SOFTWARE DEFINED RESOURCE ALLOCATION FOR SERVICE-ORIENTED NETWORKS

Nan Zhang[⋆]*, Ya-Feng Liu*[†]*, Hamid Farmanbar*[§]*, Tsung-Hui Chang*[‡]*, Mingyi Hong*[♯]*, and Zhi-Quan Luo*[‡]

[⋆]School of Mathematical Sciences, Peking University, Beijing, China
[†]LSEC, ICMSEC, AMSS, Chinese Academy of Sciences, Beijing, China
[§]Huawei Canada Research Center, Ottawa, Canada
[‡]Shenzhen Research Institute of Big Data, The Chinese University of Hong Kong, Shenzhen, China
[♯]Department of Electrical and Computer Engineering, University of Minnesota, MN, USA

Email: zhangnan625@pku.edu.cn, yafliu@lsec.cc.ac.cn, Hamid.Farmanbar@huawei.com,
{changtsunghui, luozq}@cuhk.edu.cn, mhong@umn.edu

ABSTRACT

To support multiple on-demand services over several fixed communication networks, the network operators must allow flexible customization and fast provision of their network resources. One effective approach is network virtualization, whereby each service is mapped to a virtual subnetwork providing dedicated on-demand support. In practice, each service consists of a prespecified sequence of functions, called a service function chain (SFC). Moreover, each function in a SFC can only be provided by some given network nodes. Thus, to support a given service, we must select network function nodes according to the SFC, and determine the routing strategy through the function nodes in the specified order. A crucial problem that needs to be addressed is how to optimally allocate the network resources while satisfying multiple service requirements specified by the service function chains, subject to link and node capacity constraints. In this paper, we formulate the problem as a mixed binary linear program and establish its NP-hardness. Furthermore, we propose an efficient penalty successive upper bound minimization algorithm to solve the problem. We also present simulation results to demonstrate the effectiveness of the proposed algorithm.

***Index Terms*—** Software Defined Network, Network Function Virtualization, Resource Allocation.

1. INTRODUCTION

Today's communication networks are increasingly required to support multiple services with diverse characteristics and requirements. Network function virtualization (NFV) [1] is an important technology that enables the service providers intelligently integrate a variety of network resources owned by different operators to establish a service customized virtual network (VN) for each service request. In practice, each service consists of a sequence of service functions that can only be provided by certain specific nodes, called NFV-enabled nodes. As all of the VNs share a common resource pool, it is crucial to allocate the network resources economically while meeting the diverse service requirements, and satisfying the capacity constraints at NFV-enabled nodes and over network links.

Recently, reference [2] proposed a novel 5G wireless network architecture MyNET and an enabling technique called SONAC (Service-Oriented Virtual Network Auto-Creation). In SONAC, there are two key components: software defined topology (SDT), and software defined resource allocation (SDRA). SDT determines the VN graph and the VN logical topology for each service. The determination of the VN logical topology, also called VN graph embedding, maps the service functions onto the physical NFV-enabled infrastructures, so that each function on the corresponding SFC is realized. SDRA maps the logical topology to physical network resources, including both communication and computational resources.

In software defined networks, centralized traffic control enables joint VN gragh embedding and resource allocation. Specifically, it controls the flow routing such that each flow gets processed at some NFV-enabled nodes in the order of the service functions defined in the corresponding service function chain (SFC). In recent years, there are some works on the related problem of joint node selection and routing [3, 4, 5, 6, 7]. References [3, 4] simplified "routing" by only considering one-hop routing or selecting paths from a predetermined path set. Reference [5] considered the so-called consolidated middleboxes where a flow could receive all the required functions. It also proposed a two-stage heuristic algorithm to route each flow through its single associated function node. Such formulation is not applicable to the case where multiple functions are provided in sequence at nodes. An important common assumption in [4, 6, 7] is that the instantiation of a service function for a traffic flow can be split over multiple nodes. The functional splitting assumption significantly simplifies the problem since no binary variable needs to be introduced in the problem formulation. However, this might result in high coordination overhead in practice, especially when the number of service requests is large.

