Approximation Algorithms for the NFV Service Distribution Problem

Hao Feng*, Jaime Llorca†, Antonia M. Tulino†, Danny Raz†, Andreas Molisch*
* University of Southern California, Email: {haofeng, molisch}@usc.edu
† Nokia Bell Labs, Email: {jaime.llorca, a.tulino, danny.raz}@nokia-bell-labs.com

***Abstract*—Distributed cloud networking builds on network functions virtualization (NFV) and software defined networking (SDN) to enable the deployment of network services in the form of elastic virtual network functions (VNFs) instantiated over general purpose servers at distributed cloud locations. We address the design of fast approximation algorithms for the NFV service distribution problem (NSDP), whose goal is to determine the placement of VNFs, the routing of service flows, and the associated allocation of cloud and network resources that satisfy client demands with minimum cost. We show that in the case of load-proportional costs, the resulting fractional NSDP can be formulated as a *multi-commodity-chain* flow problem on a cloud-augmented graph, and design a queue-length based algorithm, named QNSD, that provides an $O(\epsilon)$ approximation in time $O(1/\epsilon)$. We then address the case in which resource costs are a function of the integer number of allocated resources and design a variation of QNSD that effectively pushes for flow consolidation into a limited number of active resources to minimize overall cloud network cost.**

I. Introduction

Distributed cloud networking builds on network functions virtualization (NFV) and software defined networking (SDN) to enable the deployment of network services in the form of elastic virtual network functions (VNFs) instantiated over general purpose servers at multiple cloud locations, and interconnected by a programmable network fabric that allows dynamically steering client flows [1]-[3]. A *cloud network* operator can then host a variety of services over a common physical infrastructure, reducing both capital and operational expenses. In order to make the most of this attractive scenario, a key challenge is to find the placement of VNFs and the routing of client flows through the appropriate VNFs that minimize the use of the physical infrastructure.

The work in [4] first addressed the problem of placing VNFs in a capacitated cloud network. The authors formulated the problem as a generalization of Facility Location and Generalized Assignment, and provided near optimal solutions with bi-criteria constant approximations guarantees. However, the model in [4] does not account for function ordering or *service chaining* nor flow routing optimization. Subsequently, the work in [5] introduced a flow based model that allows optimizing the distribution (function placement and flow routing) of services with arbitrary function relationships (*e.g.*, chaining) over capacitated cloud networks. Cloud services are described via a directed acyclic graph and the function placement and flow routing is determined by solving a minimum cost network flow problem with service chaining constraints. In [5], it is shown that the cloud service distribution problem (CSDP) admits optimal polynomial-time solutions under convex cost functions, but remains hard otherwise. The model in [5] also allows accounting for multicast flows, in which case, general polynomial solvability requires a convex cost function and the ability to perform intra-session network coding.

In this paper, we address the design of fast approximation algorithms for the NFV Service Distribution Problem (NSDP), a specially relevant class of the CSDP in which the service graph is a line graph (*i.e.*, chain) and all flows are unicast.

Our contributions can be summarized as follows:

- We formulate the NSDP as a minimum cost *multi-commodity-chain* network design (MCCND) problem on a *cloud-augmented graph*, where the goal is to find the placement of service functions, the routing of client flows through the appropriate service functions, and the corresponding allocation of cloud and network resources that minimize the cloud network operational cost.
- We first address the case of load-proportional resource costs. We show that the fractional NSDP becomes a min-cost multi-commodity-chain flow (MCCF) problem that admits optimal polynomial-time solutions. We design a queue-length based algorithm, named QNSD, that is shown to provide an $O(\epsilon)$ approximation to the fractional NSDP in time $O(1/\epsilon)$. We further conjecture that QNSD exhibits an improved $O(1/\sqrt{\epsilon})$ convergence.
- We then address the case in which resource costs are a function of the integer number of allocated resources. We design a new algorithm, C-QNSD, which constrains the evolution of QNSD to effectively drive service flows to consolidate on a limited number of active resources, yielding good practical solutions to the integer NSDP.

The rest of the paper is organized as follows. We review related work in Section II. Section III introduces the system model. Section IV describes the network flow based formulation for the NSDP. Sections V and VI present the proposed approximation algorithms for the fractional and integer NSDP, respectively. We present simulations results in Section VII, discuss possible extensions in Section VIII, and conclude the paper in Section IX.

II. Related Work

To the best of our knowledge, the algorithms presented in this paper are the first approximation algorithms for the NSDP. While the NSDP can be seen as a special case of the CSDP introduced in [5], the authors only provided a network flow based formulation, without addressing the design of efficient approximation algorithms.

As shown in Section V, the fractional NSDP is a generalization of the minimum cost multi-commodity flow (MCF) problem. A large body of work has addressed the design of fast fully polynomial time approximation schemes (FPTAS) for MCF. The work in [6] summarizes the best known FPTAS for MCF and fractional packing problems. Based on [6], the fastest schemes use shortest-path computations in each iteration to provide $O(\epsilon)$ approximations in time $O(1/\epsilon^2)$. The scheme in [7] runs in time $O(1/\epsilon)$, but requires solving a convex quadratic program in each iteration, yielding slower running times for moderately small ϵ. Our proposed QNSD algorithm for the fractional NSDP is shown to provide an $O(\epsilon)$ approximation in time $O(1/\epsilon)$, while simply solving a set of linear time max-weight problems in each iteration. We further conjecture that the running time of our algorithm is in fact $O(1/\sqrt{\epsilon})$. QNSD is hence also an improved FPTAS for MCF. In [8], the authors extended the shortest-path based algorithms in [6] to design policy-aware routing algorithms that steer network flows through pre-defined sequences of network functions in order to maximize the total served flow. The problem addressed in [6] can be thought of as a maximum flow version of the fractional NSDP, but their algorithms still run in time $O(1/\epsilon^2)$.

With respect to the integer NSDP, we show in Section VI that it is a generalization of the well known NP-hard multi-commodity network design (MCND) problem [9]. Our proposed C-QNSD algorithm extends QNSD with knapsack relaxation techniques similar to those used in MCND [9].

III. System Model

A. Cloud network model

We consider a cloud network modeled as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ vertices and $m = |\mathcal{E}|$ edges representing the set of nodes and links, respectively. A cloud network node represents a distributed cloud location, in which virtual network functions or VNFs can be instantiated in the form of *e.g.* virtual machines (VMs) over general purpose servers [1]. When service flows go through VNFs at a cloud node, they consume cloud resources (*e.g.*, cpu, memory). We denote by w_u the cost per cloud resource unit (*e.g.*, server) at node u and by c_u the maximum number of cloud resource units that can be allocated at node u. A cloud network link represents a network connection between two cloud locations. When service flows go through a cloud network link, they consume network resources (*e.g.*, bandwidth). We denote by w_{uv} the cost per network resource unit (*e.g.*, 1 Gbps link) on link (u, v) and by c_{uv} the maximum number of network resource units that can be allocated on link (u, v).

B. Service model

A service $\phi \in \Phi$ is described by a chain of M_ϕ VNFs. We use the pair (ϕ, i), with $\phi \in \Phi, i \in \{1, \dots, M_\phi\}$, to denote the i-th function of service ϕ, and L to denote the total number of available VNFs. Each VNF is characterized by its cloud resource requirement, which may also depend on the specific cloud location. We denote by $r_u^{(\phi,i)}$ the cloud resource requirement (in cloud resource units per flow unit) of function (ϕ, i) at cloud node u. That is, when one flow unit goes through function (ϕ, i) at cloud node u, it consumes $r_u^{(\phi,i)}$ cloud resource units. In addition, when one flow unit goes through the fundamental transport function of link (u, v), it consumes r_{uv}^{tr} network resource units.

