement in MEC

Different from the traditional network, the mobile edge network has multiple new requirements such as data, resource, and IoT devices [9, 14, 15, 35, 40]. With the rapid increase of mobile edge computing, resource allocation related problems become a hot topic. In References [17, 32–34, 36], the authors propose a set of algorithms to solve the placement and optimization problem in MEC. For the SFC placement problem, it has been widely studied in MEC. Jemaa et al. [19] introduce VNF placement and optimization strategies to optimize resource utilization and prevent cloudlet overload. Laghrissi et al. [21] propose an enhanced predictive placement algorithm for edge slicing in the edge cloud environment. Yala et al. [37] propose a formulation of the VNF placement problem as an optimization problem of two objectives, namely maximizing service availability and minimizing access latency. And they propose a genetic algorithm to solve this problem. Li et al. [22] design a particle swarm optimization–based edge server placement algorithm to reduce the energy consumption in mobile edge computing. Cziva et al. [13] formulate the edge VNF placement problem and propose a dynamic placement scheduler to minimize VNF migrations. Song et al. [29] propose a VNF resource allocation scheme based on context-aware grouping technology to minimize the delay of edge network services. Chen et al. [10] propose a greedy-based Minimal Neighbourhood (MINI) algorithm. The MINI is aimed at finding a minimal node to optimize node utilization.

In summary, most placement solutions in MEC also use heuristic algorithms to solve the SFC placement problem. Our work is different from the mentioned works, since we divide the placement problem into two sub-problems: an LP-based graph matching problem and a Hungarian-based SFC mapping problem. Our proposed solutions can run in polynomial time and optimize resource utilization.

3 PROBLEM STATEMENT

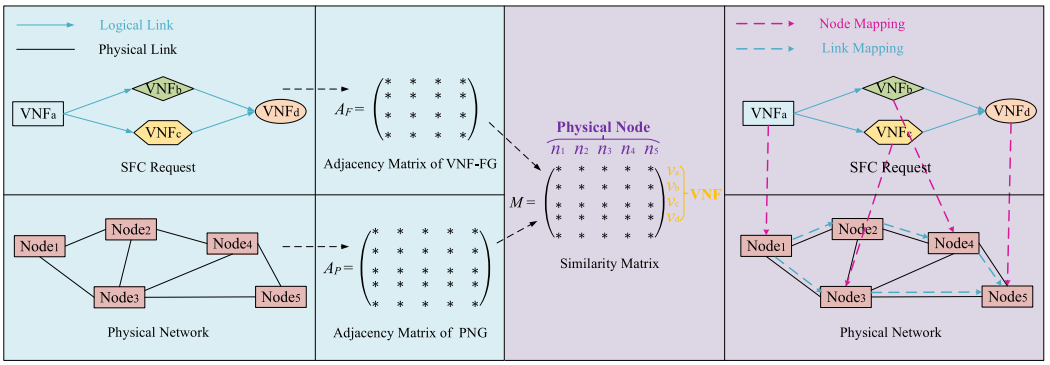
In this article, we formulate the SFC placement problem as a Weighted Graph Matching Problem (WGMP) [3]. Therefore, we divide the SFC placement problem into two sub-problems: a graph matching problem and an SFC mapping problem. In this section, we present a detailed description of the two sub-problems.

3.1 Graph Matching Problem

Figure 2(a) shows the graph matching problem, which is responsible for matching SFC request and physical network. We formulate SFC request and physical network as VNF-FG and PNG (Physical Network Graph). Figure 2(a) shows the adjacency matrices of VNF-FG and PNG. The graph matching problem can be formulated as a minimum problem aiming at minimizing the distance between VNF-FG and PNG. Therefore, we propose an LP-based algorithm (Section 5.2) to solve the minimum problem. Finally, we can get the similarity between VNFs and physical nodes.

3.2 SFC Mapping Problem

Figure 2(b) shows the SFC mapping problem aiming at mapping SFC in the physical network. The similarity matrix that describes the similarity between VNFs and physical nodes, as shown in Figure 2(b). In this article, we formulate the mapping problem as an assignment problem. Therefore, we propose a Hungarian-based algorithm (Section 5.3) to solve the assignment problem. In this algorithm, we update the similarity matrix based on the mapping result, thus optimizing resource utilization. Therefore, we can get the mapping result based on the similarity matrix.



(a) Graph Matching Problem (b) SFC Mapping Problem

Fig. 2. Two Sub-problems: (a) Graph Matching Problem and (b) SFC Mapping Problem.

4 PROBLEM FORMULATION

In this section, we formulate SFC request and physical network as two weighted graphs: VNF-FG and PNG. Then we reformulate the SFC placement problem as an optimal graph matching problem. A summary of used basic notations is found in Table 1.

4.1 VNF-FG

The SFC request can be described as a VNF-FG, that is, $F = (V, E)$. V and E indicate the set of VNFs and logical links, respectively. We use cpu_{v_p} and mem_{v_p} to indicate the CPU and memory consumption of VNF $v_p \in V$. The bandwidth consumption of each logical link is the bandwidth of VNF flows passing through it. Therefore, we use $bw_{e_{pq}}$ to indicate the requested bandwidth of logical link $e_{pq} \in E$.

4.2 PNG

We formulate the physical network as a PNG, that is, $P = (N, L)$. N and L indicate the set of physical nodes and physical links, respectively. We use $C^{cpu}_{n_i}$ and $C^{mem}_{n_i}$ to indicate the CPU and memory capacity of each physical node $n_i \in N$. And we use $C^{bw}_{l_{ij}}$ to indicate the bandwidth capacity of each physical link $l_{ij} \in L$.

4.3 Objectives

4.3.1 Resource Constraints. In this subsection, we use two binary variables $x^{v_p}_{n_i}$ and $y^{e_{pq}}_{l_{ij}}$ to indicate the mapping status of VNF and logical link:

$$x^{v_p}_{n_i} = \begin{cases} 1 & \text{if } v_p \text{ is mapped in } n_i \\ 0 & \text{otherwise} \end{cases}, \tag{1}$$

$$y^{e_{pq}}_{l_{ij}} = \begin{cases} 1 & \text{if } e_{pq} \text{ is mapped in } l_{ij} \\ 0 & \text{otherwise} \end{cases}. \tag{2}$$

We take CPU, memory, and bandwidth constraints into consideration. In detail, the CPU and memory consumption of all VNFs mapped in the same physical node cannot exceed the resource capacity of this physical node. Therefore, the CPU and memory constraints are as follows:

$$\sum_{v_p} cpu_{v_p} \cdot x^{v_p}_{n_i} \le C^{cpu}_{n_i} \qquad \forall n_i \in N, \tag{3}$$

Table 1. Basic Notations Used throughout This Paper

Symbol	Definition
VNF-FG	
$V = \{v_1, v_2 \cdots\}$	the set of VNFs.
$E = \{e_{pq} \cdots\}$	the set of logical links.
v_p, v_q	two VNFs in SFC.
$e_{pq} = (v_p, v_q)$	the logical link between v_p and v_q.
PNG	
$N = \{n_1, n_2 \cdots\}$	the set of physical nodes.
$L = \{l_{ij} \cdots\}$	the set of physical links.
n_i, n_j	two physical nodes in physical network.
$l_{ij} = (n_i, n_j)$	the physical link between n_i and n_j.
Resource	
cpu_{v_p}	the CPU consumption of v_p.
mem_{v_p}	the memory consumption of v_p.
$bw_{e_{pq}}$	the bandwidth consumption of logical link l_{pq} (between v_p and v_q).
$C^{cpu}_{n_i}$	the CPU capacity of n_i.
$C^{mem}_{n_i}$	the memory capacity of n_i.
$C^{bw}_{l_{ij}}$	the bandwidth capacity of physical link l_{ij} (between n_i and n_j).
$U^{cpu}_{n_i}$	the CPU utilization of n_i.
$U^{mem}_{n_i}$	the memory utilization of n_i.
$U^{bw}_{l_{ij}}$	the bandwidth utilization of physical link l_{ij} (between n_i and n_j).
Variables	
$x^{v_p}_{n_i}$	whether v_p is mapped in n_i.
$y^{e_{pq}}_{l_{ij}}$	whether e_{pq} is mapped in l_{ij}.

