Network Service Embedding in Multiple Edge Systems: Profit Maximization by Federation

Yu-Chen Tai and Li-Hsing Yen
Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan.
Email: {taiyuchen.cs07g, lhyen}@nctu.edu.tw

***Abstract*—A service chain (SC) comprises a series of service functions realized as virtualized network functions (VNFs). Each VNF has specific resource, bandwidth, and location requirements. This study assumes multiple SCs to be placed on geo-distributed heterogeneous edge servers owned by multiple edge service providers (ESPs). Each ESP selectively hosts SCs to earn profit. We consider the problem to maximize the total profit (i.e., social welfare) of all ESPs by organizing ESPs into a set of disjoint federations called federation structure. We formulate the problem of finding an optimal federation structure, identify the difficulties in solving the problem, and present two time-efficient heuristics as our approach. Simulation results show that elaborate federation structures do bring in higher social welfare compared with simple all-in or all-out ESP organization. The proposed heuristics can also approximate the optimal result in many cases.**

I. Introduction

Service Function Chaining (SFC) is to link independent service functions (SFs) together in a particular order to form a service chain (SC), which enacts a specific network service (NS) in data network. SFC has been a prominent use case of and tightly coupled with Network Functions Virtualization (NFV) [1], which realizes SFs as software modules known as virtualized network functions (VNFs). VNFs run on commodity servers and can be boosted by cloud-based virtualization technologies. With the support of software defined networking (SDN), NFV/SFC enables dynamic instantiation, configuration, and termination of NSs, which significantly enhances scalability and elasticity. It also facilitates innovative network services such as network slicing [2].

Network service embedding (NSE) is to deploy SC to an NFV infrastructure. There have been many studies on NSE in cloud datacenter (NSE-Cloud) [3], [4]. In contrast, this study assumes edge system as NFV infrastructure. Edge system brings cloud service to the proximity of end users by leveraging geo-distributed edge servers. NSE in the edge is essential to latency-sensitive, location-dependent, or context-aware applications. For instance, the cache server of content delivery network (CDN) in the edge can provide better user experience [5]. Also, a firewall placed in edge can preserve a greater amount of bandwidth when a denial-of-service (DoS) attack occurs [6].

Figure 1 shows the target environment of this study. We assume multiple SCs to be placed on geo-distributed heterogeneous edge servers owned by multiple edge service providers (ESPs). Each ESP receives payments but also incurs operational costs from hosting SCs.

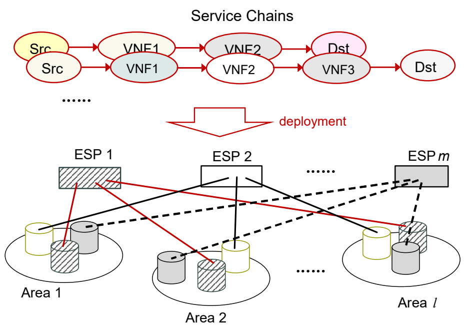


Fig. 1: Multiple SCs embedded to geo-distributed edge servers owned by multiple ESPs

Individual Edge Profit Maximization (IEPM) problem is to maximize an individual ESP's profit by hosting a selected set of SCs. The selection is subject to computation resource, bandwidth, and location constraints. The IEPM problem possesses some unique features that differentiate it from NSE-Cloud such as limited computation resource and bandwidth, heterogeneous server costs, and dispersed server locations. However, it still shares some properties with NSE-Cloud. For example, some type of VNF (such as firewall or video codec [7]) may have different bandwidth demands for inflow and outflow traffics, respectively (i.e., *traffic scaling* [8]). IEPM is at least as hard as NSE-Cloud, because NSE-Cloud is only a special case of IEPM (NSE-Cloud assumes homogeneous servers with vast resource and uniform cost within a single area).

We further consider the problem to maximize the total profit (i.e., *social welfare*) of all ESPs by forming disjoint ESP *federations*. An ESP federation is a set of ESPs that share their resources for NSE. When considerable SCs come in for possible deployment, it is possible that no ESP alone can host some profitable SC due to inadequate computing or bandwidth resource in some area. If two or more ESPs could form a federation, they may earn extra profit and increase the social welfare by jointly hosting the SC. Federation Profit Maximization (FPM) problem is to maximize the profit of a particular federation by hosting a selected set of SCs. It is like the IEPM problem with an additional consideration of extra federation maintenance cost [9].

Our main problem, *social welfare maximization* (SWM), is to identify a *federation structure* that maximizes the social welfare. A federation structure is a partition of all ESPs into a set of disjoint federations. SWM consists of two parts. One part is to enumerate all possible federation structures, which is challenging because the number of possible federation structures exhibits exponential growth. The other part is to figure out the profit of each federation structure. This part cannot be directly decomposed into independent FPM problems because of the conflict of interest among federations within the same federation structure regarding their own profit maximization.

