# Orchestrating Data-Intensive vNF Service Chains in Inter-DC Elastic Optical Networks

Wei Lu, Lipei Liang, Zuqing Zhu[†]
School of Information Science and Technology, University of Science and Technology of China, Hefei, China
[†]Email: {zqzhu}@ieee.org

***Abstract*—We investigate the problem of data-intensive vNF service chain (vNF-SC) orchestration in inter-datacenter EONs. After analyzing the $\mathcal{NP}$-hardness of this problem, we solve it in a sequential manner by optimizing both the request serving sequence and the data-intensive vNF-SC orchestration. Specifically, we propose a request sorting algorithm and a data-intensive vNF-SC orchestration algorithm based on dynamic programming to minimize the service completion time. We conduct simulations to evaluate the proposed algorithms, and simulation results verify their effectiveness.**

***Index Terms*—Network Function Virtualization, vNF Service Chain, Bulk-Data Transfer, Elastic optical networks.**

## I. Introduction

Network function virtualization (NFV) emerges as a promising network architecture framework to facilitate the deployment of new network services [1]. Under the paradigm of NFV, network operators can realize virtual network functions (vNFs) on general-purpose servers to replace traditional middleboxes. A network service corresponds to a request of vNF-SC that demands its data traffic to be processed and forwarded by a sequence of vNFs [2]. In inter-datacenter (inter-DC) networks, vNFs can be pre-deployed on high-volume servers in DCs to support vNF-SC requests. Meanwhile, elastic optical networks (EONs) [3, 4] have been considered as a promising physical infrastructure of next-generation inter-DC networks [5], since they can facilitate agile bandwidth management in the optical layer. Hence, it would be relevant to investigate how to efficiently serve vNF-SC requests in inter-DC EONs.

The problem of vNF-SC provisioning involves both vNF-SC embedding and steering traffic among vNFs. Basically, for vNF-SC embedding, network operators need to find a specific DC to instantiate each vNF required in the vNF-SC, while for traffic steering among vNFs, network operators need to establish lightpaths between vNFs, assign a certain amount of bandwidth on them, and perform task scheduling in DCs to provide satisfying vNF-SC services. With diverse requirements on traffic steering, vNF-SC requests can be generally divided into two types: 1) bandwidth-intensive vNF-SC requests that demand bandwidth-guaranteed connections in between vNFs, and 2) data-intensive vNF-SC requests each of which needs to transfer a specific amount of data through a sequence of vNFs before a deadline. When serving either type of vNF-SC requests, network operators have to manage bandwidth resources and IT resources jointly to not only satisfy the quality-of-service (QoS) requirements of vNF-SC services but also realize high efficiency of resource utilization.

Previously, researchers have studied the problem of orchestrating bandwidth-/data-intensive vNF-SCs under different network scenarios [6–10]. However, these studies have not yet considered EONs as the physical infrastructure, and thus their solutions cannot be applied to the vNF-SC orchestration in inter-DC EONs. Meanwhile, the existing studies on bandwidth-intensive vNF-SC orchestration in inter-DC EONs [11–13] have not addressed the scheduling of data transfers through vNF-SCs. For instance, the work in [11] formulated an integer linear programming (ILP) model to solve the problem of bandwidth-intensive vNF-SC orchestration exactly and also proposed several heuristics to use bandwidth and IT resources in a balanced manner. Zeng *et al.* [12, 13] studied tree-type bandwidth-intensive vNF-SC orchestration in inter-DC EONs, formulated a mixed integer linear programming (MILP) model, and proposed two path-intersection-based heuristics. However, to the best of our knowledge, the problem of data-intensive vNF-SC orchestration in inter-DC EONs have not been touched yet, especially when the scheduling of data transfers has to be considered. Since data-intensive services are surging in today's inter-DC networks, it is not only important but also necessary for us to investigate how to serve data-intensive vNF-SCs effectively in inter-DC EONs.

In this work, we consider an inter-DC EON in which there are dynamic background traffic and pre-deployed vNFs and study how to orchestrate data-intensive vNF-SCs to minimize their average service completion time (SCT). We first explain the problem and analyze its complexity, and then solve it in a sequential manner by optimizing both the serving sequence of vNF-SCs and each vNF-SC's provisioning scheme. Numerical simulations are conducted to evaluate the performance of the proposed vNF-SC orchestration algorithm, and the results verify that the proposed algorithm can significantly reduce the average SCT of data-intensive vNF-SC requests.

The rest of the paper is organized as follows. Section II provides the problem description. In Section III, we propose a request sorting algorithm and a data-intensive vNF-SC orchestration algorithm based on dynamic programming. Section IV discusses the simulations for performance evaluation. Finally, we summarize the paper in Section V.