$$\sum_{v_p} mem_{v_p} \cdot x^{v_p}_{n_i} \leq C^{mem}_{n_i} \qquad \forall n_i \in N. \tag{4}$$

Similarly, the bandwidth consumption between two VNFs cannot exceed the bandwidth capacity between two physical nodes where two VNFs placed in. Therefore, the bandwidth constraints are

$$\sum_{e_{pq}} bw_{e_{pq}} \cdot y^{e_{pq}}_{l_{ij}} \leq C^{bw}_{l^r_{ij}} \qquad \forall l_{ij} \in L. \tag{5}$$

4.3.2 Resource Utilization. In this article, we solve the SFC placement aiming at optimizing resource utilization. Similarly, we focus on CPU, memory, and bandwidth. Therefore, resource utilization can be formulated as

$$U^{cpu}_{n_i} = \frac{\sum_{v_p} cpu_{v_p} \cdot x^{v_p}_{n_i}}{C^{cpu}_{n_i}} \qquad \forall n_i \in N, \tag{6}$$

$$U^{mem}_{n_i} = \frac{\sum_{v_p} mem_{v_p} \cdot x^{v_p}_{n_i}}{C^{mem}_{n_i}} \qquad \forall n_i \in N, \tag{7}$$

$$U^{bw}_{l_{ij}} = \frac{\sum_{e_{pq}} bw_{e_{pq}} \cdot y^{e_{pq}}_{l_{ij}}}{C^{bw}_{l^r_{ij}}} \qquad \forall l_{ij} \in L. \tag{8}$$

In this article, our goal is to improve resource utilization:

$$\max \left(\frac{\sum_{n_i} U_{n_i}^{cpu}}{\sum_{n_i} x_{n_i}^{vp}} + \frac{\sum_{n_i} U_{n_i}^{mem}}{\sum_{n_i} x_{n_i}^{vp}} + \frac{\sum_{l_{ij}} U_{l_{ij}}^{bw}}{\sum_{l_{ij}} y_{l_{ij}}^{epq}} \right) \quad s.t. \quad \text{Equation (3)} \quad \text{to} \quad \text{Equation (8)}. \tag{9}$$

In this article, we formulate the SFC placement problem as the WGMP consisting of a graph matching problem and an SFC mapping problem. In the graph matching problem, matching similar VNF and the physical node can improve resource utilization. In the SFC mapping problem, updating the similarity matrix based on the mapping result can also optimize resource utilization.

5 PROPOSED SOLUTIONS

As we mentioned, we divide the SFC placement problem into two sub-problems: a graph matching problem and an SFC mapping problem. In this section, we propose an LP-based algorithm and a Hungarian-based algorithm to solve the two sub-problems. While LP in the Weighted Graph Matching Problem (WGMP) [3] is an old idea, it has not been widely applied to the SFC placement problem in the MEC-NFV environment.

5.1 LP-based Approach for WGMP

The main idea of WGMP is to minimize the distance between two weighted graphs. Therefore, we can optimize resource utilization by using WGMP in the MEC-NFV environment. However, there are also some limitations in the LP-based solutions for WGMP.

(1) WGMP requires two weighted graphs of the same size. However, PNG size is often different from VNF-FG. In the MEC-NFV environment, PNG size is significantly larger than VNF-FG.
(2) High similarity means that the weight of VNF and physical node is similar but may not meet the resource constraints. For example, the weight of physical node is slightly less than VNF.
(3) The network in the MEC-NFV environment is dynamic and complex. The network status changes after the VNF is placed. This makes the original similarity matrix not suitable for the current network.

In summary, there are some problems in the LP-based solutions for WGMP. Therefore, we propose an LP-based matching algorithm and a Hungarian-based mapping algorithm to solve these problems. Here are the corresponding solutions we designed in the two algorithms:

(1) We extend the size of VNF-FG adjacency matrix to be the same as PNG. The newly added elements in the adjacency matrix of VNF-FG are all zero. Therefore, the adjacency matrices of VNF-FG and PNG are the same size.
(2) We design a function in the Hungarian-based algorithm. This function is responsible for checking whether the physical nodes and links can hold VNFs and logical links.
(3) After mapping one VNF in one physical node, we update the corresponding column of the similarity matrix to fit the current network.

5.2 LP-based Graph Matching

In this subsection, we propose an LP-based algorithm to solve the graph matching problem. In Algorithm 1, we use the SFC request and the physical network as input. The output is the similarity matrix.

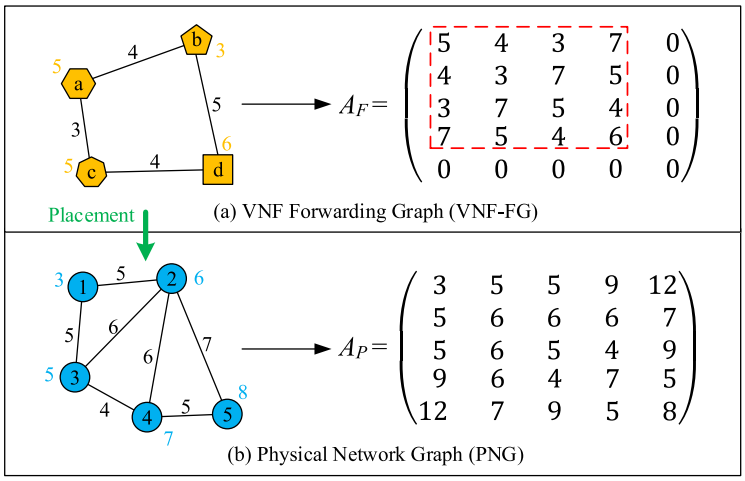


Fig. 3. The Adjacency Matrices of (a) VNF-FG and (b) PNG.

ALGORITHM 1: LP-based Graph Matching

Input: The SFC request: $F = (V, E)$; The physical network: $P = (N, L)$;
Output: The similarity matrix: M;

1 **Step1:** Compute the adjacency matrices A_F and A_P:

$$A_P = \begin{cases} p_{ii} = C^{cpu}_{n_i} \\ p_{ij} = C^{bw}_{l_{ij}} \end{cases} \quad A_F = \begin{cases} f_{pp} = cpu_{v_p} \\ f_{pq} = bw_{e_{pq}} \end{cases}$$

2 **Step2:** Compute matrix A_{PF} from A_F and A_P:

$$A_{PF} = \begin{bmatrix} \{A_P - f_{11}I_n\} & \{-f_{21}I_n\} & \cdots & \{-f_{n1}I_n\} \\ \{-f_{12}I_n\} & \{A_P - f_{22}I_n\} & \cdots & \{-f_{n2}I_n\} \\ \vdots & \vdots & \ddots & \vdots \\ \{-f_{1n}I_n\} & \{-f_{2n}I_n\} & \cdots & \{A_P - h_{nn}I_n\} \end{bmatrix}$$

where I_n is an identity matrix.

3 **Step3:** Use the Simplex method to solve the LP problem in Equation (9). The matrix B is define by:$b_{ij} = \begin{cases} 1 & \text{for } i = 1, 2, 3, \ldots, n \text{ and } j = i, i+n, i+2n, \ldots, i+(n-1)n \\ 0 & \text{otherwise} \end{cases}$

4 **Step4:** Get the similarity matrix M from vector m;

5.2.1 Adjacency Matrix (Step 1). At Step 1, A_F and A_P indicate the adjacency matrices of VNF-FG and PNG, respectively. We use p_{ii} to indicate the CPU capacity of physical node n_i, so is f_{ii}. Note that we set a certain ratio between CPU and memory so that the association between the adjacency matrix and memory can be established. And we use p_{ij} to indicate the distance between physical nodes n_i and n_j (so is f_{ij}). We use the Dijkstra method to compute the distance between physical nodes (or VNFs) that are not directly connected.