A client requesting service $\phi \in \Phi$ is represented by a destination node $d \in \mathcal{D}(\phi) \subset \mathcal{V}$, where $\mathcal{D}(\phi)$ denotes the set of clients requesting service ϕ. The demand of client d for service ϕ is described by a set of source nodes $\mathcal{S}(d, \phi) \in \mathcal{V}$ and demands $\lambda_s^{d,\phi}, \forall s \in \mathcal{S}(d, \phi)$, indicating that a set of source flows, each of size $\lambda_s^{d,\phi}$ flow units and entering the network at $s \in \mathcal{S}(d, \phi)$, must go through the sequence of VNFs of service ϕ before exiting the network at destination node $d \in \mathcal{D}(\phi)$. We set $\lambda_u^{(d,\phi)} = 0, \forall u \notin \mathcal{S}(d, \phi)$ to indicate that only nodes in $\mathcal{S}(d, \phi)$ have source flows for the request of client d for service ϕ. We note that the adopted destination-based client model allows the total number of clients to scale linearly with the size of the network, as opposed to the quadratic scaling of the source-destination client model.

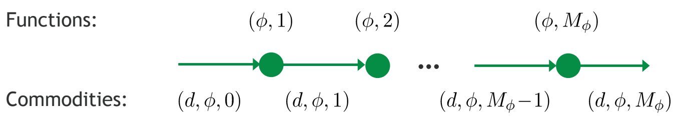


Fig. 1: Multi-commodity-chain flow model. Service ϕ for destination d must take source commodity $(d, \phi, 0)$ and process it via function $(\phi, 1)$ to create commodity $(d, \phi, 1)$, which then needs to be processed by function $(\phi, 2)$ to create commodity $(d, \phi, 2)$... until function (ϕ, M_ϕ) produces final commodity (d, ϕ, M_ϕ).

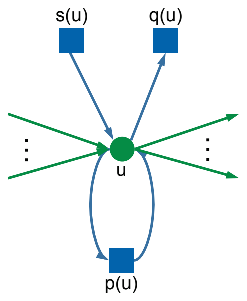


Fig. 2: Cloud-augmented graph for node u, where $p(u)$ represents the processing unit that hosts flow processing functions, $s(u)$ the source unit from which flows enter the cloud network, and $q(u)$ the demand unit via which flows exit the cloud network.

IV. The NFV Service Distribution Problem

The goal of the NFV service distribution problem (NSDP) is to find the placement of service functions, the routing of service flows, and the associated allocation of cloud and network resources, that meet client demands with minimum overall resource cost. In the following, we show how the NSDP can be solved by computing a *chained* network flow on a properly constructed graph.

We adopt a *multi-commodity-chain flow* (MCCF) model, in which a commodity is uniquely identified by the triplet (d,ϕ,i), which indicates that commodity (d,ϕ,i) is the output of the i-th function of service ϕ for client d (see Fig. 1).

We formulate the NSDP as a MCCF problem on the cloud-augmented graph that results from augmenting each node in $\mathcal{G}$ with the gadget in Fig. 2, where $p(u)$, $s(u)$, and $q(u)$ denote the processing unit, source unit, and demand unit at cloud node u, respectively. The resulting graph is denoted by $\mathcal{G}^a = (\mathcal{V}^a, \mathcal{E}^a)$, where $\mathcal{V}^a = \mathcal{V} \cup \mathcal{V}'$ and $\mathcal{E}^a = \mathcal{E} \cup \mathcal{E}'$, with $\mathcal{V}'$ and $\mathcal{E}'$ denoting the set of processing, source, and demand unit nodes and edges, respectively. We denote by $\delta^-(u)$ and $\delta^+(u)$ the set of incoming and outgoing neighbors of node u in $\mathcal{G}^a$.

In the cloud-augmented graph $\mathcal{G}^a$, each edge $(u,v) \in \mathcal{E}^a$ has an associated capacity, unit resource cost, and per-function resource requirement, as follows:

For the set of edges $\{(u,p(u)),(p(u),u) : u \in \mathcal{V}\}$ representing the set of compute resources, we have:

- $c_{u,p(u)} = c_u$, $c_{p(u),u} = c_u^{max}$
- $w_{u,p(u)} = w_u$, $w_{p(u),u} = 0$
- $r_{u,p(u)}^{(\phi,i)} = r_u^{(\phi,i)}, \forall (\phi,i)$

where $c_u^{max} = \sum_{(d,\phi,i)} \sum_{s\in\mathcal{S}(d,\phi)} \lambda_s^{(d,\phi)} r_u^{(\phi,i)}$. Note that we model the processing of network flows and associated allocation of compute resources using link $(u,p(u))$, and let edge $(p(u),u)$ be a free-cost edge of sufficiently high capacity that carries the processed flow back to node u.

For the set of edges $\{(s(u),u),(u,s(u)) : u \in \mathcal{V}\}$ representing the resources via which client flows enter and exit the cloud network, we have:

- $c_{s(u),u} = c_u^{max}$, $c_{u,q(u)} = c_u^{max}$
- $w_{s(u),u} = 0$, $w_{u,q(u)} = 0$
- $r_{s(u),u}^{(\phi,i)} = r_{u,q(u)}^{(\phi,i)} = 0, \forall (\phi,i)$

Note that we model the ingress and egress of network flows via free-cost edges with sufficiently high capacity. In addition, the irrelevant per-function resource requirement for these edges is set to zero.

Given that the set of network resources only perform the fundamental transport function of moving bits between cloud network nodes, the per-function resource requirement for the set of edges in the original graph $(u,v) \in \mathcal{E}$ is set to r_{uv}^{tr} for all (ϕ,i), *i.e.,* $r_{uv}^{(\phi,i)} = r_{uv}^{tr}, \forall(\phi,i)$. The capacity and unit resource cost of network edge $(u,v) \in \mathcal{E}$ is given by c_{uv} and w_{uv}, respectively.

We now define the following MCCF flow and resource allocation variables on the cloud-augmented graph $\mathcal{G}^a$:

- *Flow variables:* $f_{uv}^{(d,\phi,i)}$ indicates the fraction of commodity (d,ϕ,i) on edge $(u,v) \in \mathcal{E}^a$, *i.e.,* the fraction of flow output of function (ϕ,i) for destination d carried/processed by edge (u,v).
- *Resource variables:* y_{uv} indicates the total amount of resource units (*e.g.,* cloud or network resource units) allocated on edge $(u,v) \in \mathcal{E}^a$.