We formally formulate the SWM problem, discuss the difficulties in solving it, and present two time-efficient heuristics as our approach. We conducted simulations to compare the performance of the proposed heuristics against simple federation structures such as grand federation (which includes all ESPs in it) and singleton federations (where all ESPs work alone). Simulation results show that a well-structured federation structure generally increases social welfare. In many cases, the proposed heuristics also approximate the optimal result.

The rest of this paper is organized as follows: Sec. II reviews related work, presents the system model, and formulates the problem. The following section details the issues of the SWM problem and presents the proposed approach. Sec. IV shows the simulation results and the last session concludes this paper.

II. Background and Problem Formulation

A. Related Work

NSE mainly comprises VNF placement and VNF chaining. VNF placement allocates a server with associated resource to each VNF of the SC to be deployed while VNF chaining links together all these VNFs with guaranteed bandwidth or end-to-end (E2E) latency. The objectives of NSE are diverse. Some typical examples are operational cost and service latency minimizations. Cost minimization can be achieved by consolidating multiple VNFs on the same host [3] or sharing a VNF among multiple SCs [4]. Latency minimization can be achieved by replicating VNF instances for load balancing [10].

Some prior work has addressed VNF deployment on edge servers [11], [12] but did not consider the chaining of multiple VNFs. There have been some studies on SC deployment on geo-distributed infrastructure. The work by Dietrich et al. [8] aims at NSE with a set of heterogeneous geo-distributed datacenters. Their approach partitions a network function into components, each is embedded into a datacenter. The goal is to minimize overall cost while achieving load balancing among datacenters. Jia et al. [13] jointly optimized instance provisioning and traffic routing for service chaining in geo-distributed datacenters. They aimed at minimizing the cost of NS providers. Zhou et al. [14] considered a similar problem in an environment where one central cloud and multiple edge clouds coexist. Compared with central cloud, edge clouds have significantly constrained resource capacity. Leivadeas et al. [15] also addressed VNF placement problem in a hybrid cloud-edge environment with a goal to minimize SC deployment cost as well as E2E communication delay. They specifically considered location requirements imposed by VNFs.

All the mentioned papers assumed one infrastructure provider. Benkacem et al. [16] considered VNF deployment for CDN on the top of multiple public clouds. The proposed approach allows the CDN provider to make a possible trade-off between deployment cost and quality of user experience. Chen and Yen [17] introduced a business model between ESPs and network service providers (NSPs), where NSPs lease cloud resources from ESPs for NS deployments. The authors adapted a matching mechanism for the dispatch of NSs to ESPs and another matching mechanism for the deployment of VNFs to edge servers.

This paper follows the model proposed in [18], which studied the federation of ESPs for resource provisioning to multiple requesters. However, the work in [18] considered general resource allocation without regarding to NFV-based NS deployment. The approach to federation structure generation in [18] (merge-and-split) is also different from ours.

B. System Model

We assume n ESPs $P = \{p_1, p_2, \ldots, p_n\}$ and l different areas $A = \{a_1, a_2, \ldots, a_l\}$ in the system. We consider t different types of computing resources (CPU, memory, storage, etc.) numbered from 1 to t. For each ESP $p_i \in P$, let $R^i(k, r)$ denote p_i's capacity of Type-r resource in area $a_k \in A$, where $1 \leq r \leq t$. Likewise, we use $B^i(p, q)$ to denote ESP p_i's bandwidth capacity between two areas a_p and a_q. Every ESP has its own cost structure. We use $c_i^r \in \mathbb{Z}^+$ and $c_i^{\mathrm{b}} \in \mathbb{Z}^+$ to denote p_i's unit costs of Type-r computing resource and bandwidth, respectively. All these physical resources on the edge server can be virtualized into a pool of virtual resource managed by Virtualized Infrastructure Manager (VIM), which is one of the major functional blocks of NFV Management and Orchestration Architecture (NFV-MANO) [19].

On the other side, we assume m SCs $S = \{s_1, s_2, \ldots, s_m\}$ to be deployed. Let $s_j \in S$ consist of δ_j VNFs. Each VNF is characterized by its resource demand and designated deployment area. Let the Virtual Deployment Unit (VDU) descriptor [20] of the v-th VNF in s_j be denoted by a tuple $\mathcal{V}_j^v = (q_1^{j,v}, q_2^{j,v}, \ldots, q_t^{j,v}, \alpha^{j,v}, \beta^{j,v})$, where $q_r^{j,v}$, $1 \leq r \leq t$, is the amount of Type-r resource required, $\alpha^{j,v} \in A$ is the designated area, and $\beta^{j,v}$ is the bandwidth required from areas $\alpha^{j,v}$ to $\alpha^{j,v+1}$, the designated area of the next VNF. For the last VNF in s_j, the last term in the tuple is the bandwidth required from α^{j,δ_j} (the area of the egress node) to $\alpha^{j,1}$ (the area of the ingress node). Each SC $s_j \in S$ is specified as a chaining sequence of VNFs associated with a payment $\rho_j \in \mathbb{Z}^+$ that will be given to the ESP that accommodates s_j.

