Poster:A Linear Programming Approach for SFC Placement in Mobile Edge Computing

Meng Wang, Bo Cheng*, Junliang Chen
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications, Beijing, China
{mengwang,chengbo,chjl}@bupt.edu.cn

ABSTRACT

Mobile Edge Computing (MEC) is a promising architecture where network services are deployed to the network edge. Recent studies tend to deploy Network Function Virtualization (NFV) services to MEC. Network services in NFV are deployed as Service Function Chains (SFCs). In this paper, we focus on the SFC placement problem in a MEC-NFV environment, which is different from the data center network. Firstly, we formulate this problem as a weighted graph matching problem consisting of graph matching and SFC mapping. Then, we propose a linear programming-based approach to match the edge network and SFC. Finally, we design a Hungarian-based placement algorithm to map SFC in the edge network. A heuristic-based greedy algorithm is also designed to compare the performance. Evaluation results show that our proposed solutions outperform the greedy algorithm in terms of execution time.

CCS CONCEPTS

• **Networks → Network resources allocation**; • **Human-centered computing → Mobile computing**.

KEYWORDS

SFC; Placement; MEC; NFV; Linear Programming

ACM Reference Format:
Meng Wang, Bo Cheng*, Junliang Chen. 2019. Poster:A Linear Programming Approach for SFC Placement in Mobile Edge Computing. In *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom '19), October 21–25, 2019, Los Cabos, Mexico.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3300061.3343394

*Bo Cheng is the corresponding author.

MobiCom '19, October 21–25, 2019, Los Cabos, Mexico

ACM ISBN 978-1-4503-6169-9/19/10.
https://doi.org/10.1145/3300061.3343394

1 INTRODUCTION

Driven by 5G communications, smart devices and IoT sensors, applications with extreme requirements are increasing. Mobile Edge Computing (MEC) and Network Function Virtualization (NFV) are emerging as two core technologies to satisfy the network service demands.

NFV is an emerging technology that decouples network functions from the hardware. By bringing great convenience and flexible provisioning of Virtual Network Function (VNF), NFV significantly decreases the Capital Expenditure (CAPEX) and Operating Expense (OPEX). Based on the application requirements, the NFV chain is described as a Service Function Chain (SFC), which consists of an ordered set of VNFs.

In this paper, we focus on the SFC placement problem at edge servers. In MEC, the demands of SFC requests are more complex and the edge network is dynamic. Therefore, SFC placement problem in MEC becomes an important but difficult problem, which has received increasing attention [3]. Most of the existing works propose heuristic algorithms to solve the SFC placement problem in MEC. However, heuristic approaches iteratively solve the problem and it can affect the quality of the solutions and increase the time to find a suboptimal solution.

Given these facts, we formulate the SFC placement problem in the MEC-NFV environment as the Weighted Graph Matching Problem (WGMP) [2]. Then we propose a Linear Programming (LP)-based approach to solve the WGMP.

In summary, the main contributions are as follows:

- Formulate the SFC placement problem as the WGMP, and formulate the mobile edge network and SFC as two weighted graphs.
- Propose an LP-based graph matching approach and a Hungarian-based SFC mapping algorithm. Our proposed solutions can run in polynomial time.
- Design a bipartite graph matching based greedy algorithm as the baseline. And our proposed solutions outperform the greedy algorithm.

The rest of this paper is organized as follows. Section 2 formulates the SFC placement problem. Section 3 discusses the proposed solutions. Section 4 evaluates the performance. Finally, Section 5 concludes this study.

2 PROBLEM FORMULATION

In this paper, we formulate the physical network and SFC request as Physical Network Graph (PNG) and VNF Forwarding Graph (VNF-FG), respectively.

The physical network can be considered as a PNG $P = (N, L)$. N is the set of physical nodes and L is the set of physical links between two physical nodes. $C_{n_i}^{cpu}$ and $C_{n_i}^{mem}$ indicate the CPU and memory capacity of physical node $n_i \in N$. $C_{l_{ij}}^{bw}$ is the bandwidth capacity of physical link $l_{ij} \in L$.

We formulate SFC request as a VNF-FG $F = (V, E)$. V indicates the set of VNFs and E indicates the set of logical links between two VNFs. cpu_{v_p} and mem_{v_p} indicate the CPU and memory consumption of VNF $v_p \in V$. And $bw_{e_{pq}}$ is the requested bandwidth of logical link $e_{pq} \in E$.