The NSDP can then be formulated via the following compact linear program:

$$\min \quad \sum_{(u,v)\in\mathcal{E}^a} w_{uv}\, y_{uv} \tag{1a}$$

$$\text{s.t.} \quad \sum_{v\in\delta^-(u)} f_{vu}^{(d,\phi,i)} = \sum_{v\in\delta^+(u)} f_{uv}^{(d,\phi,i)} \qquad \forall u,d,\phi,i \tag{1b}$$

$$f_{p(u),u}^{(d,\phi,i)} = f_{u,p(u)}^{(d,\phi,i-1)} \qquad \forall u,d,\phi,i\neq 0 \tag{1c}$$

$$\sum_{(d,\phi,i)} f_{uv}^{(d,\phi,i)} r_{uv}^{(\phi,i+1)} \le y_{uv} \le c_{uv} \qquad \forall (u,v) \tag{1d}$$

$$f_{s(u),u}^{(d,\phi,0)} = \lambda_u^{(d,\phi)} \qquad \forall u,d,\phi \tag{1e}$$

$$f_{u,q(u)}^{(d,\phi,M_\phi)} = 0 \qquad d,\phi,u\neq d \tag{1f}$$

$$f_{uv}^{(d,\phi,i)} \ge 0,\; y_{uv} \in \mathbb{Z}^+ \qquad \forall (u,v),d,\phi,i \tag{1g}$$

where, when not specified for compactness, $u \in \mathcal{V}$, $d \in \mathcal{V}$, $\phi \in \Phi$, $i \in \{1,\ldots,M_\phi\}$, and $(u,v) \in \mathcal{E}^a$.

The objective is to minimize the overall cloud network resource cost, described by (1a). Recall that the set $\mathcal{E}^a$ contains all edges in the augmented graph $\mathcal{G}^a$, representing both cloud and network resources. Eq. (1b) describes standard flow conservation constraints applied to all nodes in $\mathcal{V}$. A specially critical set of constraints are the *service chaining* constraints described by (1c). These constraints establish that in order to have flow of a given commodity (d,ϕ,i) coming out of a processing unit, the input commodity $(d,\phi,i-1)$ must be entering the processing unit. Constraints (1d) make sure that the total flow at a given cloud network resource is covered by enough resource units without violating capacity.[1] Eqs. (1e) and (1f) establish the source and demand constraints. Note from (1f) that no flows of final commodity (d,ϕ,M_ϕ) are allowed to exit the network other than at the destination node d. Finally, (1g) describe the fractional and integer nature of flow and resource allocation variables, respectively.

We define two main versions of the NSDP depending on the integer or fractional nature of the resource allocation variables:

- *Integer NSDP*: The case of fractional flow variables and integer resource variables allows splitting client flows to improve resource utilization while capturing the allocation of an integer number of general purpose resource units (*e.g.,* servers). In this case, the NSDP becomes a generalization of multi-commodity network design (MCND), where the network is augmented with *compute edges* that model the processing of service flows and where there are additional service chaining constraints that make sure flows follow service functions in the appropriate order. In fact, for the special case that each service is composed of a single commodity, the NSDP is equivalent to the MCND. We refer to the resulting MCCND problem as the integer NSDP.
- *Fractional NSDP*: The case in which resource allocation variables are also fractional becomes a specially relevant model when the size of the cloud network resource units

[1]Recall from the MCCF model that commodity (d,ϕ,i) gets processed by function $(\phi,i+1)$, and that $r_{uv}^{(\phi,i)} = r_{uv}^{tr}, \forall (u,v) \in \mathcal{E}, \phi, i$.

is much smaller than the total flow served at a given location. This is indeed the case for services that serve a large number of large-size flows (*e.g.*, telco services [1]) and/or services deployed using small-size resource units (*e.g.*, micro-services [10]). In this case, the NSDP becomes a generalization of min-cost MCF, with the exact equivalence holding in the case of single commodity services. We refer to the resulting MCCF problem as the fractional NSDP.

V. Fractional NSDP

As discussed in the previous section, the fractional NSDP can be formulated as the MCCF problem that results from the linear relaxation of (1), *i.e.*, by replacing $y_{uv} \in \mathbb{Z}^+$ with $y_{uv} \geq 0$ in (1g). While the resulting linear programming formulation admits optimal polynomial-time solutions, it requires solving a linear program with a large number of constraints. In this work, we are interested in designing a fast fully polynomial approximation scheme (FPTAS) for the fractional NSDP. A natural direction would be to build on state of the art FPTAS for MCF that rely on shortest-path computations [6], [8]. However, these techniques have only been shown to provide $O(\epsilon)$ approximations in time $O(1/\epsilon^2)$. Here, we target the design of faster approximations with order improvements in running time, *i.e.*, $O(1/\epsilon)$ and $O(1/\sqrt{\epsilon})$, by redesigning queue-length based algorithms that have been shown to be very effective for stochastic network optimization, in order to construct fast iterative algorithms for static MCCF problems such as the fractional NSDP.

A. QNSD algorithnm

In the following, we describe the proposed queue-length based network service distribution (QNSD) algorithm. QNSD is an iterative algorithm that mimics the time evolution of an underlying cloud network queueing system. QNSD exhibits the following main key features:

- QNSD builds on a dynamic cloud network control algorithm recently introduced in [11], which was designed to stabilize a stochastic cloud network system while minimizing the time-average resource cost, and uses the average over the algorithm iterations to compute the solution to the fractional NSDP.
- Inspired by a recent result that characterizes the transient and steady state phases in queue-length based algorithms [12], QNSD computes the solution to the fractional NSDP by averaging over a limited iteration horizon, yielding an $O(\epsilon)$ approximation in time $O(1/\epsilon)$. In the algorithm description, we use $j \in \mathbb{Z}^+$ to index the *iteration frame* over which averages are computed.
- QNSD further exploits the speed-up shown in gradient methods for convex optimization when combining gradient directions over consecutive iterations [13], [14], leading to a conjectured $O(1/\sqrt{\epsilon})$ convergence.

Before describing the algorithm, we shall introduce the following queue and demand variables:

- *Actual queues*: $Q_u^{(d,\phi,i)}(t)$ denotes the queue backlog of commodity (d,ϕ,i) at node $u \in \mathcal{V}$ in iteration t. These queues represent the actual packet build-up that would take place in an equivalent dynamic cloud network system in which iterations correspond to time instants.
- *Virtual queues*: $U_u^{(d,\phi,i)}(t)$ denotes the virtual queue backlog of commodity (d,ϕ,i) at node $u \in \mathcal{V}$ in iteration t. These virtual queues are used to capture the *momentum* generated when combining differential queue backlogs (acting as gradients in our algorithm) over consecutive iterations [14].

The QNSD algorithm works as follows:

1) Initialization:

$$
\begin{array}{ll}
f_{uv}^{(d,\phi,i)}(0) = y_{uv}(0) = 0 & \forall (u,v) \\
Q_u^{(d,\phi,i)}(0) = 0 & \forall (u,v), d, \phi, i \\
Q_d^{(d,\phi,M_\phi)}(t) = 0 & \forall d, \phi, t \\
U_u^{(d,\phi,i)}(0) = U_u^{(d,\phi,i)}(-1) = 0 & \forall u, d, \phi, i \\
U_d^{(d,\phi,M_\phi)}(t) = 0 & \forall d, \phi, i, t \\
f_{s(u),u}^{(d,\phi,i)}(t) = \left\{ \begin{array}{ll} \lambda_u^{(d,\phi)} & \forall u \in \mathcal{S}(d,\phi), t, i = 0 \\ 0 & \text{otherwise} \end{array} \right. & \\
j = 0 &
\end{array}
$$

Note that the queues associated with the final commodities at their respective destinations are set to zero for all t to model the egress of flows from the cloud network.