C. Social Welfare Maximization (SWM)

For a federation structure with Φ federations $\mathcal{F} = \{F_1, F_2, \ldots, F_\Phi\}$, the profit of each federation $F_\phi \in \mathcal{F}$ is the total payment collected from all SCs hosted by F_ϕ minus the associated cost. The cost comprises two parts. One is associated with resource consumption, which is proportional to the amount of consumed computing and networking resources.

Let $y^i_{k,r}$ be the amount of Type-r resource in area a_k that is allocated by ESP p_i to accommodate SCs. Define $y^i_r = \sum_k y^i_{k,r}$ to be the total amount of p_i's Type-r resource allocated for SCs in all areas. Then p_i's total cost on computing resource is $\sum_r (y^i_r c^r_i)$. Let $z^i_{p,q}$ be the amount of link bandwidth between two areas a_p and a_q allocated by ESP p_i for SCs. Then p_i's total cost on bandwidth consumption is $\sum_{(a_p,a_q)\in A^2}(z^i_{p,q} c^{\text{b}}_i)$.

The other is additional maintenance cost associated with the deployment of SCs across different ESPs. We model this part as some power of the cardinality of the federation. Let μ be the maintenance cost with a federation of size two. Then the maintenance cost with F_ϕ is $\mu(|F_\phi| - 1)^\gamma$, where γ is the growth rate of the maintenance cost.

For any federation $F_\phi \in \mathcal{F}$, let x^j_ϕ be an indicator variable such that $x^j_\phi = 1$ if F_ϕ serves $s_j \in S$ and $x^j_\phi = 0$ otherwise. The service decision of F_ϕ can be represented as $X_\phi = \{x^1_\phi, x^2_\phi, \ldots, x^m_\phi\}$. The utility of $F_\phi \in \mathcal{F}$ with a decision X_ϕ is the profit that F_ϕ can earn with X_ϕ. Formally,

$$\nu_\phi(X_\phi) = \sum_{s_j\in S} x^j_\phi \left(\rho_j - \mu\left(|F_\phi| - 1\right)^\gamma\right) - \sum_{p_i\in F_\phi}\left(\sum_{r=1}^{t} y^i_r \cdot c^r_i + \sum_{(a_p,a_q)\in A^2} z^i_{p,q}\cdot c^{\text{b}}_i\right). \quad (1)$$

Define $X = \bigcup_{\phi=1}^{\Phi} X_\phi$. The social welfare of $\mathcal{F}$ with X is defined as

$$u(\mathcal{F}, X) = \sum_{F_\phi\in\mathcal{F}} \nu_\phi(X_\phi). \quad (2)$$

The SWM problem is formally defined as follows. Let $\mathbb{F}(P)$ be the set of all possible federation structures that can be formed on P. The problem is to find out a federation structure $\mathcal{F} \in \mathbb{F}(P)$ and a dispatch of SCs to all federations in $\mathcal{F}$ (i.e., determining the contents of X) such that the social welfare is maximized, i.e.,

$$(\hat{\mathcal{F}}, \hat{X}) = \underset{\mathcal{F}\in\mathbb{F}(P),X}{\arg\max}\; u(\mathcal{F}, X). \quad (3)$$

The problem is subject to the following constraints.

$$\sum_{\phi=1}^{\Phi} x^j_\phi \le 1,\ \forall s_j \in S, \quad (4)$$

$$y^i_{k,r} \le R^i(k, r),\ \forall p_i \in P, a_k \in A, 1 \le r \le t, \quad (5)$$

$$z^i_{p,q} + z^i_{q,p} \le B^i(p, q),\ \forall p_i \in P, (a_p, a_q) \in A^2, \quad (6)$$

$$\sum_{p_i\in F_\phi} y^i_{k,r} \ge \sum_{j=1}^{m} x^j_\phi \cdot \sum_{v=1}^{\delta_j} \alpha^{j,v}_k \cdot q^{j,v}_r,\quad \forall F_\phi \in \mathcal{F}, a_k \in A, 1 \le r \le t, \quad (7)$$

and

$$\sum_{p_i\in F_\phi} z^i_{p,q} \ge \sum_{j=1}^{m} x^j_\phi \cdot \sum_{v=1}^{\delta_j-1} \alpha^{j,v}_p \cdot \alpha^{j,v+1}_q \cdot \beta^{j,v} + \alpha^{j,\delta_j-1}_p \cdot \alpha^{j,1}_q \cdot \beta^{j,\delta_j-1},\quad \forall F_\phi \in \mathcal{F}, (a_p, a_q) \in A^2 \wedge p \ne q. \quad (8)$$