In this paper, we consider joint VN graph embedding and resource allocation, where a set of service requests in a given network

THIS WORK IS PARTIALLY SUPPORTED BY NSF GRANTS CCF-1526434 AND CCF-1526078 AND PARTIALLY SUPPORTED BY NSFC GRANTS 61571384, 61571385, 11671419 AND 11631013.

are simultaneously routed and processed. Our considered problem differs from the previously mentioned works in that we require each service function of a given SFC to be provided at exactly one NFV-enabled node (or function node). We formulate the problem as a novel mixed binary linear program. We show that checking the feasibility of this problem is NP-hard in general. Moreover, we propose an efficient iterative algorithm to solve the problem. The proposed algorithm solves a linear program (LP) at each iteration. Our simulation results show that the proposed algorithm can find a high-quality (near-optimal) solution of the problem.

2. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we introduce the model and present a formulation of the joint VN graph embedding and resource allocation problem.

We consider a communication network represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, where $\mathcal{V} = \{v_i\}$ is the set of nodes and $\mathcal{L} = \{l_{ij}\}$ is the set of directed links. Denote the subset of nodes that can provide the service function f as V_f. Each function node i has a known computational capacity μ_i, and we assume that processing one unit of data flow requires one unit of computational capacity. Suppose that there are K data flows, each requesting a distinct service in the network. The requirement of each service is given by a service function chain $\mathcal{F}(k)$, consisting of a predefined set of functions that have to be performed in sequence by the network. Different from previously mentioned works [4, 6, 7], we require that each service function in $\mathcal{F}(k)$ must be instantiated at exactly one NFV-enabled node, i.e., all data packets of flow k should be directed to the same function node to get processed. Notice that it is possible that a common function requested by different SFCs is provided by different nodes. The source-destination pair of flow k is given as $(S(k), D(k))$, and arrival data rate of flow k is given as $\lambda(k)$. The joint VN graph embedding and resource allocation problem is to determine the routes and the rates of all flows on the routes while satisfying the SFC requirements and the capacity constraints of all links and nodes.

Let $r_{ij}(k)$ be the rate of flow k on link (i, j). The capacity of link (i, j) is assumed to be known as C_{ij} which is a fixed constant. This assumption is reasonable when the channel condition is stable during the considered period of time. The total flow rates on link (i, j) is then upper bounded by capacity C_{ij}, i.e.,

$$\sum_k r_{ij}(k) \le C_{ij},\ \forall\,(i,j) \in \mathcal{L}. \tag{2.1}$$

To describe the SFC requirement, we introduce binary variables for the function nodes. Let $x_{i,f}(k)$ be the binary variable indicating whether or not node i provides function f for flow k (i.e., $x_{i,f}(k) = 1$ if node i provides function f for flow k; otherwise $x_{i,f}(k) = 0$). To ensure each flow k is served by exactly one node for each $f \in \mathcal{F}(k)$, we have the following constraint

$$\sum_{i \in V_f} x_{i,f}(k) = 1,\ \forall\, f \in \mathcal{F}(k),\ \forall\, k. \tag{2.2}$$

Suppose that the function chain of flow k is $\mathcal{F}(k) = (f_1^k \to f_2^k \to \cdots \to f_n^k)$. To ensure flow k goes into the function nodes in the exact order of the functions in $\mathcal{F}(k)$, we introduce new virtual flows labelled (k, f): flow k just after receiving the service function f is labelled (k, f). We let (k, f_0^k) denote flow k just coming out of the source node $S(k)$ without receiving any service function. See Fig. 1 for an illustration. Let $r_{ij}(k, f)$ be the rate of flow (k, f) over link (i, j). Then the following flow conservation constraints hold for all nodes i and $s = 1, \ldots, n$:

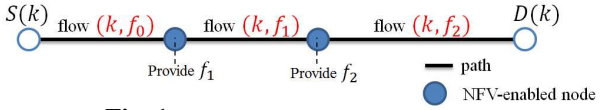

Fig. 1. An illustration of the virtual flow.