As Figure 3(a) shows, the adjacency matrix $(4 * 4)$ of VNF-FG is in the red dotted line box. As Figure 3(b) shows, the adjacency matrix of PNG is a $5 * 5$ matrix. Therefore, we extend the adjacency matrix of VNF-FG to a $5 * 5$ matrix, that is, A_F (the newly added elements are all zeros).

5.2.2 Distance (Step 2). Based on Reference [3], the distance between VNF-FG and PNG can be defined as:

$$J(\Phi) = \sum_{i=1}^{n}\sum_{j=1}^{n}(p(n_i, n_j) - f(\Phi(n_i), \Phi(n_j)))^2, \tag{10}$$

where $p(n_i, n_j)$ indicates the weight between physical node n_i and n_j. Φ indicates the one-to-one correspondence between VNF and physical node. For example, we assume that $(\Phi(n_i), \Phi(n_j)) = (v_p, v_q)$. Therefore, $f(\Phi(n_i), \Phi(n_j))$ is the weight between VNF v_p and v_q.

The WGMP is aimed at minimizing the distance $J(\Phi)$. Therefore, the matching problem can be reformulated as:

$$\min_{M} \|A_P - MA_FM^{\mathrm{T}}\|_1, \tag{11}$$

where A_F and A_P are the adjacency matrices of VNF-FG and PNG, respectively. The permutation matrix M indicates the mapping function Φ. Note that $||.||$ is the L_1 norm.

The WGMP in Equation (11) is equivalent to Equation (12):

$$\min_{M} \|A_PM^{\mathrm{T}} - M^{\mathrm{T}}A_F\|_1. \tag{12}$$

Then, we define a $n * n$ matrix R:

$$R = A_PM^{\mathrm{T}} - M^{\mathrm{T}}A_F. \tag{13}$$

The matrices $R = \{r_{ij}\}$ and $M = \{m_{ij}\}$ can be partitioned by columns:

$$\begin{aligned} VEC(R) &= \{r_{11}, r_{21}, \ldots, r_{1n}, r_{2n}, \ldots, r_{nn}\}^{\mathrm{T}} \\ VEC(M^{\mathrm{T}}) &= \{m_{11}, m_{21}, \ldots, m_{1n}, m_{2n}, \ldots, m_{nn}\}^{\mathrm{T}}. \end{aligned} \tag{14}$$

Therefore, the WGMP in Equation (11) can be reformulated as:

$$\min_{M} \|VEC(R)\|_1 = \min_{M} \left\|A_{PF}VEC(M^{\mathrm{T}})\right\|_1, \tag{15}$$

where A_{PF} is a $n^2 * n^2$ matrix derived from $A_P = \{p_{ij}\}$ and $A_F = \{f_{ij}\}$, which is described in Algorithm 1 (Step 2).

From Equation (11) to Equation (15), we can conclude that the matching problem in Equation (10) is equivalent to Equation (16):

$$\min_{m} \|A_{PF}m\|_1 \qquad m \geq 0, \tag{16}$$

where $m = VEC(M^{\mathrm{T}})$ is a $n^2 * 1$ vector.

5.2.3 LP Problem (Step 3). Finally, we reformulate the minimization problem in Equation (16) as an LP problem:

$$\begin{aligned} &\min_{m,S,T} \sum_{i}^{n^2} S_i + T_i \\ s.t. \quad &A_{PF}m + S - T = 0 \\ &Bm = e \\ &m \geq 0, S \geq 0, T \geq 0, \end{aligned} \tag{17}$$

where $\{S_i\}$ and $\{T_i\}$ indicate two sets of real positive decision variables, e is an identity matrix, and B is a $2n * n^2$ matrix defined in Algorithm 1 (Step 3). B indicates the constraints that the permutation matrix M needs to keep the sum of any rows or columns to be 1.

Therefore, we use the Simplex method (Step 3 in Algorithm 1) to solve the LP problem in Equation (17). Finally, we can get the solution of the minimization problem (vector m).

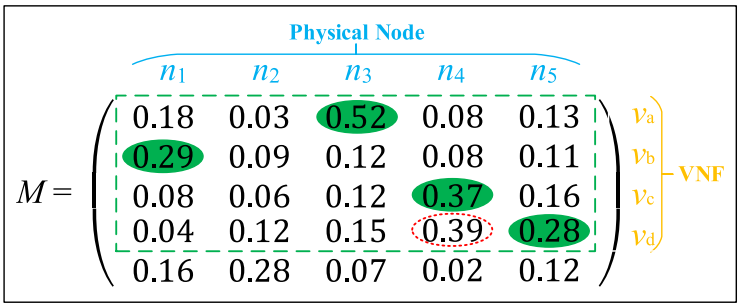


Fig. 4. Similarity Matrix M.

5.2.4 Similarity Matrix (Step 4). In this step, we can get the similarity matrix M from vector m. As Figure 4 shows, the data in the green dotted line box indicate the similarity between VNFs and physical nodes. The rows and columns of the similarity matrix M correspond to VNFs and physical nodes, respectively. For example, m_{12} indicates the similarity between VNF v_a and physical node n_2.

Based on the highest similarity (the green data) in Figure 4, the optimal match should be that v_a, v_b, and v_c are mapped to n_3, n_1, and n_4, respectively. And v_d should be also mapped to n_4 (the red data). However, n_4 cannot satisfy the resource consumption of v_c and v_d ($C_{n_4}^{cpu} = 7$ while $cpu_{v_c} = 5$ and $cpu_{v_d} = 6$). Therefore, v_d should be mapped to n_5, since the similarity between v_d and n_5 is the next highest. Besides, we also need to consider other constraints such as memory and bandwidth.

Based on these facts, we design a Hungarian-based SFC mapping algorithm that computes the mapping results to meet the resource constraints.

5.3 Hungarian-based SFC Mapping

Based on the similarity matrix, the SFC mapping problem can be formulated as an assignment problem. To solve this problem, we propose a Hungarian-based SFC mapping algorithm. In this algorithm, we use the SFC request, the physical network, and the similarity matrix as input. The output of this algorithm is the mapping result matrix.

At line 1 of Algorithm 2, we define *find_node* to indicate that whether we find a physical node that can hold VNF. Then we traverse all VNFs to find physical nodes to hold them (lines 2–21). At line 3, we get the row in matrix M corresponding to this VNF. At line 4, we set *find_node* to false. Then, the algorithm starts to find one physical node until *find_node* is true (line 5). At line 6, we get the index of the highest element (the highest similarity). If the largest value is at two or more places, then we compare resource utilization of the corresponding physical nodes. And we choose the place with higher resource utilization as the highest element. After getting the highest element, the algorithm calls *CheckNodeAndLink* function to determine whether the corresponding physical node can hold the VNF (line 7). Besides, the function checks whether the physical links between the previous physical node and this physical node meet the bandwidth consumption. If all of the resource meets the constraints, then the network state is updated (lines 8–11). It is worth mentioning that we update the similarity matrix (line 9). Since the available resource of the physical node changes, we update the similarity of this physical node and VNFs. Sometimes, the resource capacity of physical nodes and links may fail to meet the resource constraints. In this case, we set this highest value to −1 (line 14). Therefore, the VNF can find a physical node with the next highest similarity value (line 6). After mapping all VNFs, the algorithm finally returns the mapping result matrix (line 18).