## II. Problem Description and Analysis

We model an inter-DC EON as $G(V, L)$, where $V$ is the DC set and $L$ is the established lightpath set. The set of vNF types that can be instantiated in the network is denoted as $T = \{vNF_1, vNF_2, \cdots, vNF_N\}$, where $N$ is the total number of vNF types. Accordingly, $\Gamma = \{\gamma_1, \gamma_2, \cdots, \gamma_N\}$ and $\Delta = \{\delta_1, \delta_2, \cdots, \delta_N\}$ are the sets of processing rate and data change ratio[1] of the vNF types, respectively. There are certain pre-deployed vNFs on each node $v \in V$, which are denoted as set $T_v \subseteq T$. Between each node pair $(s, d) \in V^2$, there are established lightpaths denoted as set $L_{s,d}$, each of which is carrying dynamic background traffic and thus has 2D spectrum fragments on it [5]. Here, a 2D spectrum fragment refers to a non-aligned and isolated unused bandwidth block in both the time and spectrum domains. More specifically, for the $k$-th lightpath in $L_{s,d}$, we denote its 2D spectrum fragment set as $\Psi^k_{s,d} = \{(t^{s,d,k}_{s,m}, t^{s,d,k}_{e,m}, bw^{s,d,k}_m) : m = 1, 2, \cdots\}$, where the elements in the tuple $(t^{s,d,k}_{s,m}, t^{s,d,k}_{e,m}, bw^{s,d,k}_m)$ are the start-time, end-time and bandwidth, respectively, of the $m$-th 2D spectrum fragment.

We model the $i$-th data-intensive vNF-SC request as $R_i(s_i, d_i, \zeta_{i0}, SC_i)$, where $s_i$ and $d_i$ are the source and destination nodes, $\zeta_{i0}$ is the initial data volume from the source node, and $SC_i = \{f_{i1}, f_{i2}, \cdots, f_{iN_i}\}$ is the requested vNF-SC, in which each vNF $f_{ij}$ can only be realized with a pre-deployed one, *i.e.*, the DC node $v$ that is used to carry $f_{ij}$ should satisfy $f_{ij} \in T_v$, and $N_i$ is the total number of requested vNFs. Note that, the data volume of $R_i$ could change after each vNF, due to the vNFs' data change ratios. Thus, we use $\{\zeta_{i0}, \zeta_{i1}, \cdots, \zeta_{iN_i}\}$ to record the data volumes, in which the data volume after vNF $f_{ij}$ is $\zeta_{ij} = \zeta_{i(j-1)} \cdot \delta_{f_{ij}}$.

To serve a data-intensive vNF-SC request, the network operator needs to: 1) select a DC node $v^*_{ij} \in V_{ij}$[2] to carry each vNF $f_{ij} \in SC_i$, 2) select available lightpaths from $L_{s_i,v^*_{i1}} \cup \{L_{v^*_{ij},v^*_{i(j+1)}} : j \in [1, (N_i - 1)]\} \cup L_{v^*_{iN_i},d_i}$ to connect vNF-SC $s_i \rightarrow v^*_{i1}(f_{i1}) \cdots v^*_{iN_i}(f_{iN_i}) \rightarrow d_i$, and 3) schedule data transfers over the vNF-SC. Note that, we consider the service completion time (SCT) as the key metric to evaluate the service of a data-intensive vNF-SC. Basically, the vNF-SC's service begins when the source node $s_i$ forwards the initial data $\zeta_{i0}$ to the first vNF $f_{i1}$, and ends when the destination node $d_i$ has received all the data $\zeta_{iN_i}$ from the last vNF $f_{iN_i}$. During this process, whenever the data $\zeta_{ij}$ is ready to be transferred between a vNF pair $(s', d')$, where $s'$ and $d'$ are two adjacent nodes on the vNF-SC, a valid 2D spectrum fragment $(t^{s',d',k^*}_{s,m^*}, t^{s',d',k^*}_{e,m^*}, bw^{s',d',k^*}_{m^*})$ that satisfies

$$\zeta_{ij} \leq \left[ t^{s',d',k^*}_{e,m^*} - t^{s',d',k^*}_{s,m^*} + 1 \right] \cdot bw^{s',d',k^*}_{m^*}, \tag{1}$$

on a lightpath $l_{s',d',k^*} \in L_{s',d'}$ would be used to accomplish the data transfer. Note that, if a feasible 2D spectrum fragment cannot be found on the established lightpaths immediately, the data would be buffered on node $s'$. Hence, there could be "data-to-be-transferred" buffering delay, besides the data transfer latency.