3 PROPOSED SOLUTIONS

In this section, we propose the LP-based approach and a Hungarian-based algorithm to solve the WGMP. While LP in WGMP [2] is an old idea, it has not been widely applied to the SFC placement in MEC.

Algorithm 1: LP-based Graph Matching

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

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

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

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

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

3 **Step3:** Solve the LP problem in *Eq.* 4 by using the Simplex method. The matrix B is define by:

$$b_{ij} = \begin{cases} 1 & \text{for } i = 1, 2, 3, \cdots, n \text{ and } j = i, i+n, i+2n, \cdots, i+(n-1)n \\ 0 & \text{otherwise} \end{cases}$$

4 **Step4:** Get the similarity matrix M from vector m (generated at Step 3);

5 **Step5:** Map SFC by using the Hungarian-based algorithm;

3.1 LP-based Graph Matching

3.1.1 Adjacency Matrix (Step 1). At Step 1, we use A_P and A_F to indicate the adjacency matrices of PNG and VNF-FG. p_{ii} indicates the CPU capacity of node n_i. p_{ij} indicates the bandwidth capacity between node n_i and n_j (so is f_{ii} and f_{ij}). We compute the weight between nodes (or VNFs) that are not directly connected based on the Dijkstra method.

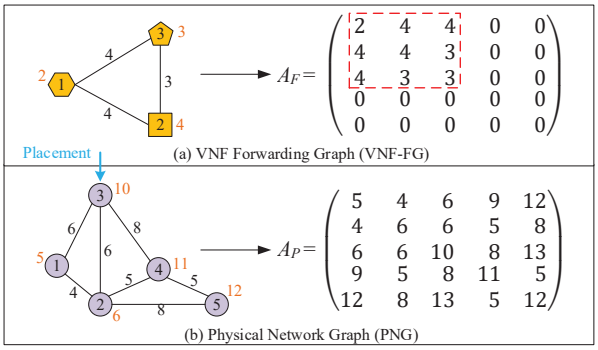


Figure 1: (a) VNF-FG; (b) PNG.

For example, PNG is a $5 * 5$ matrix A_P, as shown in Figure 1(b). The adjacency matrix $(3 * 3)$ of VNF-FG is in the red dotted line box, as shown in Figure 1(a). We extend it to a $5 * 5$ matrix A_F (the newly added elements are all zeros).

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

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

Where $p(n_i, n_j)$ indicates the weight between n_i and n_j. Φ is the one-to-one correspondence between node and VNF.

The goal of WGMP is to minimize the distance $J(\Phi)$. Therefore, the matching problem can be defined as:

$$\min_{M} \left\| A_P - MA_FM^{\mathrm{T}} \right\|_1 \tag{2}$$

Where permutation matrix M indicates the mapping function Φ. A_P and A_F are the adjacency matrices of two weighted graphs. Note that $||.||$ is the L_1 norm.

According to [2], we can conclude that the minimization problem in *Eq.* 2 is equivalent to *Eq.* 3:

$$\min_{m} \|A_{PF}m\|_1 \qquad m \geq 0 \tag{3}$$

Where $m = VEC(M^{\mathrm{T}})$ indicates an $n^2 * 1$ vector.

3.1.3 Linear Programming (Step 3). Finally, we reformulate the minimization problem in *Eq.* 3 as an LP problem:

$$\begin{aligned} &\min_{m,S,T} \sum_{i}^{n^2} S_i + T_i \\ s.t. \quad &A_{PF}m + S - T = 0 \\ &Bm = e \\ &m \geq 0, S \geq 0, T \geq 0 \end{aligned} \tag{4}$$

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

Therefore, we solve the LP problem in *Eq.* 4 by using the Simplex method (Step 3 in Algorithm 1).

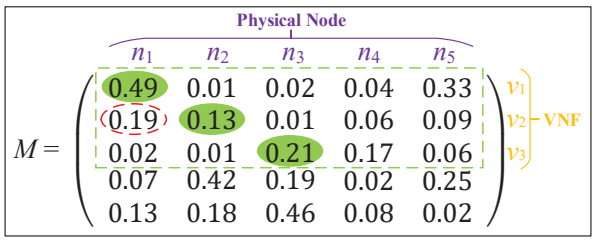


Figure 2: Similarity Matrix *M*.