ALGORITHM 2: Hungarian-based SFC Mapping

Input: The SFC request: $F = (V, E)$; The physical network: $P = (N, L)$; The similarity matrix: M;
Output: The mapping result matrix: M_{res};

```
1  Initialize: find_node = false;
2  foreach v_p in V do
3      row_p = pth row in similarity matrix M;
4      find_node = false;
5      while !find_node do
6          i = the index of highest element value in row_p;
7          if CheckNodeAndLink(v_p, n_i) then
8              Update the physical network status;
9              Update the ith column in similarity matrix M;
10             m_pi = 1;
11             find_node = true;
12         end
13         else
14             row_p[i] = −1;
15         end
16     end
17 end
18 return M_res;
```

The *CheckNodeAndLink* function is responsible for checking whether the physical node and link can hold the VNF and logical link. The input is VNF and the physical node. The output of this function is true or false.

At line 1 in Function 1, we check the CPU and memory constraints. If it succeeds, then we check the bandwidth constraints (line 2). However, the candidate physical node and the previous one may not be directly connected. Therefore, we need to compute the physical link path between two physical nodes and check all physical links (line 2). If the physical node and links can hold the VNF and logical link, then the function updates the physical network status and returns true (lines 3 and 4). Otherwise, returns false (lines 7 and 11).

FUNCTION 1: CheckNodeAndLink

Input: The VNF: v_p; The physical node: n_i;
Output: Boolean: *true* or *false*;

```
1  if C_{n_i}^{cpu} ≥ cpu_{v_p} and C_{n_i}^{mem} ≥ mem_{v_p} then
2      if C_l^{bw} ≥ bw_e, ∀ l, e ∈ (n_{i−1}, n_i) then
3          Update the physical network status;
4          return true;
5      end
6      else
7          return false;
8      end
9  end
10 else
11     return false;
12 end
```

5.4 Complexity Analysis

In this subsection, we discuss the time complexity of our proposed solutions.

In Algorithm 1, the complexity of Step1, Step 2, and Step 4 is $O(|N|^2)$, $O(|N|^4)$, and $O(|N|^2)$, respectively. Note that the Step 3 is implemented using a Simplex method but it has shown acceptable performance in most practical applications. Therefore, the time complexity of Algorithm 1 (except Step 3) is at the level of $O(|N|^4)$.

In Algorithm 2, the *foreach* (line 2) and *while* (line 5) run $|V|$ and $|N|$ times, respectively. At line 7, Algorithm 2 calls Function 1. However, since the limited number of physical links, the consumption of Function 1 (traversing the physical links) can be ignored. At line 8, the Algorithm 2 runs $|V|$ times. Therefore, the time complexity of Algorithm 2 is at the level of $O(|V|^2\,|N|)$.

In summary, the complexity of our proposed solutions is equal to the complexity of the combination of Algorithm 1 and Algorithm 2. Therefore, our proposed LP-based solutions are at the level of $O((|V|^2 + |N|^3)|N|)$. Compared with the most heuristic algorithms, our proposed algorithm has shown better performance (elaborate in Section 6.5), although it is at the level of $O(|N|^4)$.

5.5 Baseline

In this subsection, we design a bipartite matching-based Greedy algorithm as a baseline to compare the performance of our proposed solutions.

The Greedy algorithm also formulates SFC request and physical network as two weighted graphs: VNF-FG and PNG. Different from the LP-based solutions, the Greedy algorithm matches VNF-FG and PNG based on bipartite matching and maps SFC in a greedy way. Our proposed LP-based solutions match VNF-FG and PNG based on linear programming that can run in polynomial time. Besides, the LP-based solutions map SFC based on the Hungarian algorithm, which can update the network status during the placement process.

In conclusion, the Greedy algorithm matches VNF-FG and PNG based on bipartite matching and maps SFC in a greedy way. In this algorithm, we use the SFC request and the physical network as input. And the output is the mapping result matrix.

ALGORITHM 3: Greedy Algorithm

Input: The SFC request: $F = (V, E)$ The physical network: $P = (N, L)$;;
Output: The mapping result matrix: M_{res};
1 **Step1:** Compute the adjacency matrices A_F and A_P;
2 **Step2:** Initialize the potential value and compute the bipartite graphs using A_F and A_P;
3 **Step3:** Compute max matching with min cost;
4 **Step4:** Check the physical node mapping:

$$\begin{cases} \textit{success,} & \textit{go to Step 5} \\ \textit{fail,} & \textit{update the potential value and go to Step 3} \end{cases}$$

5 **Step5:** Compute the physical link path and check the physical link mapping:

$$\begin{cases} \textit{success,} & m_{ij} = 1,\ \textit{update the physical network status} \\ \textit{fail,} & \textit{update the potential value and go to Step 3} \end{cases}$$

6 **Step6:** Return the mapping result matrix M_{res}.

At Step 1 of Algorithm 3, we compute the adjacency matrices for VNF-FG and PNG. Similarly, adjacency matrices are the same as in Figure 3. At Step 2, we initialize a potential value and create the bipartite graphs using the two adjacency matrices. At Step 3, we compute the max matching with min cost. At Step 4, we check whether the candidate physical node can hold the candidate VNF. If it succeeds, then go to Step 5. Otherwise, the algorithm updates the potential value and returns Step 3. At Step 5, we first compute the physical link path between the candidate physical

node and the previous one. Then we check whether all physical links can hold the logical link, just like the link check in Function 1 (Section 5.3). If it succeeds, then the algorithm updates the network status and the mapping result matrix. Otherwise, the algorithm updates the potential value and returns Step 3. Therefore, the algorithm starts another iteration. Finally, after multiple iterations, we can get the mapping result matrix (Step 6).

6 PERFORMANCE EVALUATION

In this article, we evaluate the performance of our proposed LP-based solutions in three parts:

- **Resource Utilization** is usually specified as a percentage to indicate the proportion of resource consumed compared to the peak resource. We evaluate the resource utilization of the four placement algorithms with different physical network topologies (Section 6.3).
- **Acceptance Ratio** indicates the ratio of the number of accepted SFC requests to the total number of SFC requests. As the number of SFC requests increases, we observe changes in the acceptance ratio. We evaluate the acceptance ratio of the three placement algorithms. (Section 6.4).
- **Execution Time** We evaluate the execution time of the three placement algorithms with different VNF-FG size and PNG size (Section 6.5).

6.1 Comparison Algorithms

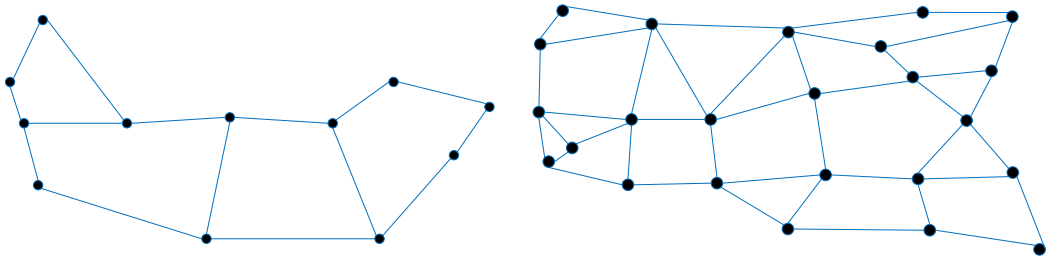
In this subsection, we evaluate the performance of our proposed LP-based solutions and compare it with some related works. The MINI algorithm [10] optimizes resource utilization and acceptance ratio in the mobile computing environment. Therefore, we compare our proposed LP-based solutions with MINI. Besides, most of the related works require a long execution time and only find a suboptimal result. Among them, we compare the execution time of our proposed LP-based solutions with Multi-Stage [6], which can be run in polynomial time. Besides, in Section 5.5, we design the bipartite matching-based Greedy algorithm as a baseline.