$$\lambda(k) x_{i,f_s^k}(k) = \sum_{j:(j,i)\in\mathcal{L}} r_{ji}(k, f_{s-1}^k) - \sum_{j:(i,j)\in\mathcal{L}} r_{ij}(k, f_{s-1}^k), \tag{2.3}$$

$$\lambda(k) x_{i,f_s^k}(k) = \sum_{j:(i,j)\in\mathcal{L}} r_{ij}(k, f_s^k) - \sum_{j:(j,i)\in\mathcal{L}} r_{ji}(k, f_s^k), \tag{2.4}$$

$$\sum_{j:(S(k),j)\in\mathcal{L}} r_{S(k)j}(k, f_0^k) = \lambda(k), \tag{2.5}$$

$$\sum_{j:(j,D(k))\in\mathcal{L}} r_{jD(k)}(k, f_n^k) = \lambda(k), \tag{2.6}$$

where (2.3) and (2.4) imply that if $x_{i,f_s^k} = 1$, then flow (k, f_{s-1}^k) going into node i and flow (k, f_s^k) coming out of node i both have rate $\lambda(k)$; otherwise each virtual flow (k, f_s^k) coming out of node i and going into node i are with the same rate; (2.5) and (2.6) ensure that flow k coming out of $S(k)$ and going into $D(k)$ both have rate $\lambda(k)$. These constraints guarantee that each flow k goes into the nodes which provide functions in $\mathcal{F}(k)$ sequentially and with the required data rate $\lambda(k)$.

By the definitions of $r_{ij}(k, f)$ and $r_{ij}(k)$, we have

$$r_{ij}(k) = \sum_{f\in\mathcal{F}(k)} r_{ij}(k, f),\ \forall\, k,\ \forall\,(i,j) \in \mathcal{L}. \tag{2.7}$$

We also assume that each function node provides at most one function for each flow:

$$\sum_{f\in\mathcal{F}(k)} x_{i,f}(k) \le 1,\ \forall\, k,\ \forall\, i. \tag{2.8}$$

This assumption is without loss of generality. This is because, if a NFV-enabled node can provide multiple services for a flow, we can introduce virtual nodes such that each virtual node provides one function for the flow and all these virtual nodes are connected with each other.

Since processing one unit of data flow consumes one unit of computational capacity, the node capacity constraint can be expressed as

$$\sum_f \sum_k \lambda(k) x_{i,f}(k) \le \mu_i,\ \forall\, i. \tag{2.9}$$

Now we present our joint VN graph embedding and resource allocation formulation to minimize the total link rates $g(\mathbf{r}) := \sum_{k,(i,j)} r_{ij}(k)$, which can avoid cycles in choosing routing paths:

$$\begin{aligned} \min_{\mathbf{r},\mathbf{x}}\quad & g(\mathbf{r}) := \sum_{k,(i,j)} r_{ij}(k) \\ \text{s.t.}\quad & (2.1) - (2.9), \\ & r_{ij}(k) \ge 0,\ \forall\, k,\ \forall\,(i,j) \in \mathcal{L}, \\ & r_{ij}(k, f) \ge 0,\ \forall\, f \in \mathcal{F}(k),\ \forall\, k,\ \forall\,(i,j) \in \mathcal{L}, \\ & x_{i,f}(k) \in \{0, 1\},\ \forall\, i \in V_f,\ \forall\, f \in \mathcal{F}(k),\ \forall\, k, \end{aligned} \tag{2.10}$$

where $\mathbf{r} := \big(\{r_{ij}\}, \{r_{ij}(k)\}\big)$, $\mathbf{x} := \{x_{i,f}(k)\}$. The objective can also be others, such as the cost of the consumed computational resources and the number of activated function nodes.

Problem (2.10) is a mixed binary linear program which turns out to be NP-hard. The proof is based on a polynomial time reduction from the 3-dimensional matching problem [8]. We omit the proof due to space limitation.