Meanwhile, whenever a DC node $v^*_{ij}$ has received all the data $\zeta_{i(j-1)}$ from the previous node, a data processing task for vNF $f_{ij}$ is created, and task scheduling is performed to reserve a valid service time window (ST-Wnd) $[t^{f_{ij},v^*_{ij}}_{s^*}, t^{f_{ij},v^*_{ij}}_{e^*}]$ on it, which satisfies

$$\left[ t^{f_{ij},v^*_{ij}}_{e^*} - t^{f_{ij},v^*_{ij}}_{s^*} + 1 \right] \geq \left\lceil \frac{\zeta_{i(j-1)}}{\gamma_{f_{ij}}} \right\rceil, \tag{2}$$

*i.e.*, the duration is long enough to process the data $\zeta_{i(j-1)}$ with a processing rate of $\gamma_{f_{ij}}$, and also does not overlap with any other task's ST-Wnd in the same DC[3]. Therefore, if the data $\zeta_{i(j-1)}$ arrives at DC node $v^*_{ij}$ before its scheduled ST-Wnd, it would be buffered before being processed by vNF $f_{ij}$, *i.e.*, there is a "data-to-be-processed" buffering delay, besides the data-processing latency. When the data processing has been done, the new data $\zeta_{ij}$ is generated and should be transferred to the next vNF in the built vNF-SC. In this work, we assume that a DC node only has one active instance for each of its pre-deployed vNFs, which can only process the data from one task at each time slot (TS), *i.e.*, the one processor assumption in scheduling theory [14].[4] Finally, the SCT of a data-intensive vNF-SC is the summation of the total "data-to-be-processed" buffering, data processing, and "data-to-be-transferred" buffering delay in the vNFs and the total data transfer latency on the lightpaths. In this work, we consider a dynamic network scenario where the data-intensive vNF-SC requests come and leave on the fly and study how to orchestrate them effectively to minimize the average SCT.

Fig. 1 shows an example of data-intensive vNF-SC orchestration. There are four data-intensive vNF-SC requests. We first build their vNF-SCs by selecting DC nodes and assigning lightpaths, part of which is snapshot as Fig. 1(a). In the built vNF-SCs, each arrow represents the data transfer in between two vNFs/nodes, the notation above which is for the assigned lightpath. The notation in the brackets beside each vNF is the selected DC node. Fig. 1(b) shows the task scheduling and data transfer results. For a vNF-SC, its task scheduling in vNFs and data transfers on lightpaths should be in sequence. For example, $R_1$'s data transfer on $L1$ has to be scheduled before its data processing in $vNF_1$ on $DC1$, followed by the data transfer on $L5$ and the data processing in $vNF_5$ on $DC3$. Note that, since the ST-Wnd on $L5$ is discontinued with that in $vNF_1$ on $DC1$, the data of $R_1$ processed by $vNF_1$ has to be buffered in $DC1$ until the ST-Wnd on $L5$ begins, *i.e.*, there is a "data-to-be-transferred" buffering delay. After $R_1$'s data has been transferred on $L5$, a ST-Wnd of $vNF_5$ on $DC3$ has already been arranged for $R_4$, and thus the data of $R_1$ cannot be processed by $vNF_5$ immediately and has to be buffered in $DC3$, *i.e.*, there is a "data-to-be-processed" buffering delay.

[1]Here, the data change ratio of a vNF refers to the ratio of its output data volume to its input data volume.

[2]In this paper, we use a decision variable with a star superscript to represent its final value from problem solving.

[3]Here, we consider non-preemptive task scheduling, *i.e.*, a task cannot be divided into small pieces during task scheduling.

[4]Note that, this assumption only affects the calculation of the earliest ST-Wnd for a task in *Lines* 1 and 5 of *Algorithm* 2, but has no influence on the main bodies of the proposed request sorting algorithm in Section III. A and data-intensive vNF-SC orchestration algorithm based on dynamic programming in Section III. B.