- **MINI** [10] (mentioned in Section 2.2) is aimed at finding a minimal neighborhood node in a greedy way, thus optimizing node utilization.
- **Multi-Stage** [6] (mentioned in Section 2.1) is a dynamic programming-based algorithm aiming at finding a near-optimal solution that minimizes the physical resource fragmentation.
- **LP-based solutions** divide the SFC placement problem in the MEC-NFV environment as two sub-problems: a graph matching problem and an SFC mapping problem. The proposed LP-based solutions consist of an LP-based algorithm and a Hungarian-based algorithm.
- **Greedy** (mentioned in Section 5.5) matches VNF-FG and PNG based on bipartite matching and maps SFC in a greedy way.

6.2 Experiment Setup

In this experiment, we evaluate the performance using a laptop of Windows 10 with a 2.2-GHz Intel Core i5 processor and 8 GB memory. We implement the four placement algorithms (LP, Multi-Stage, MINI, and Greedy) in Java based on Alevin [16], a widely used evaluation environment for NFV resource allocation.

6.2.1 Topology. In this article, we use a wide range of network topologies: Internet2 research network [6, 12], US mesh network [4, 20], and Shanghai Telecom base station mobile network [17, 32–34, 36].



(a) Internet2 Research Network (b) US Mesh Network

Fig. 5. (a) Internet2 Research Network and (b) US Mesh Network.

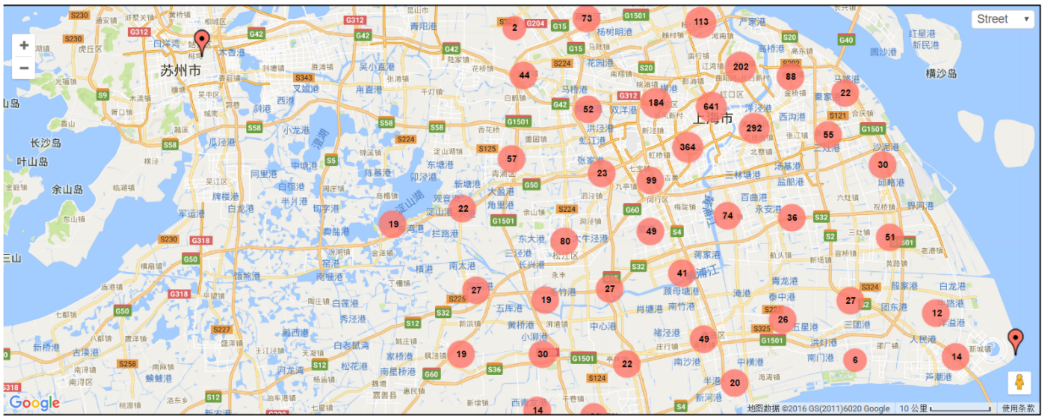


Fig. 6. Distribution of base stations of Shanghai Telecom.

Since the Multi-Stage algorithm is designed for traditional networks, we use the Internet2 research network (12 nodes, 15 links) and US mesh network (24 nodes, 42 links), as shown in Figure 5. We can observe that in traditional networks, the distribution of physical nodes is relatively scattered. And the node connection rate of the US mesh network is higher than Internet2 research network.

For network traffic, we use traffic matrix tracking to generate time-varying traffic [6] for the Internet2 research network and US mesh network.

In this article, we solve the SFC placement problem in the MEC-NFV environment. Therefore, we use the dataset of Shanghai Telecom base station, which contains the contact information of mobile users accessing 3,233 base stations. More specifically, the dataset contains more than 7.2 million records of accessing the network through the base stations for six months. Figure 6 clearly shows the base station distribution contained in the dataset of Shanghai Telecom base station. The number within the range of the red circles indicates the number of base stations. Figure 6 shows that in Shanghai, the base stations are very densely distributed.

In this experiment, we simulate the distribution of edge servers based on the distribution of base stations. Besides, the number of base stations indicates the resource capacity of edge servers. Therefore, we can get the mobile edge server topology (40 nodes, 70 links). In this article, we use this topology as the MEC network topology.

Table 2. Workload of Some Base Stations Included in Shanghai Telecom Base Station Dataset

Base Station ID	40	76	328	531	664	1039	1265	1927	2160	2513	2748
User Number	1824	6	108	81	151	254	499	821	3	162	74
Workload (min)	2571744	2830	140960	109145	190703	321462	624866	1042672	4226	205734	98623

Table 3. Parameters of VNF and Physical Node

Parameters	**VNF [25]**	**Physical Node [2]**
Type	Nat, Firewall, Proxy, IDS	HVS (High Volume Server)
CPU	$cpu_{v_p} \in \{2, 4, 4, 8\}$	$C_{n_i}^{cpu} = 16$

In Table 2, we randomly select a subset of base station and present their workload information analyzed from the dataset of Shanghai Telecom base station. In this experiment, we use the workload information of base station as the network traffic in the mobile network.

6.2.2 VNF and Physical Node. We design the parameters of VNF-FG and PNG by reference to the scenarios in the literature [2, 5, 6, 25]. The parameters used can be found in Table 3.

In this article, each SFC request consists of 3 VNFs such as Network Address Translation (NAT), Firewall (FW), Proxy, and Intrusion Detection System (IDS). The SFC requests arrive following the Poisson distribution. We set the average arrival rate to 5 SFC requests per 100 time units. For the physical network, each node is High Volume Server (HVS) with computing and storage resource.

6.3 Resource Utilization

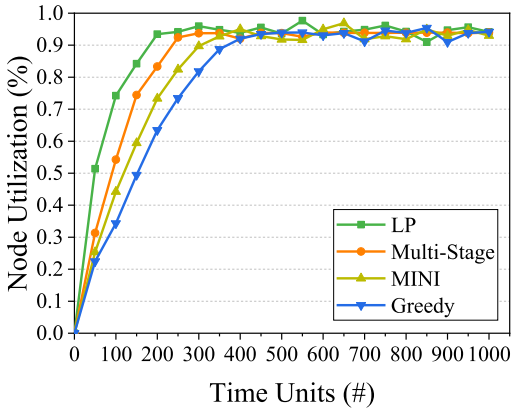
In this article, we focus on resource utilization including physical node utilization and physical link utilization. In this experiment, we evaluate resource utilization of the four placement algorithms in different network topologies (traditional network and mobile edge network). The MINI is aimed at optimizing node utilization without link utilization. Therefore, we do not consider the link utilization of MINI.

In this experiment, the physical node resource contains CPU and memory. As we mentioned (Section 5.2.1), we set a certain ratio between CPU and memory, so CPU and memory utilization are similar. Therefore, we use CPU utilization to indicate physical node utilization.

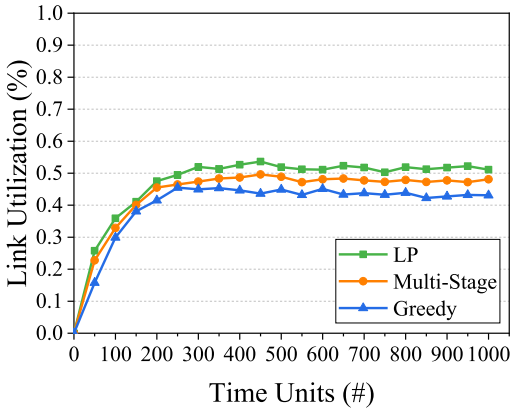
6.3.1 Resource Utilization in Traditional Network. In this subsection, we evaluate resource utilization of the LP algorithm, Multi-Stage algorithm, MINI algorithm (without link utilization), and Greedy algorithm. Since the Multi-Stage algorithm is designed for traditional networks, we first evaluate resource utilization in two traditional networks (Internet2 research network and US mesh network).

Figure 7(a) and (b) shows resource utilization of the four placement algorithms in the Internet2 research network. As Figure 7(a) shows, in the beginning, node utilization of the four algorithms is increasing. Among them, LP rises fastest and Greedy is the slowest. Finally (between 300 and 400 time units), node utilization of the four placement algorithms are the same (about 94%). As Figure 7(b) shows, link utilization of the three algorithms increases as the number of SFC requests increases. We can also observe that LP has the highest link utilization, Multi-Stage is the second, and Greedy is the lowest. However, link utilization of the three algorithms is relatively close (43–52%).