Theorem 2.1 *Checking the feasibility of problem* (2.10) *is NP-complete, and thus solving problem* (2.10) *itself is NP-hard.*

3. THE PROPOSED PSUM ALGORITHM

Since problem (2.10) is NP-hard, it is computationally expensive to solve it to global optimality. In this section, we propose an efficient penalty successive upper bound minimization (PSUM) algorithm to solve it approximately. The basic idea of our proposed algorithm is to relax the binary variables in problem (2.10) to real ones and add penalty terms to induce binary solutions.

Notice that problem (2.10) becomes an LP when we relax the variables $\{x_{i,f}(k)\}$ to be continuous. Problem (2.10) and its relaxation problem are generally not equivalent (in the sense that the solution $\mathbf{x}$ of the relaxation problem is not binary). The following Theorem 3.1 provides some conditions under which the two problems are equivalent.

Theorem 3.1 *Suppose $\mu_i \geq \bar{\mu}$ for all i, and $C_{ij} \geq \bar{C}$ for all (i,j), where*

$$\bar{\mu} = \sum_k \lambda(k),\ \bar{C} = \sum_k \lambda(k)(|\mathcal{F}(k)|+1),$$

and $|\mathcal{F}(k)|$ denotes the number of functions in $\mathcal{F}(k)$. Then the relaxation problem of (2.10) *always has a binary solution of $\{x_{i,f}(k)\}$. Moreover, the lower bounds in the above are tight in the sense that there exists an instance of problem* (2.10) *such that its relaxation problem does not have a binary solution of $\{x_{i,f}(k)\}$ if one of the lower bounds is violated.*

Theorem 3.1 suggests that, if the link and node capacity are sufficiently large, then problem (2.10) and its relaxation problem are equivalent.

To solve the general problem (2.10), our basic idea is to add an L_p penalty term in the objective of the relaxation problem of (2.10) to promote the binary solutions.

Let $\mathbf{x}_f(k) = \{x_{i,f}(k)\}_{i\in V_f}$. Then, we can rewrite (2.2) as

$$\|\mathbf{x}_f(k)\|_1 = 1,\ \forall\, f \in \mathcal{F}(k),\ \forall\, k. \tag{3.1}$$

We have the following fact [9].

Fact: For any k and any $f \in \mathcal{F}(k)$, consider

$$\begin{array}{ll} \min\limits_{\mathbf{x}_f(k)} & \|\mathbf{x}_f(k)+\epsilon\mathbf{1}\|_p^p := \sum\limits_{i\in V_f}(x_{i,f}(k)+\epsilon)^p \\ \text{s.t.} & \|\mathbf{x}_f(k)\|_1 = 1, \\ & x_{i,f}(k)\in[0,1],\ \forall\, i\in V_f, \end{array} \tag{3.2}$$

where $p \in (0,1)$ and ϵ is any positive constant. The optimal solution of problem (3.2) is binary, that is, only one element is one and all the others are zero, and its optimal value is $c_{\epsilon,f} := (1+\epsilon)^p + (|V_f|-1)\epsilon^p$. Moreover, the objective function of problem (3.2) is differentiable with respect to each element $x_{i,f}(k) \in [0,1]$.

Motivated by the above fact, we propose to solve the following penalized problem

$$\begin{array}{ll} \min\limits_{\mathbf{r},\mathbf{x}} & g(\mathbf{r}) + \sigma P_\epsilon(\mathbf{x}) \\ \text{s.t.} & (2.1)-(2.9), \\ & r_{ij}(k) \geq 0,\ \forall\, k,\ \forall\,(i,j)\in\mathcal{L}, \\ & r_{ij}(k,f) \geq 0,\ \forall\, f\in\mathcal{F}(k),\ \forall\, k,\ \forall\,(i,j)\in\mathcal{L}, \\ & x_{i,f}(k)\in[0,1],\ \forall\, i\in V_f,\ \forall\, f\in\mathcal{F}(k),\ \forall\, k, \end{array} \tag{3.3}$$

where σ is the penalty parameter, and

$$P_\epsilon(\mathbf{x}) = \sum_k \sum_{f\in\mathcal{F}(k)} \big(\|\mathbf{x}_f(k)+\epsilon\mathbf{1}\|_p^p - c_{\epsilon,f}\big). \tag{3.4}$$