2) Main Procedure: In each iteration $t > 0$:

- *Queue updates:* For all $(d,\phi,i) \neq (u,\phi,M_\phi)$:

$$
Q_u^{(d,\phi,i)}(t) = \Bigg[Q_u^{(d,\phi,i)}(t-1) - \sum_{v \in \delta^+(u)} f_{uv}^{(d,\phi,i)}(t-1) + \sum_{v \in \delta^-(u)} f_{vu}^{(d,\phi,i)}(t-1) \Bigg]^+ \quad (2)
$$

$$
\Delta Q_{uv}^{(d,\phi,i)}(t) = Q_u^{(d,\phi,i)}(t) - Q_u^{(d,\phi,i)}(t-1) \quad (3)
$$

$$
U_u^{(d,\phi,i)}(t) = U_u^{(d,\phi,i)}(t-1) + \Delta Q_{uv}^{(d,\phi,i)}(t) + \theta \left(U_u^{(d,\phi,i)}(t-1) - U_u^{(d,\phi,i)}(t-2) \right) \quad (4)
$$

  where $\theta \in [0,1)$ is a control parameter that drives the differential queue backlog momentum. Note that actual queues are updated according to standard queuing dynamics, while virtual queues are updated as a combination of the actual differential queue backlog and the virtual differential queue backlog in the previous iteration.
- *Transport decisions:* For each link $(u,v) \in \mathcal{E}$:
  - Compute the *transport utility weight* of each commodity (d,ϕ,i):

$$
W_{uv}^{(d,\phi,i)}(t) = \frac{1}{r_{uv}^{tr}} \left(U_u^{(d,\phi,i)}(t) - U_v^{(d,\phi,i)}(t) \right)
$$

    where V is a control parameter that governs the optimality v.s. running-time tradeoff.

- Compute the max-weight commodity $(d,\phi,i)^*$:
$$(d,\phi,i)^* = \arg\max_{(d,\phi,i)} \left\{ W_{uv}^{(d,\phi,i)}(t) \right\}$$
- Allocate network resources and assign flow rates
$$y_{uv}(t) = \begin{cases} c_{uv} & \text{if } W_{uv}^{(d,\phi,i)}(t) - V w_{uv} > 0 \\ 0 & \text{otherwise} \end{cases}$$
$$f_{uv}^{(d,\phi,i)^*} = y_{uv}/r_{uv}^{tr}$$
$$f_{uv}^{(d,\phi,i)}(t) = 0, \quad \forall (d,\phi,i) \neq (d,\phi,i)^*$$

• *Processing decisions:* For each node $u \in \mathcal{V}$:
- Compute the *processing utility weight* of each commodity (d,ϕ,i):
$$W_u^{(d,\phi,i)}(t) = \frac{1}{r_u^{(\phi,i+1)}} \left(U_u^{(d,\phi,i)}(t) - U_u^{(d,\phi,i+1)}(t) \right)$$
This key step in the QNSD computes the benefit of processing commodity (d,ϕ,i) via function $(\phi,i+1)$ at node u in iteration t, taking into account the difference between the (virtual) queue backlog of commodity (d,ϕ,i) and that of the next commodity in the service chain $(d,\phi,i+1)$. Note also how a high cloud resource requirement $r_u^{(\phi,i+1)}$ reduces the benefit of processing commodity (d,ϕ,i).
- Compute max-weight commodity $(d,\phi,i)^*$:
$$(d,\phi,i)^* = \arg\max_{(d,\phi,i)} \left\{ W_{uv}^{(d,\phi,i)}(t) \right\}$$
- Allocate cloud resources and assign flow rates
$$y_{u,p(u)}(t) = \begin{cases} c_u & \text{if } W_{uv}^{(d,\phi,i)}(t) - V w_{uv} > 0 \\ 0 & \text{otherwise} \end{cases}$$
$$y_{p(u),u} = y_{u,p(u)}$$
$$f_{u,p(u)}^{(d,\phi,i)^*}(t) = y_{u,p(u)}/r_u^{(\phi,i+1)}$$
$$f_{p(u),u}^{(d,\phi,i+1)^*}(t) = f_{u,p(u)}^{(d,\phi,i)^*}(t)$$
$$f_{u,p(u)}^{(d,\phi,i)}(t) = f_{p(u),u}^{(d,\phi,i)}(t) = 0 \quad \forall (d,\phi,i) \neq (d,\phi,i)^*$$

• *Solution construction:*
- If $t = 2^j$, then $t_{start} = t$ and $j = j+1$
- Flow solution
$$\bar{f}_{uv}^{(d,\phi,i)} = \frac{\sum_{\tau=t_{start}}^{t} f_{uv}^{(d,\phi,i)}(\tau)}{t - t_{start} + 1} \qquad \forall (u,v), d, \phi, i$$
- Resource allocation solution
$$\bar{y}_{uv} = \frac{\sum_{\tau=t_{start}}^{t} y_{uv}(\tau)}{t - t_{start} + 1} \qquad \forall (u,v)$$

□

Remark 1: Note that QNSD solves $m+n$ max-weight problems in each iteration, leading to a running-time per iteration of $O((m+n)nL) = O(mnL)$.[2]

[2]Recall that the number of clients scales as $O(n)$ and L is the total number of functions. Hence, the number of commodities scales as $O(nL)$.

B. Performance of QNSD

Theorem 1: Let the input service demand $\boldsymbol{\lambda} = \{\lambda_u^{(d,\phi)}\}$ be such that the fractional NSDP is feasible and the Slater condition is satisfied. Then, letting $V = 1/\epsilon$, the QNSD algorithm provides an $O(\epsilon)$ approximation to the fractional NSDP in time $O(1/\epsilon)$. Specifically, for all $t \geq T(\epsilon)$, the QNSD solution $\{\bar{f}_{uv}^{(d,\phi,i)}, \bar{y}_{uv}\}$ satisfies:
$$\sum_{(u,v)\in\mathcal{E}^a} w_{uv}\, \bar{y}_{uv} \leq h^{\mathsf{opt}} + O(\epsilon) \tag{5}$$
$$\sum_{v\in\delta^-(u)} \bar{f}_{vu}^{(d,\phi,i)} - \sum_{v\in\delta^+(u)} \bar{f}_{uv}^{(d,\phi,i)} \leq O(\epsilon) \qquad \forall u,d,\phi,i \tag{6}$$
$$\text{(1c)} - \text{(1g)} \tag{7}$$
where h^{opt} denotes the optimal objective function value and $T(\epsilon)$ is an $O(1/\epsilon)$ function, whose expression is derived in the theorem's proof given in Section X.

Proof: See Appendix in Section X. ■

Remark 2: While the claim of Theorem 1 does not specify the dependence of the approximation on the size of the cloud network (m,n), in Section X, we show that, in time $O(m/\epsilon)$, the total cost approximation is $O(m\epsilon)$ and the flow conservation violation is $O(\epsilon)$. The total running time of QNSD is then $O(\epsilon^{-1}m^2nL)$.

Conjecture 1: With a properly chosen $\theta \in [0,1)$, QNSD provides an $O(\epsilon)$ approximation to the fractional NSDP in time $O(1/\sqrt{\epsilon})$.

■

Our conjecture is based on the fact that: *i*) as shown in the proof of Theorem 1, $\theta = 0$ is sufficient for QNSD to achieve $O(1/\epsilon)$ convergence, *ii*) recent results have shown $O(1/\sqrt{\epsilon})$ convergence of queue-length based algorithms for stochastic optimization when including a first-order memory or momentum of the differential queue backlog [14], *iii*) simulation results in Section VII show a significant improvement in the running time of QNSD with nonzero θ.