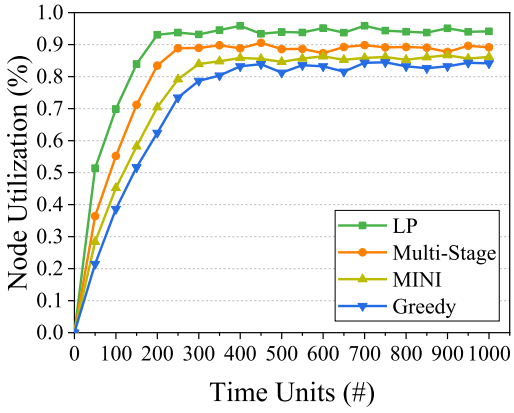
In this experiment, we also evaluate resource utilization of the four placement algorithms in US mesh network, as shown in Figure 7(c) and (d). Compared with the Internet2 research network,



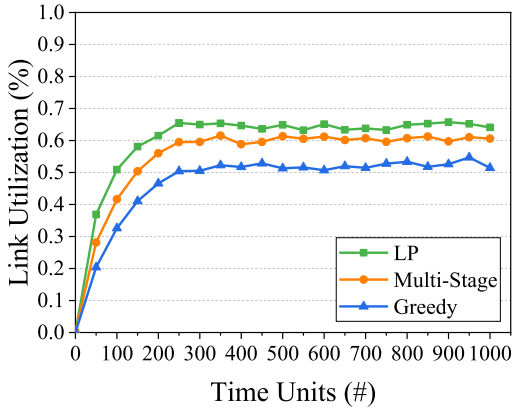
(a) Node Utilization in Internet2 Research Network.



(b) Link Utilization in Internet2 Research Network.



(c) Node Utilization in US Mesh Network.



(d) Link Utilization in US Mesh Network.

Fig. 7. Resource Utilization in Internet2 Research Network and US Mesh Network.

US mesh network has more nodes and links, and the connection rate between nodes is high. As Figure 7(c) shows, LP has the highest node utilization (94%), Multi-Stage is the second (88%), MINI is the third (85%), and Greedy is the lowest (83%). As Figure 7(d) shows, link utilization of the three algorithms are 64%, 60%, and 51%, respectively.

In summary, node utilization of LP outperforms Multi-Stage, MINI, and Greedy in traditional networks. Similarly, we can also conclude that link utilization of LP is better than Multi-Stage and Greedy. Compared with the Internet2 research network, Multi-Stage, MINI, and Greedy perform better in the US mesh network. It means that the higher node connection rate, the easier it is to find the appropriate placement node and link. Different from the other three algorithms, LP behaves the same in both Internet2 research network and US mesh network. It means that our proposed LP solution is suitable for traditional networks with different node connection rates.

6.3.2 Resource Utilization in MEC Network. Since we focus on the SFC placement problem in the MEC-NFV environment, we evaluate resource utilization of the four placement algorithms in the mobile edge network. As we mentioned (Section 6.2.1), we use the dataset of Shanghai Telecom base station as the mobile network. Figure 8(a) and (b) shows resource utilization of the four placement algorithms.

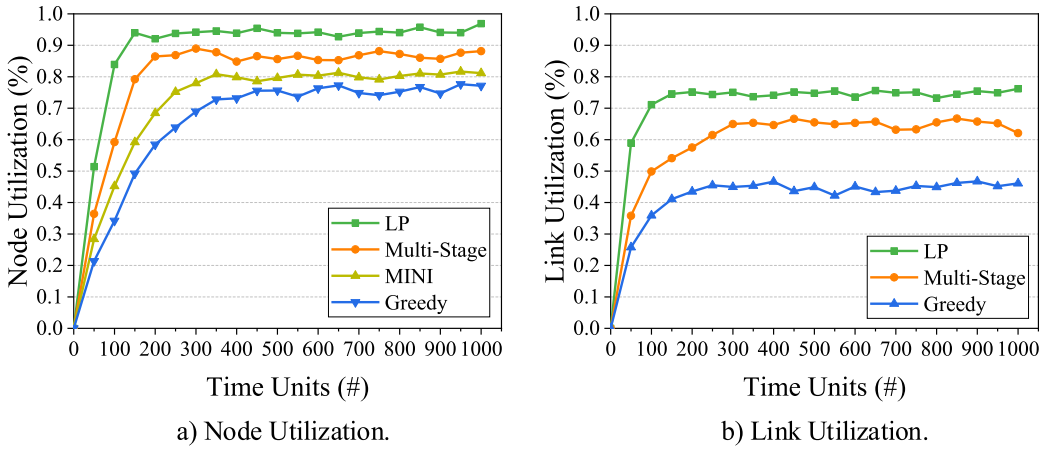


Fig. 8. (a) Physical Node Utilization and (b) Link Utilization in Shanghai Telecom Base Station Mobile Network.

As Figure 8(a) shows, the four placement algorithms can be stabilized after a period of time. The time required for LP (150 time units) to stabilize is shorter than Multi-Stage (200 time units), MINI (350 time units), and Greedy (450 time units). Besides, LP (93%) has a higher node utilization than Multi-Stage (86%), MINI (80%), and Greedy (74%). All of these indicate that LP has a significant effect on optimizing node utilization.

As Figure 8(b) shows, link utilization of LP (74%) is higher than Multi-Stage (65%) and Greedy (44%). We can observe that link utilization is not as high as node utilization. It is because the communication between two nodes may use several links, and the links used by different nodes are likely to be non-coincident. We can conclude that LP is better than Multi-Stage and Greedy in optimizing link utilization.

In summary, our proposed LP approach outperforms Multi-Stage, MINI, and Greedy in the MEC-NFV environment. This proves that the LP-based matching algorithm and the Hungarian-based mapping algorithm can effectively optimize resource utilization. More specifically, the LP-based matching algorithm can calculate the similarity matrix between VNF-FG and PNG. And the Hungarian-based mapping algorithm can select the optimal physical link path and update the similarity matrix in a dynamic network environment.

6.4 Acceptance Ratio

In this subsection, we evaluate the acceptance ratio of the placement algorithms. As we described in Section 6, we use the acceptance ratio to indicate the ratio of the number of accepted SFC requests to the total number of SFC requests. The acceptance ratio is an important indicator for evaluating the performance of the placement algorithm. A high acceptance ratio means a good performance of the placement algorithm.

In Reference [10], we can conclude that the acceptance ratio of MINI is less than 80%. Therefore, we do not compare the acceptance ratio of MINI. As Figure 9 shows, we evaluate the acceptance ratio of the three placement algorithms. In this experiment, we set the PNG size to 100.

Figure 9 shows the acceptance ratio of the three placement algorithms. At the beginning, all three placement algorithms have a high acceptance ratio (close to 1). And the acceptance ratio decreases as the number of SFC requests increases. We can also observe that the acceptance ratio of the three algorithms can be stable after a certain period of time (after 200 time units). In this

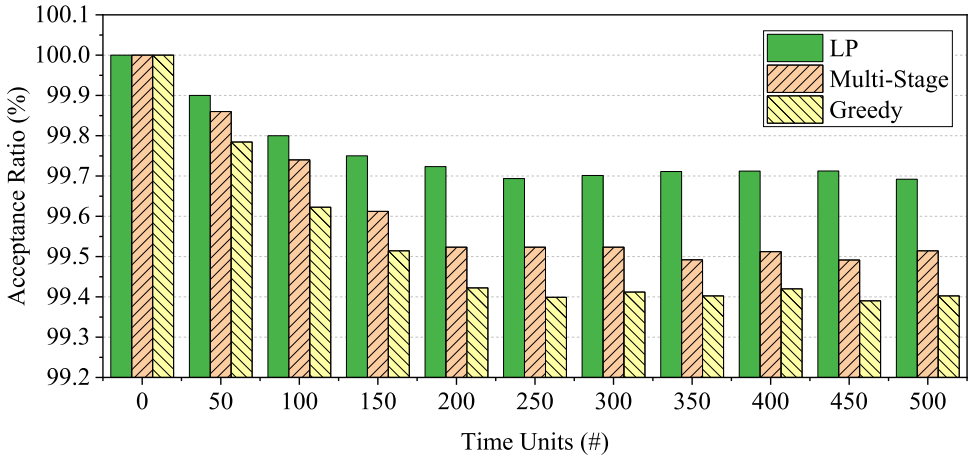


Fig. 9. Acceptance ratio.