For ease of presentation, we define $\mathbf{z} = (\mathbf{r},\mathbf{x})$, $g_\sigma(\mathbf{z}) = g(\mathbf{r}) + \sigma P_\epsilon(\mathbf{x})$, $P^t = P_\epsilon(\mathbf{x}^t)$, and $g^t = g(\mathbf{r}^t)$. Theorem 3.2 reveals the relationship between the solutions of problems (2.10) and (3.3).

Theorem 3.2 *For any fixed $\epsilon > 0$, let $\mathbf{z}^t$ be a global minimizer of problem* (3.3) *with the objective function $g_{\sigma_t}(\mathbf{z})$. Suppose the positive sequence $\{\sigma_t\}$ is monotonically increasing and $\sigma_t \to +\infty$. Then any limit point of $\{\mathbf{z}^t\}$ is a global minimizer of* (2.10)*.*

Proof. Since $\mathbf{z}^t$ is a global minimizer of (3.3) with the objective function $g_{\sigma_t}(\mathbf{z})$, it follows that

$$g_{\sigma_t}(\mathbf{z}^t) \leq g_{\sigma_t}(\mathbf{z}^{t+1}),\ g_{\sigma_{t+1}}(\mathbf{z}^{t+1}) \leq g_{\sigma_{t+1}}(\mathbf{z}^t),\ \forall\, t.$$

Combining the above with the assumption $\sigma_t \leq \sigma_{t+1}$, we obtain

$$\sigma_t(P^t - P^{t+1}) \leq g^{t+1} - g^t \leq \sigma_{t+1}(P^t - P^{t+1}),\ \forall\, t,$$

which shows that $\{g^t\}$ is increasing and $\{P^t\}$ is decreasing.

Suppose that $\mathbf{z}^*$ is a global minimizer of problem (2.10). Then $P_\epsilon(\mathbf{z}^*) = 0$. By the definition of $\mathbf{z}^t$, we have $g_{\sigma_t}(\mathbf{z}^t) \leq g_{\sigma_t}(\mathbf{z}^*) = g(\mathbf{z}^*)$, which further implies that

$$0 \leq g^t + \sigma_t P^t \leq g(\mathbf{z}^*). \tag{3.5}$$

This, together with the facts that $g^t \geq 0$, $P^t \geq 0$, and $\sigma_t \to +\infty$, shows that $\sigma_t P^t \to 0$ and $P^t \to 0$ as $t \to +\infty$.

Let $\bar{\mathbf{z}} = (\bar{\mathbf{r}},\bar{\mathbf{x}})$ be any limit point of $\{\mathbf{z}^t\}$, and $\{\mathbf{z}^t\}_\mathcal{T}$ be a subsequence converging to $\bar{\mathbf{z}}$. Since $P^t \to 0$, we have $P_\epsilon(\bar{\mathbf{x}}) = 0$, which shows that $\bar{\mathbf{z}}$ is feasible for problem (2.10). Furthermore, taking limit along $\mathcal{T}$ in (3.5), we have $g(\bar{\mathbf{z}}) \leq g(\mathbf{z}^*)$. Therefore, $\bar{\mathbf{z}}$ is a global minimizer of problem (2.10). □

Theorem 3.2 suggests that the penalty parameter σ should go to infinity to guarantee that the solution $\mathbf{x}$ of problem (3.3) is binary. In practice, however, the parameter σ only needs to be large enough so that the values of $\{x_{i,f}(k)\}$ are either close to zero or one. Then, a (feasible) binary solution of (2.10) can be obtained by rounding.