VI. Integer NSDP

It is immediate to show that the integer NSDP is NP-Hard by reduction from MCND. Recall that the integer NSDP is equivalent to MCND for the special case of single-commodity services. Hence, no $O(\epsilon)$ approximation can in general be computed in sub-exponential time. After recognizing the difficulty of approximating the integer NSDP, we now establish key observations on the behavior of QNSD that allows us to add a simple, yet effective, condition on the evolution of QNSD that enables constructing a solution to the integer NSDP of good practical performance.

We first observe that the QNSD algorithm evolves by allocating an integer number of resources in each iteration. In fact, QNSD solves a max-weight problem in each iteration and allocates either zero or the maximum number of resource units to a single commodity at each cloud network location. However, the solution in each iteration may significantly violate flow conservation constraints. On the other hand, the average over the iterations is shown to converge to a feasible, but, in general, fractional solution. Based on these observations, we propose C-QNSD (constrained QNSD), a variation of QNSD designed to constrain the solution over the algorithm iterations to satisfy flow conservation across consecutive iterations, such that when the iterative flow solution converges, we can guarantee a feasible solution to the integer NSDP. C-QNSD works just as QNSD, but with the max-weight problems solved in each iteration replaced by the fractional knapsak problems that result from adding the *conditional flow conservation constraints*:

$$\sum_{v\in\delta^+(u)} f_{uv}^{(d,\phi,i)}(t) \leq \sum_{v\in\delta^-(u)} f_{vu}^{(d,\phi,i)}(t-1) \qquad \forall d,\phi,i \tag{8}$$

Specifically, in each iteration of the main procedure, after the queue updates described by (2)-(4), the transport and processing decisions are jointly determined as follows:

- *Transport and processing decisions:* For each $u\in\mathcal{V}$:

$$\max \quad \sum_{v\in\delta^+(u)} \sum_{(d,\phi,i)} W_{uv}^{(d,\phi,i)} f_{uv}^{(d,\phi,i)}(t) - V y_{uv}(t) w_{uv}$$

$$\text{s.t.} \quad (8), (1d), (1g)$$

where

$$W_{uv}^{(d,\phi,i)}(t) = \frac{1}{r_{uv}^{tr}} \left(U_u^{(d,\phi,i)}(t) - U_v^{(d,\phi,i)}(t) \right) \qquad \forall v\in\delta^+(u)\backslash\{p(u)\}$$

$$W_{u,p(u)}^{(d,\phi,i)}(t) = \frac{1}{r_u^{(\phi,i+1)}} \left(U_u^{(d,\phi,i)}(t) - U_u^{(d,\phi,i+1)}(t) \right)$$

Observe that without the *conditional flow conservation* constraints (8), the problem above can be decoupled into the set of max-weight problems whose solutions drive the resource allocation and flow rate assignment of QNSD. When including (8), the solution to the above maximization problem is forced to fill-up the cloud network resource units with multiple commodities, smoothing the evolution towards a feasible integer solution. In C-QNSD, the above maximization problem is solved via a linear-time heuristic that decouples the problem into a set of fractional knapsacks, one for each neighbor node $v\in\delta^+(u)$ and resource allocation choice $y_{uv}\in\{0,1,\ldots,c_{uv}\}$.

As shown in the following section, C-QNSD effectively consolidates service flows into a limited number of active resources. Providing some form of performance guarantee is of interest to the authors, but out of the scope of this paper.

VII. Simulation Results

In this section, we evaluate the performance of QNSD and C-QNSD in the context of Abilene US continental network, composed of 11 nodes and 28 directed links, as illustrated in Fig. 3. We assume each node and link is equipped with 10 cloud resource units and 10 network resource units, respectively. The cost per cloud resource unit is set to 1 at nodes 5 and 6, and to 3 at all other nodes. The cost per network resource unit is set to 1 for all 28 links.

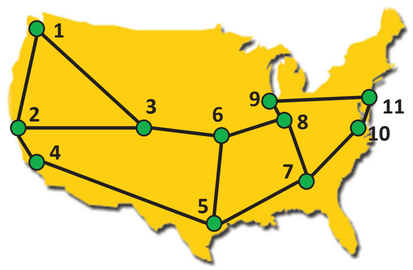
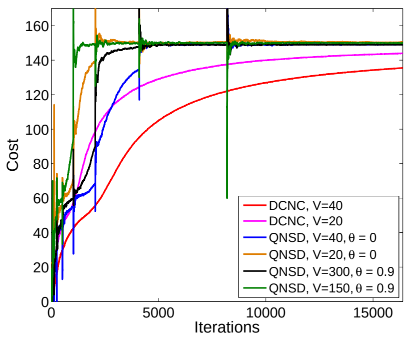


Fig. 3: Abilene US continental network topology.

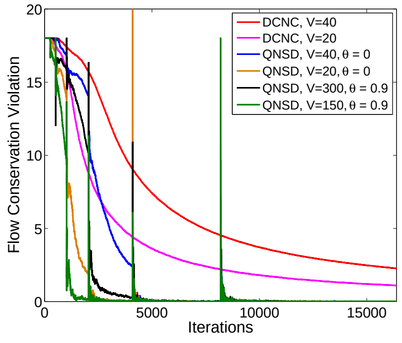
A. QNSD

We first test the performance of QNSD. We consider a scenario with 2 services, each composed of 2 functions. Function $(1,1)$ (service 1, function 1) has resource requirement 1 resource unit per flow unit; while functions $(1,2)$, $(2,1)$, and $(2,2)$ require 3, 2, and 2 resource units per flow unit, respectively. We assume resource requirements do not change across cloud locations, and that all links require 1 network resource unit per flow unit. There are 6 clients, represented by destination nodes $\{1,2,4,7,10,11\}$ in Fig. 3. Each destination node in the west coast $\{1,2,4\}$ has each of the east coast nodes $\{11,10,7\}$ as source nodes, and viceversa, resulting in a total of 18 source-destination pairs. We assume that each east coast client requests service 1 and each west coast client requests service 2, and that all input flows have size 1 flow unit.

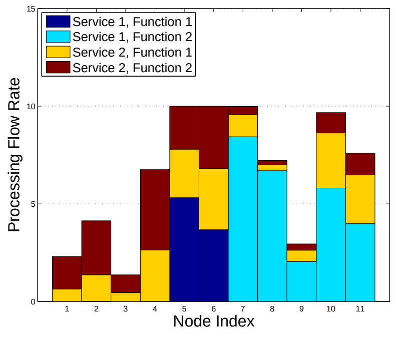
Fig. 4a shows the evolution of the cost function over the algorithm iterations. Recall that QNSD exhibits 3 main features: queue-length driven, truncated average computation, and first-order memory. In Fig. 4, we refer to QNSD without truncation and without memory as DCNC, as it resembles the evolution of the dynamic cloud network control algorithm in [11]. We use QNSD with $\theta=0$ to refer to QNSD with truncation, but without memory. And finally, QNSD with $\theta>0$ refers to the QNSD algorithm with both truncation and memory. It is interesting to observe the improved convergence obtained when progressively including QNSD's key features. Observe how DCNC evolves the slowest and it is not able to reach the optimal objective function value within the 16000 iterations shown. Decreasing the control parameter V from 40 to 20 speeds up the convergence of DCNC, but yields a further-from-optimal approximation, not appreciated in the plot due to the slow convergence of DCNC. In fact, QNSD without truncation and memory can only guarantee a $O(1/\epsilon^2)$ convergence, which is also the convergence speed of the FPTAS for MCF in [6]. On the other hand, when including truncation, QNSD is able to reach the optimal cost value of 149 in around 6000 iterations, clearly illustrating the faster $O(1/\epsilon)$ convergence. The peaks exhibited by the curves of QNSD with truncation illustrate the reset of the average computations at the beginning of each iteration frame. Note again that reducing V further speeds up convergence at the expense of slightly increasing the optimality gap. Finally, when including the memory feature with a value of $\theta=0.9$, QNSD is able to converge to the optimal solution even faster, illustrating the



(a)



(b)



(c)

Fig. 4: Performance of QNSD. a) Evolution of total cost over algorithm iterations; b) Evolution of flow conservation violation over algorithm iterations; c) Processing resource allocation distribution across cloud network nodes after running QNSD with $V = 300$ and $\theta = 0.9$ for 15000 iterations.