3.1.4 Similarity Matrix (Step 4). At Step 4, we can get the similarity matrix M from vector m. As Figure 2 shows, data in the green dotted line box indicates the similarity between physical nodes and VNFs. The columns and rows of M correspond to physical nodes and VNFs, respectively. For example, m_{12} indicates the similarity between VNF v_1 and node n_2.

3.1.5 Hungarian-based SFC Mapping (Step 5). At Step 5, we design Hungarian-based algorithm to map SFC. According to the highest similarity in Figure 2, the optimal match should be that v_1 and v_2 are mapped to n_1, and v_3 is mapped to n_3. However, the resource capacity of n_1 cannot satisfy v_1 and v_2 ($C^{cpu}_{n_1} = 5$ while $cpu_{v_1} = 2$ and $cpu_{v_2} = 4$). Therefore, v_2 should be mapped to n_2 since the similarity between n_2 and v_2 is the next highest. In addition, we take memory and bandwidth constraints into consideration.

4 PERFORMANCE EVALUATION

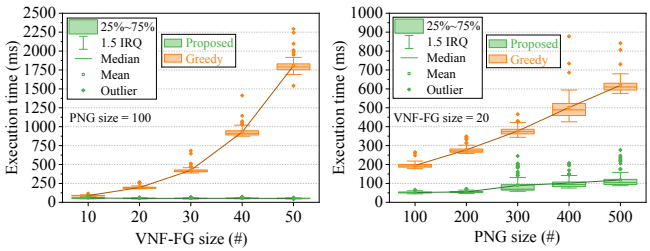
4.1 Simulation Setup

We evaluate the performance using a laptop of Windows 10 with 2.2 GHz Intel Core i5 processor and 8 GB memory. We implement our proposed LP-based approach and Greedy algorithm in Java based on Alevin [4], a wider simulation environment for NFV resource allocation.

Table 1: Parameters of PNG and VNF-FG.

Parameters	PNG	VNF-FG
Size	$N \in [100-500]$	$V \in [10-50]$
Node Resource	$C^{cpu}_{n_i} \in [50-100]$	$cpu_{v_p} \in [0-20]$
Link Resource	$C^{bw}_{l_{ij}} \in [50-100]$	$bw_{e_{pq}} \in [0-20]$
Connectivity	$\alpha = 0.5$	$\alpha = 0.5$

We run each scenario 100 times, so the statistics are almost unaffected by the accidental events. We use the GT-ITM [1] topology generator in NS-2 to randomly generate PNGs and VNF-FGs. 50% of the nodes (VNFs) are directly connected. The parameters of PNG and VNF-FG used in the simulation can be found in Table 1.



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

Figure 3: Execution Time.

4.2 Execution Time

As Figure 3(a) shows, we evaluate the execution time of LP and Greedy with different VNF-FG size (PNG size is 100). It is worth mentioning that the execution time of LP is independent of the VNF-FG size. And we can conclude that LP runs faster than Greedy.

Figure 3(b) shows the LP runs faster than Greedy with different PNG size (VNF-FG size is 20). We can also observe that the execution time of both LP and Greedy increases as PNG size increases. And the execution time of LP increases slowly while Greedy is fast.

5 CONCLUSION

In this paper, we formulate the SFC placement problem as the WGMP. And we propose an LP-based approach and a Hungarian-based algorithm to solve this problem. We also design a heuristic-based greedy algorithm to compare the performance. Evaluation results show that our proposed solutions can efficiently reduce the execution time. In future work, we focus on resource optimization in SFC placement.

6 ACKNOWLEDGMENTS

This work was supported in part by the BUPT Excellent Ph.D. Students Foundation (Grant No.CX2019214), in part by the National Key Research and Development Program of China (Grant No.2017YFB1400603), and in part by the Natural Science Foundation of China (Grant No.61772479).

REFERENCES

[1] [n.d.]. *GT-ITM Topology Generator.* https://www.isi.edu/nsnam/ns/ns-topogen.html#gt-itm

[2] 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.

[3] Richard Cziva, Christos Anagnostopoulos, and Dimitrios P. Pezaros. 2018. Dynamic, Latency-Optimal vNF Placement at the Network Edge. *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications* (2018), 693–701.

[4] Juliver Gil-Herrera and Juan Felipe Botero. 2016. Resource Allocation in NFV: A Comprehensive Survey. *IEEE Transactions on Network and Service Management* 13 (2016), 518–532.