Solving problem (3.3) directly is not easy since it is a linearly constrained nonlinear program. To efficiently solve problem (3.3), we apply the SUM (Successive Upper bound Minimization) method [10, 11], which solves a sequence of approximate objective functions which are lower bounded by the original objective function. Due to the concavity of $P_\epsilon(\mathbf{x})$, the first order approximation of $P_\epsilon(\mathbf{x})$ is an upper bound of itself, i.e., $P_\epsilon(\mathbf{x}) \leq P_\epsilon(\mathbf{x}^t) + \nabla P_\epsilon(\mathbf{x}^t)^T(\mathbf{x}-\mathbf{x}^t)$, where $\mathbf{x}^t$ is the current iterate. At the $(t+1)$-st iteration, we solve the following problem

$$\begin{aligned} \min_{\mathbf{r},\mathbf{x}} \quad & g(\mathbf{r}) + \sigma_{t+1}\nabla P_\epsilon(\mathbf{x}^t)^T\mathbf{x} \\ \text{s.t.} \quad & (2.1)-(2.9), \\ & r_{ij}(k) \geq 0,\ r_{ij}(k,f) \geq 0,\ \forall\, f,\ \forall\, k,\ \forall\, (i,j) \in \mathcal{L}, \\ & x_{i,f}(k) \in [0,1],\ \forall\, i \in V_f,\ \forall\, f \in \mathcal{F}(k),\ \forall\, k. \end{aligned} \tag{3.6}$$

Notice that each subproblem (3.6) is an LP which can be efficiently solved to global optimality. The proposed PSUM algorithm for solving problem (2.10) is described in Algorithm 1, where γ and η are two predefined constants satisfying $0 < \eta < 1 < \gamma$. We remark that (1) the parameter ϵ is adaptively updated as the iteration number increases, which turns out to be very helpful in improving numerical performance of the algorithm [12]; (2) reference [13] proposed a Penalty-BSUM algorithm which relaxes some equality constraints and applies Block-SUM to solve the penalized problem, while our PSUM algorithm enforces the relaxed variables being binary.

Algorithm 1 PSUM Algorithm for Solving Problem (2.10)

0. Solve (3.3) with $\sigma = 0$, and obtain solution $\mathbf{z}^0 = (\mathbf{r}^0, \mathbf{x}^0)$;
1. Initialize $\epsilon_1, \sigma_1, t_{max}$, and let $t = 0$;
2. **While** $t < t_{max}$ **Do**

Solve problem (3.6) with $\sigma = \sigma_{t+1}$ and $\epsilon = \epsilon_{t+1}$, and obtain solution $\mathbf{z}^{t+1} = (\mathbf{r}^{t+1}, \mathbf{x}^{t+1})$;

If $\mathbf{x}^{t+1}$ is binary, stop; otherwise set $t = t+1$, and let $\sigma_{t+1} = \gamma\sigma_t,\ \epsilon_{t+1} = \eta\epsilon_t$;

End

4. NUMERICAL EXPERIMENTS

In this section, we present some numerical results to illustrate the effectiveness of the proposed algorithm. More simulation results can be found in [15]. We shall compare the PSUM algorithm with the heuristic algorithm proposed in [6]. To solve problem (2.10), the algorithm is slightly modified, as shown in Algorithm 2. In Algorithm 2, we denote the set of binary variables $\{x_{i,f}(k)\}$ as $\mathcal{S}$, the set of $\{x_{i,f}(k)\}$ taking value of 1 as $\mathcal{S}_1$, and those taking value of 0 as $\mathcal{S}_0$. All LP subproblems are solved by the solver Gurobi 7.0.1 [14].

We consider a mesh network with 100 nodes and 684 directed links. There are in total 5 service functions $\{f_1, \ldots, f_5\}$ and $|V_f| = 10$ candidate nodes for each function. We consider $K = 30$ flows with demands $\lambda(k) = 1$ for all k. The SFC $\mathcal{F}(k) = (f_1^k \rightarrow f_2^k)$ and $(S(k), D(k))$ are uniformly randomly chosen for each flow $(f_1^k \neq f_2^k, S(k), D(k) \notin V_{f_s^k}, s = 1, 2)$. The link capacity C_{ij} is uniformly randomly chosen in $[0.5, 5.5]$, and the node capacity μ_i is uniformly randomly chosen in $[0.5, 8]$. For the PSUM algorithm, we set $t_{max} = 20,\ \sigma_1 = 2,\ \epsilon_1 = 0.001,\ \gamma = 1.1,\ \eta = 0.5$. For the heuristic algorithm, we set $\theta_1 = 0.1,\ \theta_2 = 0.9,\ T_{max} = 19$.