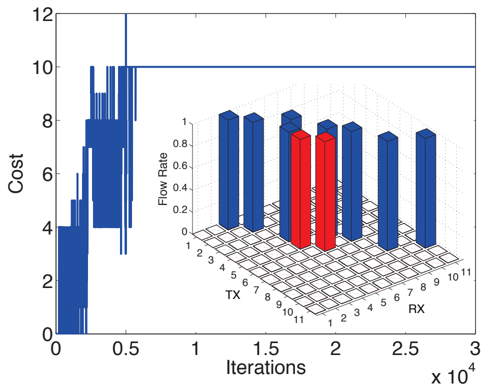


Fig. 5: Cost evolution and flow distribution of the C-QNSD solution with input rate 1 flow unit per client and control parameters $V = 1000$, $\theta = 0.9$.

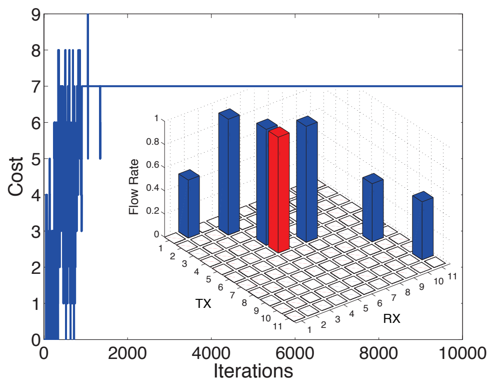


Fig. 6: Cost evolution and flow distribution of the C-QNSD solution with input rate 0.5 flow units per client and control parameters $V = 100$, $\theta = 0.9$.

conjectured $O(1/\sqrt{\epsilon})$ speed-up from the momentum generated when combining gradient directions (see Section V). Decreasing V again illustrates the speed-up v.s. optimality tradeoff.

Fig. 4b shows the convergence of the violation of the flow conservation constraints. We can observe a similar behavior as in the cost convergence, with significant speed-ups when progressively adding truncation and memory to QNSD.

Finally, Fig. 4c shows the processing resource allocation distribution across cloud network nodes. As expected, most of the flow processing concentrates on the cheapest nodes 5 and 6. Note how function $(1, 1)$, which has the lowest processing requirement (1 resource unit per flow unit) gets exclusively implemented in nodes 5 and 6, as it gets higher priority in QNSD's scheduling decisions. Functions $(2, 1)$ and $(2, 2)$, which require 2 resource units per flow unit, share the remaining processing capacity at nodes 5 and 6. Finally, function $(1, 2)$, with resource requirement 3 resource units per flow unit, and following function $(1, 1)$ in the service chain, gets distributed closer to the east coast nodes, destinations for service 1.

B. C-QNSD

In order to test the performance of C-QNSD, we setup a scenario with 2 s-d pairs, $(1, 11)$ and $(2, 7)$, both requesting one service composed of one function with resource requirement 1 cloud resource per flow unit. We simulate the performance of C-QNSD for input rates 1 flow unit and 0.5 flow unit per client. Observe from Fig. 5 that for input rate 1, C-QNSD is able to converge to a solution of total cost 10, in which each client flow follows the shortest path and where the flow processing happens at nodes 5 and 6, respectively. The 3D bar plot shows the flow distribution over the links (non-diagonal entries in blue) and nodes (diagonal entries in red) in the network. Observe now the solution for input rate 0.5 in Fig. 6. The flow of s-d pair $(1, 11)$ is now routed along the longer path $\{1, 2, 4, 5, 7, 10, 11\}$ in order to consolidate as much flow as possible on the activated resources for s-d pair $(2, 7)$. The flow processing of both services is now consolidated at node 5, yielding an overall cost of 7, which is in fact the optimal solution to the integer NSDP in this setting. Note that if the two client flows were following the shortest path and separately getting processed at nodes 5 and 6, without exploiting the available resource consolidation opportunities, the total cost under integer resources would be 10.

While preliminary, our results show promise on the combined use of momentum information and conditional flow conservation constraints for providing efficient solutions to the integer NSDP in practical network settings.

VIII. Extensions

While not included in this paper for ease of exposition, our model and algorithms can be extended to include:

- *Function availability:* Limiting the availability of certain functions to a subset of cloud nodes can be modeled by setting the flow variables associated with function (ϕ, i) to zero, $f_{p(u),u}^{(d,\phi,i)} = 0$, for all d and for all cloud nodes u in which function (ϕ, i) is not available.
- *Function flow scaling:* Capturing the fact that flows can change size as they go through certain functions can be modeled by letting $\xi^{(\phi,i)}$ denote the number of output flow units per input flow unit of function (ϕ, i), and modifying the service chaining constraints (1c) as $\xi^{(\phi,i)} f_{p(u),u}^{(d,\phi,i)} = f_{u,p(u)}^{(d,\phi,i-1)}$.
- *Per-function resource costs:* If the cost function depends on the number of virtual resource units (*e.g.*, VMs) instead of on the number of physical resource units (*e.g.*, servers), then we can use $y_{uv}^{(\phi,i)}$ and $w_{uv}^{(\phi,i)}$ to denote the number of allocated resource units of function (ϕ, i) and the cost per resource unit of function (ϕ, i), respectively.
- *Nonlinear cost function:* In order to capture nonlinear cost effects such as *economies of scale* in which the cost per resource unit decreases with the number of allocated resource units, the cost function can be modified as $\sum_{uv} y_{uv,k} w_{uv,k}$, with $y_{uv,k} \in \{0,1\}$ indicating the allocation of k resource units, and $w_{uv,k}$ denoting the cost associated with the allocation of k resource units. The capacity constraints in (1d) become $\sum_{(d,\phi,i)} f_{uv}^{(d,\phi,i)} r_{uv}^{(\phi,i+1)} \le k\, y_{uv,k} \le c_{uv}$.

IX. Conclusions

We have formulated the NFV service distribution problem (NSDP) as a multi-commodity-chain network design problem on a cloud-augmented graph. We have shown that under load-proportional costs, the resulting fractional NSDP becomes a multi-commodity-chain network flow problem that admits optimal polynomial time solutions, and have designed QNSD, a queue-length based iterative algorithm that provides an $O(\epsilon)$ approximation in time $O(1/\epsilon)$. We further conjectured that, by exploiting the momentum obtained when combining differential queue backlogs across consecutive iterations, QNSD converges in time $O(1/\sqrt{\epsilon})$, and illustrated it via simulations. We then addressed the case in which resource costs are a function of the integer number of allocated resources. We showed that the integer NSDP is NP-Hard by reduction from MCND and designed C-QNSD, a heuristic algorithm that constrains the evolution of QNSD to effectively consolidate flows into a limited number of active resources.