Eq. (4) ensures that an SC can be served by at most one federation. Eq. (5) is the *resource constraint*: no edge server can host a set of VNF instances that is beyond its resource capacity. Eq. (6) specifies *bandwidth constraint*: the total amount of bandwidth allocated to two-way NS traffic between areas a_p and a_q must not exceed the in-between bandwidth capacity. The last two inequalities ensure that each federation F_ϕ has enough aggregated computation resource and bandwidth to host all SCs that are dispatched to F_ϕ, where $\alpha^{j,v}_k$ is an indicator set to 1 when $\alpha^{j,v} = a_k$ and 0 otherwise.

III. Issues and Proposed Mechanisms

We outline a straightforward approach to the SWM problem as follows. First, enumerate all possible federation structures. Second, maximize the social welfare for each federation structure. Finally, select the federation structure with the maximum social welfare. However, this approach is hardly feasible. In this section, we point out some fundamental difficulties with this approach and then propose two heuristics as our solution.

A. Finding the Maximum Profit of a Federation Structure

The SWM problem is a mixed integer programming (MIP) problem. Let us consider an associated *Federation Structure Profit Maximization* (FSPM) problem: determining the dispatch of SCs to all federations in a given federation structure $\mathcal{F}$ to maximize the total profit in $\mathcal{F}$, i.e.,

$$\hat{X} = \underset{X}{\arg\max}\, u(\mathcal{F}, X) \quad (9)$$

subject to Eqs. (4) to (8). Let $u^{\max}(\mathcal{F}) = u(\mathcal{F}, \hat{X})$ be the maximal profit in $\mathcal{F}$.

The FSPM problem is still an MIP problem. As a practical approach, we consider the following divide-and-conquer strategy. First, decompose the FSPM problem into smaller *Federation Profit Maximization* (FPM) problem: selecting a set of SCs to be hosted by a federation F_ϕ to maximize F_ϕ's own profit, i.e.,

$$\hat{X}_\phi = \underset{X_\phi}{\arg\max}\; \nu_\phi(X_\phi). \quad (10)$$

Let $\nu^{\max}_\phi = \nu_\phi(\hat{X}_\phi)$ be F_ϕ's profit with $\hat{X}_\phi$. Second, take $\sum_{F_\phi\in\mathcal{F}} \nu^{\max}_\phi$ as the maximal social welfare of $\mathcal{F}$.

Unfortunately, the solution may be invalid due to possible conflict of interest (COI) among all federations belonging to the same federation structure regarding the maximization of their own profits. The solutions to the FPM problem of two federations may both assume an exclusive hosting of a common profitable SC, but at most one of them can actually host it when they are in the same federation structure. Consequently, a federation F_ϕ may earn different profits in solving the FSPM problem for different federation structures. The actual profit of F_ϕ in solving the FSPM problem for a federation structure $\mathcal{F}$ depends on not only who the members of F_ϕ are but also how other EPSs not in F_ϕ federate together (i.e., what $\mathcal{F} \setminus \{F_\phi\}$ is).

We relax the problem by disregarding COI during the search of the optimal solution. This is done by replacing Eq. (4) with

$$x_\phi^j \le 1,\ \forall s_j \in S. \tag{11}$$

This relaxation allows an independent derivation of $\nu_\phi^{\max}$ regardless how $P \setminus F_\phi$ is further partitioned. Moreover, we may take $\nu_\phi^{\max}$ as the profit of F_ϕ in any federation structure to which it belongs and use $\sum_{F_\phi \in \mathcal{F}} \nu_\phi^{\max}$ to approach $u^{\max}(\mathcal{F})$ for any federation structure $\mathcal{F}$. Of course, this relaxation allows an SC to be served by more than one federations, so the result is only an approximation.

B. *Enumerating All Possible Federation Structures*

The number of federation structures to enumerate is numerous. In fact, the number with n ESPs is given by *Bell numbers*, B_n, where

$$B_n = \sum_{k=0}^{n} S(N,k) = \sum_{k=0}^{n} \frac{1}{k!} \sum_{i=0}^{k} (-1)^i \binom{k}{i} (k-i)^n. \tag{12}$$

An upper bound of the Bell numbers is [21]

$$B_n \le \left(\frac{0.792n}{\ln(n+1)} \right)^n, \tag{13}$$

which is $O(n^n)$.