We randomly generate 50 instances of problem (2.10) and apply the two algorithms to solve them. Simulation results show that the PSUM algorithm successfully finds the feasible binary solution 48 times while the heuristic algorithm only succeeds 9 times. The left figure in Fig. 2 shows the averaged number of fractional components of the solutions returned by the two algorithms as the number of iterations increases. We can easily observe that the solutions returned by the heuristic algorithm quickly get stuck (at some uninteresting points after about 6 iterations), while the solutions returned by our proposed algorithm gradually converge to some feasible binary solutions. To measure the optimality of the obtained solution, we compute the ratio of the objective value of problem (2.10) returned by PSUM (g^*_{PSUM}) and the optimal value of the relaxation LP (g^*_{LP}). Notice that the optimal value of the relaxation LP is a lower bound of the optimal value of our interested problem (2.10). The right figure in Fig. 2 shows the number of simulations with ratios at or below ξ, where $\xi \in [1, 1.09]$. The ratios in all 50 simulations are below 1.09. This shows that the returned solutions by the PSUM algorithm are of good quality, which are close to global optimality. Notice that the complexity of both the two algorithms can be measured by the number of solved LP subproblems, and the sizes of the LPs in the two algorithms are close. We can see that the PSUM algorithm gives much better solutions with fewer LP subproblems being solved.

Algorithm 2 Modified Heuristic Algorithm [6]

For $t = 1 : T_{max}$

Solve (2.10) with relaxed binary variables and with $x_{i,f}(k) \in \mathcal{S}_1$ being fixed to be one. Let the solution be $\{x^*_{i,f}(k)\}$ and set

$\mathcal{S}_1' = \{(i,f,k) \mid x^*_{i,f}(k) \geq \theta_2\}$,

$\mathcal{S}_0' = \{(i,f,k) \mid x^*_{i,f}(k) \leq \theta_1\}$;

Update $\mathcal{S}_1'$ by checking constraints (2.9), i.e.,

let $x_{i,f}(k) = 1$ for all $(i,f,k) \in \mathcal{S}_1'$, check whether $\sum_f \sum_{k:(i,f,k)\in\mathcal{S}_1'} \lambda(k) x_{i,f}(k) \leq \mu_i$ holds for all i, remove those from $\mathcal{S}_1'$ that participate in the violated inequalities;

Let $\mathcal{S}_1 = \mathcal{S}_1',\ \mathcal{S}_0 = \mathcal{S}_0'$, and $\mathcal{S}' = \mathcal{S} \setminus (\mathcal{S}_1 \cup \mathcal{S}_0)$.

End

For each $(i,f,k) \in \mathcal{S}'$, solve the relaxation problem with $x_{i,f}(k) = 0$; if the problem is infeasible, add the index into $\mathcal{S}_1$; For $x_{i,f}(k)$ whose being assigned to zero leads to the maximum decrease or least increase in the objective, add the index into $\mathcal{S}_0$; Re-solve the problem with variables in $\mathcal{S}_1$ and $\mathcal{S}_0$ being fixed.

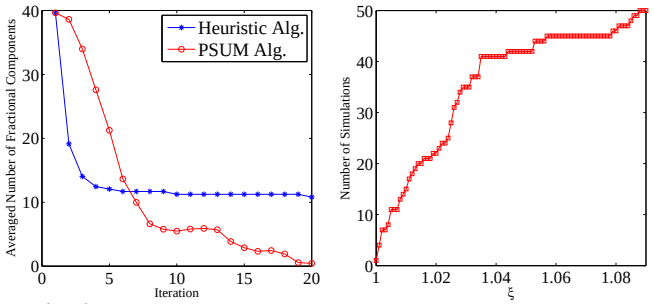

Fig. 2. Left: the average number of fractional components of the solutions returned by the two algorithms versus the number of iterations; Right: the number of simulations where the ratio $g^*_{\text{PSUM}}/g^*_{\text{LP}}$ is less than or equal to $\xi \in [1, 1.09]$.