X. Appendix: Proof of Theorem 1

Let

$$\mathbf{Q}(t+1) = [\mathbf{Q}(t) + \mathbf{A}\mathbf{f}(t)]^{+} \tag{10}$$

denote the matrix form of the QNSD queuing dynamics given by (2), where $\mathbf{f}(t)$, $\mathbf{Q}(t)$ and $\mathbf{A}$ denote the flow vector in iteration t and the queue-length vector in iteration t, and the incident matrix of the cloud-augmented network, respectively.

Let

$$Z(\mathbf{Q}) \triangleq \inf_{[\mathbf{y},\mathbf{f}]\in\mathcal{X}} \left\{V\mathbf{w}^{\dagger}\mathbf{y} + (\mathbf{A}\mathbf{f})^{\dagger}\mathbf{Q}\right\} \tag{11}$$

denote the dual function of the fractional NSDP weighted by the control parameter V, where $\mathbf{w}$ is the resource cost vector and $\mathcal{X}$ is the set of feasible solutions to the fractional NSDP.

Let $\mathcal{H}^* \triangleq \{\mathbf{Q}^*\!: \ \mathbf{Q}^* = \arg\max_{\mathbf{Q}} Z(\mathbf{Q})\}$ denote the set of all optimizers of $Z(\mathbf{Q})$.

Let $\mathcal{H}_\gamma \triangleq \{\mathbf{Q} : \sup_{\mathbf{Q}^*\in\mathcal{H}^*}\{\|\mathbf{Q}-\mathbf{Q}^*\|\} \le \gamma\}$ denote the set of queue-length vectors that are at most γ-away from any point in $\mathcal{H}^*$.

Since $Z(\mathbf{Q})$ is a piecewise linear concave function of $\mathbf{Q}$, the locally polyhedron property is satisfied, *i.e.*, $\forall\mathbf{Q} \notin \mathcal{H}_\gamma$ and $\forall\mathbf{Q}^* \in \mathcal{H}^*$, there exists a $L_\gamma > 0$ such that

$$Z(\mathbf{Q}^*) - Z(\mathbf{Q}) \ge L_\gamma \|\mathbf{Q}^* - \mathbf{Q}\|. \tag{12}$$

We then denote the set of queue-length vectors that are D-away from $\mathcal{H}^*$ as

$$\mathcal{H}_D \triangleq \left\{\mathbf{Q} : \sup_{\mathbf{Q}^*\in\mathcal{H}^*}\{\|\mathbf{Q}-\mathbf{Q}^*\|\} \le D\right\}, \tag{13}$$

with $D \triangleq \max\left\{\frac{B}{L_\gamma} - \frac{L_\gamma}{4}, \frac{L_\gamma}{2}, \gamma\right\}$, where B is an upper bound for $\|\mathbf{A}\mathbf{f}\|^2$, defined as

$$B \triangleq 2m \max_{(u,v)\in\mathcal{E}^a} \frac{c_{uv}^2}{\min_{(\phi,i)}\left(r_{uv}^{(\phi,i)}\right)^2} = O(m). \tag{14}$$

In order to prove Theorem 1, we first state the following lemmas, whose proofs can be found in [15].

Lemma 1: Under QNSD, if $\mathbf{Q}(t) \notin \mathcal{H}_D$, then

$$\mathbf{Q}(t+1) - \mathbf{Q}^*\| \le \|\mathbf{Q}(t) - \mathbf{Q}^*\| - \frac{L_\gamma}{2}, \ \ \forall\mathbf{Q}^* \in \mathcal{H}^*. \tag{15}$$

Proof: See [15, Section X]. ■

Lemma 2: Assuming the Slater condition holds, and letting

$$\mathcal{R}_D \triangleq \left\{\mathbf{Q} : \sup_{\mathbf{Q}^*\in\mathcal{H}^*} \|\mathbf{Q}-\mathbf{Q}^*\| \le D + \sqrt{B}\right\}, \tag{16}$$

$$\tau_{\mathcal{R}_D} \triangleq \inf\{t \ge 0 : \mathbf{Q}(t) \in \mathcal{R}_D\}, \tag{17}$$

then, for all $t \ge \tau_{\mathcal{R}_D}$, $\mathbf{Q}(t) \in \mathcal{R}_D$.

Proof: See [15, Section X]. ■

Lemma 3: Letting $h^{\max} = \max_{\mathbf{y}}\left\{\mathbf{w}^{\dagger}\mathbf{y}\right\}$, if the Slater condition holds, the queue-length vector of QNSD satisfies

$$\|\mathbf{Q}(t)\| \le \frac{B/2 + Vh^{\max}}{\kappa} + \sqrt{B}, \qquad \forall t > 0, \tag{18}$$

where κ is a positive number satisfying $\|\mathbf{Q}(t+1)\|^2 - \|\mathbf{Q}(t)\|^2 \le B + Vh^{\max} - \kappa\|\mathbf{Q}(t)\|$.

Proof: See [15, Section X]. ■

We now proceed to prove Theorem 1. Starting from the queuing dynamics in (10), squaring both sides of the equality, recalling that $\|\mathbf{A}\mathbf{f}(t)\|^2 \leq B$, and adding $V\mathbf{w}^\dagger\mathbf{y}(t)$ on both sides, after algebraic manipulation, we get

$$\frac{1}{2}\left[\|\mathbf{Q}(t+1)\|^2 - \|\mathbf{Q}(t)\|^2\right] + V\mathbf{w}^\dagger\mathbf{y}(t) \leq \frac{B}{2} + V\mathbf{w}^\dagger\mathbf{y}(t) + \mathbf{Q}(t)^\dagger\mathbf{A}\mathbf{f}(t). \quad (19)$$

As described in Section V-A, QNSD computes the resource and flow vectors $[\mathbf{y}(t), \mathbf{f}(t)]$ in iteration t as

$$[\mathbf{y}(t), \mathbf{f}(t)] = \arg \inf_{[\mathbf{y}', \mathbf{f}'] \in \mathcal{X}} \left\{V\mathbf{w}^\dagger\mathbf{y}' + (\mathbf{A}\mathbf{f}')^\dagger\mathbf{Q}(t)\right\}, \quad (20)$$

which implies that, for any feasible $[\hat{\mathbf{y}}, \hat{\mathbf{f}}]$,

$$V\mathbf{w}^\dagger\mathbf{y}(t) + \mathbf{Q}(t)^\dagger\mathbf{A}\mathbf{f}(t) \leq V\mathbf{w}^\dagger\hat{\mathbf{y}} + \mathbf{Q}(t)^\dagger\mathbf{A}\hat{\mathbf{f}}. \quad (21)$$

Using (21) and evaluating the right-hand side of (19) at the optimal solution $[\mathbf{y}^{\mathsf{opt}}, \mathbf{f}^{\mathsf{opt}}]$, we obtain

$$\begin{aligned}\frac{1}{2}\big(\|\mathbf{Q}(t+1)\|^2 - \|\mathbf{Q}(t)\|^2\big) + V\mathbf{w}^\dagger\mathbf{y} &\leq \frac{B}{2} + V\mathbf{w}^\dagger\mathbf{y}^{\mathsf{opt}} + \mathbf{Q}(t)^\dagger\mathbf{A}\mathbf{f}^{\mathsf{opt}}(t)\\ &= \frac{B}{2} + V\mathbf{w}^\dagger\mathbf{y}^{\mathsf{opt}} \quad (22)\end{aligned}$$

where the last equality follows from $\mathbf{A}\mathbf{f}^{\mathsf{opt}}(t) = \mathbf{0}$. Letting $h^{\mathsf{opt}} = \mathbf{w}^\dagger\mathbf{y}^{\mathsf{opt}}$, from (22), we have

$$\mathbf{w}^\dagger\mathbf{y}(t) - h^{\mathsf{opt}} \leq \frac{B}{2V} - \frac{1}{2V}\left(\|\mathbf{Q}(t+1)\|^2 - \|\mathbf{Q}(t)\|^2\right). \quad (23)$$