Although the relaxation eases deriving the social welfare of each federation structure, we still have too many federation structures to enumerate. For example, we need enumerate $115,975$ federation structures for 10 ESPs. In fact, finding an optimal federation structure among all is still NP-complete [22]. To this end, we devise two heuristics combined with the relaxation as a time-efficient approach. A common feature of these two heuristics is that we do not examine every possible federation structure.

C. *Proposed Mechanisms*

We now present two heuristics for the SWM problem. One approach (Algorithm 1) constructs a federation structure by including federations one by one based on a ranking on federations. The other (Algorithm 2) uses simulated annealing to efficiently explore the solution space of the SWM problem.

Instead of enumerating $O(n^n)$ federation structures, Algorithm 1 computes $O(2^n)$ $\nu_\phi^{\max}$ values for all $F_\phi \subseteq P$. Initially, $\mathcal{C}$ is the power set of P, $\mathbf{v}$ is an empty list, and $\mathcal{F}$ is an empty set. The rank of a federation F_ϕ is determined by $\nu_\phi^{\max}$ divided by F_ϕ's weighted size (Line 6). Note that other weighting such as $\sqrt{\epsilon |F_\phi|}$ is also possible. The algorithm picks up a federation F_i that currently ranks the highest and adds it into $\mathcal{F}$. To ensure that all federations in $\mathcal{F}$ are disjoint, any federation in $\mathcal{C}$ that contains some ESP in common with F_i is excluded from further consideration (Line 11). The process repeats until $\mathcal{C}$ becomes empty.

We use a parameter $\epsilon \ge 1$ in Algorithm 1 to control the weight between profit and federation size when ranking federations. A large ϵ weights profits more than federation sizes while a small ϵ does the opposite. We can adjust ϵ to make the algorithm adaptive to different scenarios.

Algorithm 1 Greedy Approach

```
Require: $P$; $\epsilon$
Ensure: $\mathcal{F}$
 1: $\mathcal{C} \leftarrow$ power set of $P$ (excluding $\varnothing$)
 2: $\mathbf{v} \leftarrow emptylist$
 3: $\mathcal{F} \leftarrow \varnothing$
 4: for each federation $F_\phi \in \mathcal{C}$ do
 5:     $\nu_\phi^{\max} = \max_{X_i}\{\nu_\phi(X_i)\}$        ▷ Solve the FPM problem for $F_\phi$
 6:     $\mathbf{v}[F_\phi] \leftarrow \frac{\nu_\phi^{\max}}{\sqrt[\epsilon]{|F_\phi|}}$        ▷ Find the worth of $F_\phi$
 7: end for
 8: while $\mathcal{C} \neq \varnothing$ do
 9:     Pick $F_i$ from $\mathcal{C}$ that has the largest $\mathbf{v}[F_i]$
10:     $\mathcal{F} \leftarrow \mathcal{F} \cup \{F_i\}$
11:     $\mathcal{C} \leftarrow \mathcal{C} \setminus \{F' \mid F' \in \mathcal{C}, F' \cap F_i \neq \varnothing\}$
12: end while
13: return $\mathcal{F}$
```

Simulated annealing (SA) is a randomized algorithm designed to approximate the optimal solution in problems with a large search space. As a classic meta-heuristic algorithm, there are many variants of SA. We modify one of them and integrate it with our relaxation to approximate the optimal federation structure.

The details of the algorithm are shown in Algorithm 2. We initialize the solution, $\mathcal{F}$, to be a random federation structure. The utility associated with $\mathcal{F}$, e, is named *energy* in SA and is defined to be the sum of $\nu_\phi^{\max}$'s for which $F_\phi \in \mathcal{F}$. As long as the current *temperature* t_c does not drop below a preset value t_f, we repeatedly examine a random *neighbour*, $\mathcal{F}'$, of $\mathcal{F}$. The definition of a neighbour is design dependent. In our experiments presented later, $\mathcal{F}'$ is a neighbour of $\mathcal{F}$ if only one ESP has different memberships in $\mathcal{F}$ and $\mathcal{F}'$. For example, $\{\{a,b\},c\}\}$ is a neighbor of $\{\{a\},\{b\},\{c\}\}$ while $\{\{a,b\},c\}\}$ is a neighbour of $\{\{a\},\{b,c\}\}$. If the neighbor $\mathcal{F}'$ has an energy e' higher than $\mathcal{F}$'s, $\mathcal{F}'$ is taken as a new solution in this iteration. Otherwise, there is also a probability to accept $\mathcal{F}'$ as a new solution. The acceptance probability depends on the difference of the energy Δ, a parameter k, and the current temperature t. The main reason of the stochastic acceptance is to prevent the method from being stuck at a local optimum. Another parameter $c \le 1$ is used to control the number of iterations. When the termination condition is met, the algorithm returns the value of $\mathcal{F}$ as the solution. Compared with the greedy algorithm, SA need not calculate the utility of every federation beforehand, yet the number of iterations is the key to the quality of the solution.