In summary, our simulation results demonstrate that the proposed PSUM algorithm can approximately solve problem (2.10) by returning a feasible (binary) solution with good quality. In addition to its good numerical performance, the PSUM algorithm is easily implemented and converges very fast, i.e., it usually takes no more than 20 iterations to terminate.

5. REFERENCES

[1] M. Chiosi. Network Functions Virtualization: Introductory White Paper. *SDN and OpenFlow World Congress*, Darmstadt, Germany, Oct. 2012.

[2] H. Zhang, S. Vrzic, G. Senarath, N. D. Dào, H. Farmanbar, J. Rao, C. Peng, and H. Zhuang. 5G Wireless Network: MyNET and SONAC. *IEEE Network*, vol. 29, no. 4, pp. 14–23, Jul. 2015.

[3] J. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint VM Placement and Routing for Data Center Traffic Engineering," in *Proc. IEEE INFOCOM*, pp. 2876–2880, Mar. 2012.

[4] S. Narayana, W. Jiang, J. Rexford, and M. Chiang, "Joint Server Selection and Routing for Geo-Replicated Services," in *Proc. IEEE/ACM 6th International Conference on Utility and Cloud Computing*, pp. 423–428, 2013.

[5] A. Gushchin, A. Walid, and A. Tang, "Scalable Routing in SDN-enabled Networks with Consolidated Middleboxes," in *Proc. ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, pp. 55–60, 2015.

[6] X. Li, J. Rao, and H. Zhang. Engineering Machine-to-Machine Traffic in 5G. *IEEE Internet of Things Journal*, vol. 3, no. 4, pp. 609–618, Aug. 2016.

[7] M. Charikar, Y. Naamad, J. Rexford, and X. K. Zou. Multi-commodity Flow with In-network Processing. *Manuscript*, `www.cs.princeton.edu/~jrex/papers/mopt14.pdf`.

[8] R. M. Karp. Reducibility among Combinatorial Problems. *Complexity of computer computations*, springer US, pp. 85–103, 1972.

[9] P. Liu, Y.-F. Liu, and J. Li, "An Iterative Reweighted Minimization Framework for Joint Channel and Power Allocation in the OFDMA System," in *Proc. IEEE ICASSP*, Apr. 2015, pp. 3068–3072.

[10] D. R. Hunter and K. Lange. A Tutorial on MM Algorithms. *The American Statistician*, vol. 58, no. 1, pp. 30–37, 2004.

[11] M. Razaviyayn, M. Hong, and Z.-Q. Luo. A Unified Convergence Analysis of Block Successive Minimization Methods for Nonsmooth Optimization. *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 1126–1153, 2013.

[12] B. Jiang, Y.-F. Liu, and Z. Wen. ℓ_q-Norm Regularization Algorithms for Optimization over Permutation Matrices. *SIAM Journal on Optimization*, vol. 26, no. 4, pp. 2284–2313, Jan. 2016.

[13] Q. Shi, M. Hong, X. Gao, E. Song, Y. Cai, and W. Xu. Joint Source-Relay Design for Full–Duplex MIMO AF Relay Systems. *IEEE Transactions on Signal Processing*, vol. 64, no. 23, pp. 6118–6131, Dec. 2016.

[14] Gurobi Optimization, *Gurobi Optimizer Reference Manual*. [Online]. Available: `http://www.gurobi.com`.

[15] N. Zhang, Y.-F. Liu, H. Farmanbar, T.-H. Chang, M. Hong and Z.-Q. Luo. Network Slicing for Service-Oriented Networks Under Resource Constraints. *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2512–2521, Nov. 2017.