experiment, both resource constraints and performance requirements (e.g., latency and availability) can affect the acceptance ratio. More specifically, at the beginning (within 200 time units), the acceptance ratio is declining. The main reason is resource constraints. The more SFC requests, the less empty nodes, resulting in more rejected SFC requests. After a certain period of time (after 200 time units), the acceptance ratio is stable. This is because the more SFC requests, the more empty nodes are occupied. New SFC requests can only be placed on nodes that are already running VNFs (i.e., non-empty node). At this time, the main factor affecting the acceptance ratio is the performance requirements, so the acceptance ratio tends to be stable.

In summary, the acceptance ratio of LP is higher than Multi-Stage and Greedy. After 200 time units, the acceptance ratio of LP is 99.7% while Multi-Stage and Greedy are 99.5% and 99.4%, respectively. It proves that our proposed LP can effectively find the appropriate physical nodes and links to map SFC.

6.5 Execution Time

Among the heuristic algorithms, Multi-Stage can run in polynomial time. Therefore, we compare the execution time of LP, Multi-Stage, and Greedy.

Since the size of physical network is constant (Section 6.2.1), we use a random network whose size can vary depending on our settings. In this experiment, we use the GT-ITM [30] topology generator in NS-2 [31] to randomly generate graphs. The topology of VNF-FG is similar to the physical network, since it is used to describe a network forwarding graph. Therefore, we use the GT-ITM tool to generate PNGs and VNF-FGs with different size. In the random network, we set that 50% of the physical nodes (VNFs) are directly connected. We run each scenario 100 times, so the statistics are almost unaffected by the accidental events.

Figure 10(a) and (b) shows there are many outliers. These outliers indicate that the algorithm often has extreme cases where the algorithm needs to iterate many times. We can conclude that the performance of Greedy is poor, while the performance of Multi-stage and LP is better.

Figure 10(a) shows the execution time of the three placement algorithms with different VNF-FG size (PNG size is 100). We can conclude that our proposed LP runs faster than Multi-Stage and Greedy. It is worth mentioning that the execution time of LP is independent of the VNF-FG size. As we mentioned in Section 5.2.1, we extend the size of VNF-FG adjacency matrix to the same as

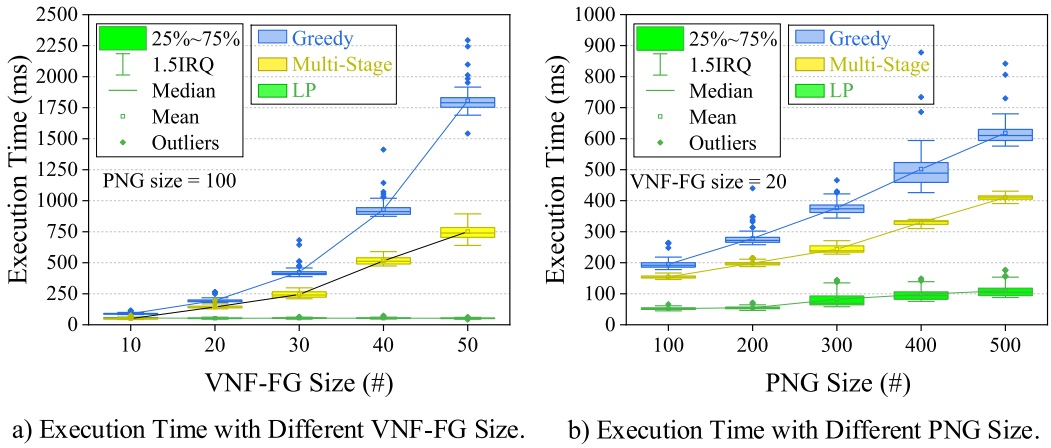


Fig. 10. Execution time with different (a) VNF-FG size and (b) PNG size.

PNG. Therefore, the execution time of LP is related to the size of PNG. In contrast, the execution time of Multi-Stage and Greedy increase dramatically as the VNF-FG size increases.

Figure 10(b) shows the execution time of the three placement algorithms with different PNG size (VNF-FG size is 20). We can conclude that LP runs faster than Multi-Stage and Greedy. We can also observe that the execution time of LP, Multi-Stage and Greedy increase as the PNG size increases. We can also conclude that the execution time of LP increases slowly while Multi-Stage and Greedy is fast.

In conclusion, in the case of different VNF-FG size and PNG size, our proposed LP solution runs faster than Multi-Stage and Greedy. It is worth mentioning that the execution time of LP is not related to the VNF-FG size. And the execution time of LP increases slowly as the PNG size increases, because it is only affected by the size of the physical network adjacency matrix.

7 CONCLUSION

This article focuses on the SFC placement problem in the MEC-NFV environment. Different from the existing works in traditional networks, we formulate the SFC placement problem in the MEC-NFV environment as a WGMP. Then we divide the WGMP into two sub-problems: a graph matching problem and an SFC mapping problem. And we propose an LP-based algorithm and a Hungarian-based algorithm to solve the two sub-problems. Compared with heuristic algorithms such as Multi-Stage and Greedy, evaluation results show that our proposed solutions can efficiently reduce the execution time and optimize resource utilization.

We also recognize the limitations of our research. First, resource utilization and service performance are typically two conflicting objectives. We do not consider the tradeoff between resource utilization and service performance (e.g., use the reachability matrix to improve the end-to-end service delay). Second, we evaluate the proposed solutions in the simulation environment. We do not implement the SFC placement algorithms in the real MEC-NFV environment. Therefore, our future work will focus on the above aspects.

REFERENCES

[1] ETSI GS NFV 003. 2014. Network Functions Virtualisation (NFV): Terminology for Main Concepts in NFV. Retrieved from https://www.etsi.org/deliver/etsi_gs/nfv/001_099/003/01.02.01_60/gs_nfv003v010201p.pdf.

[2] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. In *Proceedings of the ACM Special Interest Group on Data Communication Conference (SIGCOMM'08).*