IV. Numerical Results

We analyze the performance of the proposed heuristics against two simple solutions. One is the federation structure referred to as Singleton that contains all singleton federations (i.e., no federation at all). The other referred to as Grand is the federation structure that includes all ESPs in a single federation (i.e., the *grand federation*). We use Greedy and SA to abbreviate the federation structures found by the greedy and the SA approaches, respectively. Optimum refers to the optimal federation structure found by an exhaustive search.

Algorithm 2 Simulated Annealing (SA)

```
Require: $P$; $t_c$; $t_f$; $k$; $c$
Ensure: $\mathcal{F}$
 1: Select a random solution $\mathcal{F} \in \mathcal{F}(P)$.
 2: $e \leftarrow \sum_{F_\phi \in \mathcal{F}} \nu_\phi^{\max}$        ▷ Need solving the FPM problem for each $F_\phi$
 3: while $t_c > t_f$ do
 4:     $\mathcal{F}' \leftarrow neighbour(\mathcal{F})$
 5:     $e' \leftarrow \sum_{F_\phi \in \mathcal{F}'} \nu_\phi^{\max}$ ▷ Need solving the FPM problem for each $F_\phi$
 6:     if $e' > e$ then
 7:         $(\mathcal{F}, e) \leftarrow (\mathcal{F}', e')$
 8:     else
 9:         $\Delta \leftarrow e - e'$
10:         $r \leftarrow$ random number from $[0, 1]$
11:         if $\exp(-\Delta/(kt)) > r$ then
12:             $(\mathcal{F}, e) \leftarrow (\mathcal{F}', e')$
13:         end if
14:     end if
15:     $t_c \leftarrow c \times t_c$
16: end while
17: return $\mathcal{F}$
```

TABLE I: Simulation Setup

Parameters	Default setting
l (number of areas)	10
n (number of ESPs)	5
m (number of SCs)	50
$B^i(p,q)$, $\forall p_i \in P, (a_p, a_q) \in A^2$ (link bandwidth)	500
$R^i(k,r)$, $\forall p_i \in P, a_k \in A, 1 \le r \le t$ (resource capacity)	60
c_i^r, $\forall p_i \in P, 1 \le r \le t$ (unit cost of resource)	5
c_i^b, $\forall p_i \in P$ (unit cost of bandwidth)	5
δ_j, $\forall s_j \in S$ (SC length)	$\mathcal{N}(5, 2^2)$
$q_r^{j,v}$, $\forall s_j \in S, 1 \le v \le \delta_j, 1 \le r \le t$ (resource demand)	$\mathcal{N}(7, 2^2)$
$\beta^{j,1}$, $\forall s_j \in S$ (initial bandwidth demand)	$\mathcal{N}(50, 15^2)$
$\beta^{j,v+1} - \beta^{j,v}$, $\forall s_j \in S, 1 \le v < \delta_j$ (bandwidth increment)	±[10,20]
Payment per unit	$\mathcal{N}(15, 3^2)$
μ (maintenance cost for a two-ESP federation)	7
γ (growth rate of maintenance cost)	1.25

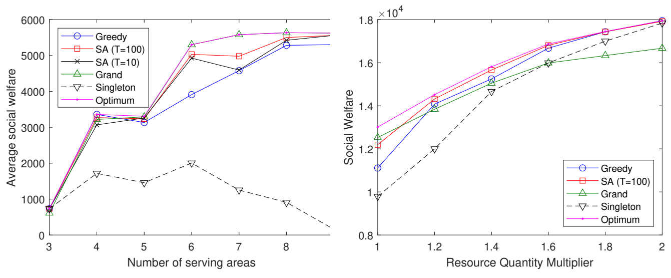
Due to huge computation cost, we tested Optimum only in small-scale experiments.

Each result of SA in all simulations is an average of 50 trials. Table I shows the major simulation parameters with default settings. We used Google OR-Tools [23] to solve all MIP problems. For a fair time comparison, all simulation programs were run on a PC with an eight-core 3.20 GHz processor and 8-GB RAM. The simulation programs were written in Python.