And taking the average of (23) over the j-th iteration frame $[t_j, t_j + \Delta_{t_j}]$, after algebraic manipulation, we obtain

$$\frac{1}{\Delta_{t_j}+1}\sum_{\tau=t_j}^{t_j+\Delta_{t_j}} \mathbf{w}^\dagger\mathbf{y}(\tau) - h^{\mathsf{opt}} \leq \frac{B}{2V} + \frac{\big(\|\mathbf{Q}(t_j)\|^2 - \|\mathbf{Q}(t_j+\Delta_{t_j})\|^2\big)}{2V(\Delta_{t_j}+1)}. \quad (24)$$

Next, we use the following two properties, whose proofs follow from Lemmas 1, 2, and 3, and can be found in [15]:

- **R1:** $\tau_{\mathcal{R}_D} = O\left(h^{\mathsf{max}}V / L_\gamma\kappa\right) = O(mV)$.
- **R2:** if $t_j \geq \tau_{\mathcal{R}_D}$, then $\|\mathbf{Q}(t_j)\|^2 - \|\mathbf{Q}(t_j+\Delta_{t_j})\|^2 = O((D+\sqrt{B})h^{\mathsf{max}}V / L_\gamma\kappa) = O(m^2V)$.

Using **R1** and **R2**, and letting $V = 1/\epsilon$, and $\Delta_{t_j} \geq \lceil mV - 1\rceil$, it follows from (25) that, for all $t_j \geq \tau_{\mathcal{R}_D}$,

$$\mathbf{w}^\dagger\left(\sum_{\tau=t_j}^{t_j+\Delta_{t_j}} \frac{\mathbf{y}(\tau)}{\Delta_{t_j}+1}\right) - h^{\mathsf{opt}} = O\left(\frac{m}{V}\right) = O(m\epsilon). \quad (25)$$

Observing that $\bar{y}_{uv} = \sum_{\tau=t_j}^{t_j+\Delta_{t_j}} \frac{y_{uv}(\tau)}{\Delta_{t_j}+1}$, Eq. (5) in Theorem 1 follows. In order to prove Eq. (6), taking the average of (10) over $[t_j, t_j+\Delta_{t_j}]$, it follows that, for all $t_j \geq \tau_{\mathcal{R}_D}$,

$$\sum_{\tau=t_j}^{t_j+\Delta_{t_j}} \frac{(\mathbf{A}\mathbf{f}(t) + \lambda)}{\Delta_{t_j}+1} \leq \frac{\mathbf{Q}(t_j+\Delta_{t_j}) - \mathbf{Q}(t_j)}{\Delta_{t_j}+1}. \quad (26)$$

Finally, the $\{u, (d, \phi, i)\}$-th element of the vectors in (26) satisfies

$$\begin{aligned}&\sum_{v\in\delta^-(u)} \bar{f}_{vu}^{(d,\phi,i)} - \sum_{v\in\delta^+(u)} \bar{f}_{uv}^{(d,\phi,i)}\\ &\leq \frac{\left|Q_u^{(d,\phi,i)}(t_j+\Delta_{t_j}) - Q_u^*\right| + \left|Q_u^{(d,\phi,i)}(t_j) - Q_u^*\right|}{\Delta_{t_j}+1}\\ &\leq \frac{2(D+\sqrt{B})}{mV} = O\left(\frac{1}{V}\right) = O(\epsilon), \quad (27)\end{aligned}$$

where

$$\bar{f}_{vu}^{(d,\phi,i)} = \sum_{\tau=t_j}^{t_j+\Delta_{t_j}} \frac{f_{vu}^{(d,\phi,i)}(\tau)}{\Delta_{t_j}+1}, \quad \bar{f}_{uv}^{(d,\phi,i)} = \sum_{\tau=t_j}^{t_j+\Delta_{t_j}} \frac{f_{uv}^{(d,\phi,i)}(\tau)}{\Delta_{t_j}+1}.$$

Hence, from (25) and (27), we have that, for all $t \geq T(\epsilon) \triangleq \tau_{\mathcal{R}_D} + \Delta_{t_j} + 1 \geq \tau_{\mathcal{R}_D} + mV = O(m/\epsilon)$, the QNSD solution approximates the optimal cost to within $O(m\epsilon)$ and violates the flow conservation constraints by $O(\epsilon)$. In addition, it is immediate to see that, by construction, the QNSD solution trivially satisfies constraints (1c)-(1g), which concludes the proof of Theorem 1.

■

References

[1] "Network Functions Virtualization: An Introduction, Benefits, Enablers, Challenges and Call for Action," *ETSI*, White Paper, October 2012.

[2] "Software-defined networking: the new norm for networks," *Open Networking Foundation*, 2012.

[3] Bell Labs Strategic White Paper, "The Programmable Cloud Network - A Primer on SDN and NFV," June 2013.

[4] R. Cohen, L. Lewin-Eytan, J. Naor, D. Raz, "Near Optimal Placement of Virtual Network Functions," *IEEE INFOCOM*, April 2015.

[5] M. Barcelo, J. Llorca, A. Tulino, N. Raman, "The Cloud Servide Distribution Problem in Distributed Cloud Networks," *IEEE ICC*, June 2015.

[6] N. Garg, J. Konemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," *SIAM Journal on Computing*, 2007.

[7] D. Bienstock, G. Iyengar, "Approximating fractional packings and coverings in O(1/epsilon) iterations." *SIAM Journal on Computing*, 2006.

[8] Z. Cao, M. Kodialam, T. V. Lakshman, "Traffic Steering in Software Defined Networks: Planning and Online Routing," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, 2014.

[9] T. G. Crainic, A. Frangioni, B. Gendron, "Bundle-based relaxation methods for multicommodity capacitated fixed charge network design," *Discrete Applied Mathematics*, vol. 112, pp. 73-99, 2001.

[10] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, Ruslan Mustafin, Larisa Safina, "Microservices: yesterday, today, and tomorrow," *arxiv*, June 2016.

[11] H. Feng, J. Llorca, A. Tulino, A. Molisch, "Dynamic Service Optimization in Distributed Cloud Networks," *IEEE INFOCOM SWFAN Workshop*, April 2016.

[12] S. Supittayapornpong, L. Huang, and M. Neely, "Time-average stochastic optimization with non-convex decision set and its convergence," *IEEE CDC*, December 2014.

[13] P. Tseng, "On accelerated proximal gradient methods for convex-concave optimization, *SIAM Journal on Optimization*, 2008.

[14] J. Liu, A. Eryilmaz, N. B. Shroff, E. S. Bentley, "Heavy-Ball: A new approach to tame delay and convergence in wireless network optimization," *IEEE INFOCOM*, April 2016.

[15] https://www.dropbox.com/sh/mr47alszzkytczy/AACht6vK5ZN7o97-iyH5MEEsa?dl=0

[16] D. P. Bertsekas, "Convex optimization theory," *Belmont: Athena Scientific*, pp. 157-226, 2009.