[3] H. A. Almohamad and Salih O. Duffuaa. 1993. A linear programming approach for the weighted graph matching problem. *IEEE Trans. Pattern Anal. Mach. Intell.* 15 (1993), 522–525.
[4] Monarch Network Architects. 2012. Sample Optical Network Topology Files. Retrieved from http://www.monarchna.com/.
[5] Md. Faizul Bari, Shihabur Rahman Chowdhury, Reaz Ahmed, and Raouf Boutaba. 2015. On orchestrating virtual network functions. In *Proceedings of the 2015 11th International Conference on Network and Service Management (CNSM'15)*, 50–56.
[6] Md. Faizul Bari, Shihabur Rahman Chowdhury, Reaz Ahmed, Raouf Boutaba, and Otto Carlos Muniz Bandeira Duarte. 2016. Orchestrating virtualized network functions. *IEEE Trans. Netw. Serv. Manage.* 13 (2016), 725–739.
[7] Michael Till Beck and Juan Felipe Botero. 2017. Scalable and coordinated allocation of service function chains. *Comput. Commun.* 102 (2017), 78–88.
[8] Ilias Benkacem, Tarik Taleb, Miloud Bagaa, and Hannu Flinck. 2018. Optimal VNFs placement in CDN slicing over multi-cloud environment. *IEEE J. Select. Areas Commun.* 36 (2018), 616–627.
[9] Yishan Chen, Shuiguang Deng, Hongtao Ma, and Jianwei Yin. 2020. Deploying data-intensive applications with multiple services components on edge. *Mobile Netw. Appl.* 25 (2020), 426–441.
[10] Zhiqi Chen, Sheng Zhang, Can Wang, Zhuzhong Qian, Mingjun Xiao, Jie Wu, and Imad Jawhar. 2018. A novel algorithm for NFV chain placement in edge computing environments. In *Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM'18)*, 1–6.
[11] Margaret Chiosi, Steve Wright, Javan Erfanian, and Brian Smith. SDN and OpenFlow World Congress. 2012. Network Functions Virtualisation (NFV). Retrieved from http://portal.etsi.org/NFV/NFV_White_Paper.pdf.
[12] The Internet2 community. 2012. Internet2 research network. Retrieved from https://www.internet2.edu/.
[13] Richard Cziva, Christos Anagnostopoulos, and Dimitrios P. Pezaros. 2018. Dynamic, latency-optimal vNF placement at the network edge. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'18)*, 693–701.
[14] Shuiguang Deng, Zhengzhe Xiang, Javid Taheri, Khoshkholghi Ali Mohammad, Jianwei Yin, Albert Zomaya, and Schahram Dustdar. 2020. Optimal application deployment in resource constrained distributed edges. *IEEE Trans. Mobile Comput.* Early Access (2020), 1–1.
[15] Shuiguang Deng, Zhengzhe Xiang, Peng Zhao, Javid Taheri, Honghao Gao, Jianwei Yin, and Albert Y. Zomaya. 2020. Dynamical resource allocation in edge for trustable IoT systems: A reinforcement learning method. *IEEE Trans. Industr. Inf.* 16 (2020), 6103–6113.
[16] Juliver Gil-Herrera and Juan Felipe Botero. 2016. Resource allocation in NFV: A comprehensive survey. *IEEE Trans. Netw. Serv. Manage.* 13 (2016), 518–532.
[17] Yan Guo, Shangguang Wang, Ao Zhou, Jinliang Xu, Jie Yuan, and Ching-Hsien Hsu. 2020. User allocation-aware edge cloud placement in mobile edge computing. *Software: Practice and Experience* 50 (2020), 489–502.
[18] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. 2015. Mobile Edge Computing: A Key Technology Towards 5G. Retrieved from https://www.etsi.org/images/files/etsiwhitepapers/etsi_wp11_mec_a_key_technology_towards_5g.pdf.
[19] Fatma Ben Jemaa, Guy Pujolle, and Michel Pariente. 2016. QoS-aware VNF placement optimization in edge-central carrier cloud architecture. In *Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM'16)*, 1–7.
[20] Jian Kong, Inwoong Kim, Xi Wang, Qiong Zhang, Hakki C. Cankaya, Weisheng Xie, Tadashi Ikeuchi, and Jason P. Jue. 2017. Guaranteed-availability network function virtualization with network protection and VNF replication. In *Proceedings of the 2017 IEEE Global Communications Conference (GLOBECOM'17)*, 1–6.
[21] Abdelquoddouss Laghrissi, Tarik Taleb, Miloud Bagaa, and Hannu Flinck. 2017. Towards edge slicing: VNF placement algorithms for a dynamic and realistic edge cloud environment. In *Proceedings of the 2017 IEEE Global Communications Conference (GLOBECOM'17)*, 1–6.
[22] Yuanzhe Li and Shangguang Wang. 2018. An energy-aware edge server placement algorithm in mobile edge computing. In *Proceedings of the 2018 IEEE International Conference on Edge Computing (EDGE'18)*, 66–73.
[23] Redowan Mahmud, Kotagiri Ramamohanarao, and Rajkumar Buyya. 2018. Latency-aware application module management for fog computing environments. *ACM Trans. Internet Technol.* 19 (2018), 9:1–9:21.
[24] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled Ben Letaief. 2017. A survey on mobile edge computing: The communication perspective. *IEEE Commun. Surv. Tutor.* 19 (2017), 2322–2358.
[25] João G. Martins, Mohamed Ahmed, Costin Raiciu, Vladimir Andrei Olteanu, Michio Honda, Roberto Bifulco, and Felipe Huici. 2014. ClickOS and the art of network function virtualization. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI'14)*.
[26] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. 2016. Network function virtualization: State-of-the-art and research challenges. *IEEE Commun. Surv. Tutor.* 18 (2016), 236–262.
[27] Pawani Porambage, Jude Okwuibe, Madhusanka Liyanage, Mika Ylianttila, and Tarik Taleb. 2018. Survey on multi-access edge computing for internet of things realization. *IEEE Commun. Surv. Tutor.* 20 (2018), 2961–2991.

[28] Gamal Sallam and Bo Ji. 2019. Joint placement and allocation of virtual network functions with budget and capacity constraints. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'19)* (2019), 523–531.
[29] S. Song, C. Lee, H. Cho, G. Lim, and J. Chung. 2019. Clustered virtualized network functions resource allocation based on context-aware grouping in 5G edge networks. *IEEE Trans. Mobile Comput.* 19 (2019), 1072–1083.
[30] University of Southern California2001. Retrieved from GT-ITM topology generator, https://www.isi.edu/nsnam/ns/ns-topogen.html.
[31] University of Southern California. 2001. The Network Simulator—ns-2. Retrieved from https://www.isi.edu/nsnam/ns/.
[32] Shangguang Wang, Yan Guo, Ning Zhang, Peng Yang, Ao Zhou, and Xuemin Sherman Shen. 2019. Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach. *IEEE Trans. Mobile Comput.* Early Access (2019), 1–1.
[33] Shangguang Wang, Yali Zhao, Lin Huang, Jinliang Xu, and Ching-Hsien Hsu. 2019. QoS prediction for service recommendations in mobile edge computing. *J. Parallel Distrib. Comput.* 127 (2019), 134–144.
[34] Shangguang Wang, Yali Zhao, Jinlinag Xu, Jie Yuan, and Ching-Hsien Hsu. 2019. Edge server placement in mobile edge computing. *J. Parallel Distrib. Comput.* 127 (2019), 160–168.
[35] Zhengzhe Xiang, Shuiguang Deng, Javid Taheri, and Albert Zomaya. 2020. Dynamical service deployment and replacement in resource-constrained edges. *Mobile Netw. Appl.* 25 (2020), 674–689.
[36] Jinliang Xu, Shangguang Wang, Bharat K. Bhargava, and Fangchun Yang. 2019. A blockchain-enabled trustless crowd-intelligence ecosystem on mobile edge computing. *IEEE Trans. Industr. Inf.* 15 (2019), 3538–3547.
[37] Louiza Yala, Pantelis A. Frangoudis, and Adlen Ksentini. 2018. Latency and availability driven VNF placement in a MEC-NFV environment. In *Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM'18)*, 1–7.
[38] Qiang Ye, Weihua Zhuang, Xu Li, and Jaya Rao. 2019. End-to-end delay modeling for embedded VNF chains in 5G core networks. *IEEE IoT J.* 6 (2019), 692–704.
[39] Zilong Ye, Xiaojun Cao, Jianping Wang, Hong-Fang Yu, and Chunming Qiao. 2016. Joint topology design and mapping of service function chains for efficient, scalable, and reliable network functions virtualization. *IEEE Netw.* 30 (2016), 81–87.
[40] Cheng Zhang, Hailiang Zhao, and Shuiguang Deng. 2018. A density-based offloading strategy for IoT devices in edge computing systems. *IEEE Access* 6 (2018), 73520–73530.
[41] Zhilong Zheng, Jianzhao Bi, Heng Yu, Haiping Wang, Chen Sun, Hongxin Hu, and Jianping Wu. 2019. Octans: Optimal placement of service function chains in many-core systems. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'19)* (2019), 307–315.

Received September 2019; revised January 2020; accepted March 2020