A. Effects of Resource Dispersion

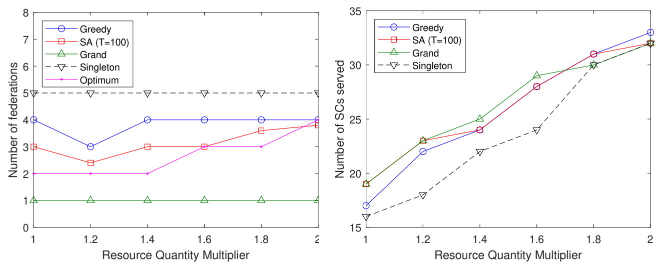
In this experiment, we assumed that all ESPs have the same amount of resources which are evenly distributed over all serving areas. We changed the number of serving areas as a way to control the degree of resource dispersion as well as the number of unserved areas.

Fig. 2a shows how the average social welfare of each approach changes with the number of serving areas. The result shows that SA has the best performance and is close to Optimum. We also note that more iterations for SA only lead to slightly better performance. The performance of Greedy is only better than Singleton. The performance of Singleton becomes worse with an increasing number of serving areas because individual federation does not have enough resource in each serving area to host SCs.



(a) vs. degree of resource dispersion (b) with uneven resource distribution

Fig. 2: Average social welfare



(a) Avg. number of federations (b) Avg. number of SCs served

Fig. 3: Results with uneven resources distribution

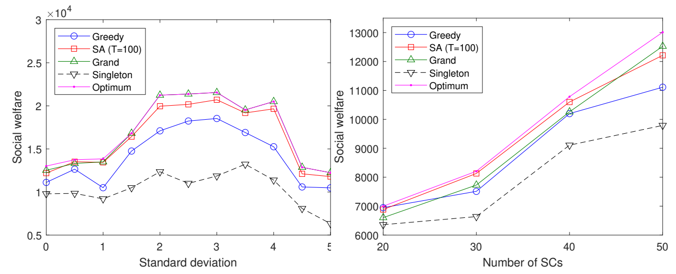
B. Effects of Uneven Resources Distribution

We also studied the effect of uneven resource distribution on performance. We used Zipf's law to distribute resource among ESPs. The resource allocated to an ESP was then evenly divided among its servers in all areas. The total amount of resource was set to 100 units initially and multiplied by a resource quantity multiplier.

As shown in Fig. 2b, social welfare generally increases as more resource is provided to the system. Greedy and SA approach Optimum when more resource is provided. Both outperform Singleton, particularly with little resource. Grand performs well when resource is not enough to serve all SCs. As resource increased, Grand is surpassed by other approaches due to its high maintenance cost.

Fig. 3a shows the sizes (cardinalities) of federation structures. Grand and Singleton surely have fixed sizes. The size of Optimum gradually changes from 2 to 4 as resource increases. Compared with Optimum, the sizes of Greedy and SA do not vary too much. Together with Fig. 2b, we can see that the size of federation structures does not have much to do with the social welfare.

Fig. 3b shows the number of SCs served by different schemes. With little resource, Grand serves more SCs than the other schemes. When the multiplier is larger than 1.6, GA and Greedy serve more SCs than Grand. Singleton serves the fewest SCs, but the gap becomes insignificant when the



(a) vs. difference of resource costs across ESPs

(b) vs. the number of SCs

Fig. 4: Average social welfare

multiplier is 1.8 or larger. Although the number of served SCs has positive correlation with social welfare, hosting more SCs does not necessarily imply higher social welfare (cf. Fig. 2b).

C. *Effects of Heterogeneous Cost Structures*

We varied the difference of resource costs across ESPs and measured the social welfares. The difference was controlled by the standard deviation of the resource cost distribution. A high standard deviation implies a high degree of cost variation.

As shown in Fig. 4a, the difference of social welfares is not significant with small standard deviation. When the standard deviation is between 2 and 4, searching good federation structures is beneficial (compared with either Grand or Singleton). When the standard deviation is even larger, the performance gap becomes smaller. The reason is that some highly profitable SCs dominates the overall profit. Once these SCs are served, the rest low-profit SCs cannot contribute the overall profit too much. In most cases, SA and Grand perform similarly and are close to Optimum.

D. *Effect of the Number of SCs*

Fig. 4b shows the relationship between social welfare and the number of SCs. With few SCs, Grand has poor performance due to its high federation maintenance cost. When the number of SCs increases, Grand outperforms Singleton and Greedy because it has enough resource and bandwidth capacity to serve all SCs. Except for 50 SCs, SA performs even better than Grand and is very close to Optimum.

V. Conclusions

In this paper, we consider maximizing the social welfare of a set of ESPs by identifying an optimal federation structure. This problem is hard due to numerous enumerations of federation structures and the infeasibility of a divide-and-conquer problem decomposition. We therefore propose two time-efficient heuristics: Greedy and SA. These two heuristics generally outperform federation structures that contain only grand or singleton federations. In particular, SA can be taken as an approximation to the optimal solution.

In the future, we shall study the profit distribution problem for federation participants. Specifically, we shall analyze payoff satisfaction level of each ESP for each possible profit distribution scheme.

References

[1] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Comm. Mag.*, vol. 53, no. 2, pp. 90–97, 2015.

[2] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, "Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges," *IEEE Comm. Mag.*, vol. 55, no. 5, pp. 80–87, 2017.

[3] D. Li, P. Hong, K. Xue, and j. Pei, "Virtual network function placement considering resource optimization and SFC requests in cloud datacenter," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 7, pp. 1664–1677, 2018.

[4] H. Guo, Y. Wang, Z. Li, X. Qiu, H. An, P. Yu, and N. Yuan, "Cost-aware placement and chaining of service function chain with vnf instance sharing," in *Proc. IEEE/IFIP NOMS*, Budapest, Hungary, Apr. 2020.

[5] I. Benkacem, T. Taleb, M. Bagaa, and H. Flinck, "Performance benchmark of transcoding as a virtual network function in CDN as a service slicing," in *Proc. IEEE WCNC*, Barcelona, Spain, Apr. 2018.

[6] C. Pham, N. H. Tran, S. Ren, W. Saad, and C. S. Hong, "Traffic-aware and energy-efficient vNF placement for service chaining: Joint sampling and matching approach," *IEEE Trans. Services Comput.*, vol. 13, no. 1, pp. 172–185, 2020.

[7] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *IEEE 3rd Int'l Conf. on Cloud Networking*, Luxembourg, Luxembourg, Oct. 2014, pp. 7–13.

[8] D. Dietrich, A. Abujoda, A. Rizk, and P. Papadimitriou, "Multi-provider service chain embedding with nestor," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 1, pp. 91–105, 2017.

[9] D. Dietrich, A. Abujoda, and P. Papadimitriou, "Network service embedding across multiple providers with Nestor," in *IFIP Netw. Conf.*, Toulouse, France, May 2015.

[10] F. Carpio, S. Dhahri, and A. Jukan, "VNF placement with replication for load balancing in NFV networks," in *Proc. IEEE ICC*, Paris, France, May 2017.

[11] F. Ben Jemaa, G. Pujolle, and M. Pariente, "QoS-aware VNF placement optimization in edge-central carrier cloud architecture," in *Proc. IEEE Globecom*, Dec. 2016.

[12] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal vNF placement at the network edge," in *Proc. IEEE INFOCOM*, 2018, pp. 693–701.

[13] Y. Jia, C. Wu, , Z. Li, F. Le, and A. Liu, "Online scaling of NFV service chains across geo-distributed datacenters," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 699–710, Apr. 2018.

[14] Z. Zhou, Q. Wu, and X. Chen, "Online orchestration of cross-edge service function chaining for cost-efficient edge computing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1866–1880, 2019.

[15] A. Leivadeas, G. Kesidis, M. Ibnkahla, and I. Lambadaris, "VNF placement optimization at the edge and cloud," *Future Internet*, vol. 11, no. 3, Jan. 2019.

[16] I. Benkacem, T. Taleb, M. Bagaa, and H. Flinck, "Optimal VNFs placement in CDN slicing over multi-cloud environment," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 616–627, Mar. 2018.

[17] Y.-C. Chen and L.-H. Yen, "Distributed profitable deployment of network services to geo-distributed edge systems," in *Proc. APNOMS*, Daegu, Korea, Sep. 2020.

[18] L.-H. Yen, C.-H. Chang, and Y.-C. Chen, "Profit maximization by forming federations of geo-distributed MEC platforms," in *Proc. IEEE WCNC Workshop*, Marrakech, Morocco, Apr. 2019.

[19] V. Sciancalepore, F. Giust, K. Samdanis, and Z. Yousaf, "A double-tier MEC-NFV architecture: Design and optimisation," in *Proc. IEEE CSCN*, Berlin, Germany, 2016.

[20] "TOSCA simple profile for network functions virtualization (NFV) version 1.0," http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/csd04/tosca-nfv-v1.0-csd04.html, accessed: 2020-10-30.

[21] D. Berend and T. Tassa, "Efficient bounds on Bell numbers and on moments of sums of random variables," *Probability and Mathematical Statistics*, vol. 30, no. 2, Jan. 2010.

[22] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohmé, "Coalition structure generation with worst case guarantees," *Artificial Intelligence*, vol. 111, no. 1, pp. 209–238, 1999.

[23] L. Perron and V. Furnon, "OR-Tools," Google. [Online]. Available: https://developers.google.com/optimization/