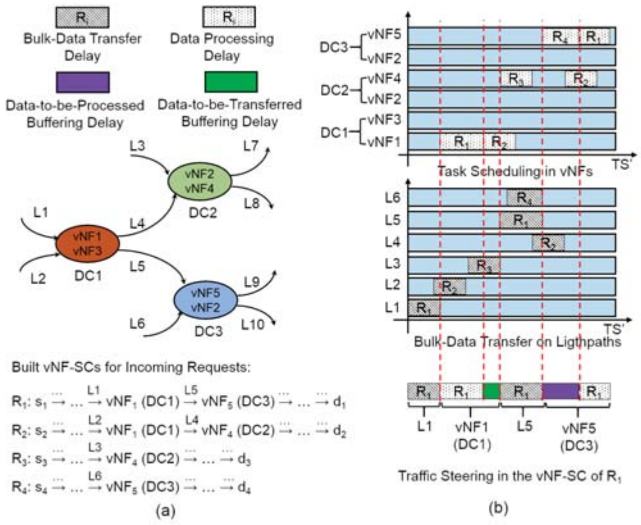

Bulk-Data Transfer Delay
Data Processing Delay
Data-to-be-Processed Buffering Delay
Data-to-be-Transferred Buffering Delay
vNF1 vNF3
DC1
vNF2 vNF4
DC2
vNF5 vNF2
DC3
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10
Built vNF-SCs for Incoming Requests:
R1: s1 → … →L1 vNF1 (DC1) →L5 vNF5 (DC3) → … → d1
R2: s2 → … →L2 vNF1 (DC1) →L4 vNF4 (DC2) → … → d2
R3: s3 → … →L3 vNF4 (DC2) → … → d3
R4: s4 → … →L6 vNF5 (DC3) → … → d4
(a)
DC3 vNF5 vNF2
DC2 vNF4 vNF2
DC1 vNF3 vNF1
Task Scheduling in vNFs
Bulk-Data Transfer on Ligthpaths
TS
L1 vNF1 (DC1) L5 vNF5 (DC3)
Traffic Steering in the vNF-SC of R1
(b)

Fig. 1. Example of data-intensive vNF-SC orchestration.

Regarding the optimization objective of minimizing the average SCT of data-intensive vNF-SCs, we only need to minimize the "data-to-be-transferred" and "data-to-be-processed" buffering delays in vNFs and the data transfer delays on lightpaths, both of which can be affected by the $vNF \rightarrow DC$ mapping and the 2D spectrum fragments on lightpaths. This is because the data processing delay in vNFs is fixed in total. If there are always enough spectrum resources on lightpaths, the "data-to-be-transferred" buffering delay and the data transfer delay could be ignored, making the optimization objective become to minimize the "data-to-be-processed" buffering delay in vNFs. Then, the data-intensive vNF-SC orchestration problem becomes the well-known flexible job shop problem, which is strong $\mathcal{NP}$-hard. Hence, with/without spectrum constraints on lightpaths, the data-intensive vNF-SC orchestration problem investigated in this work is strong $\mathcal{NP}$-hard. With this insight in mind, we turn to efficient heuristics directly. Basically, there are two ways to solve the problem: one is to orchestrate all the requested data-intensive vNF-SC together, while the other is to do it in sequence. In this work, we focus on the latter and try to optimize both the serving sequence of vNF-SCs and the provisioning scheme of each vNF-SC.

## III. Proposed Service Provisioning Scheme

### A. Request Sequencing Optimization

As illustrated in Fig. 1(b), an earlier vNF-SC's service can affect that of a latter one, *e.g.*, $R_1$'s "data-to-be-processed" delay in $DC3$ is actually caused by the data processing of $R_4$. Thus, a good request serving sequence is desirable to relieve this kind of follow-up effect and eventually minimize the average SCT of vNF-SCs. Here, we propose a scheme that properly considers the relation between the serving sequence and the optimization objective.

Firstly, we define the following variables:

- $\mathcal{P}$: the set of pending data-intensive vNF-SC requests.
- $\omega$: the network status before any request in $\mathcal{P}$ is served.
- $o(\mathcal{P}, \omega)$: starting with status $(\mathcal{P}, \omega)$, the smallest total SCT of data-intensive vNF-SCs in $\mathcal{P}$ with the optimal serving sequence.
- $o(\mathcal{P}, \omega, i, j)$: starting with status $(\mathcal{P}, \omega)$ and having requests $i$ and $j$ served successively, the smallest total SCT of data-intensive vNF-SCs in $\mathcal{P}/\{i,j\}$ with the optimal serving sequence.
- $g(\omega, i)$: a function to calculate the smallest SCT of request $i$ given a network status $\omega$.
- $\tau(\omega, i)$: the new network status associated with the result of function $g(\omega, i)$.
- $\tau(\omega, i, j)$: the network status after serving requests $i$ and $j$ successively given a network status $\omega$.

Then, we get:

$$\begin{aligned} & o\Big(\mathcal{P}, \omega, i, j\Big) \\ & = g\Big(\omega, i\Big) + g\Big(\tau(\omega, i), j\Big) + o\Big(\mathcal{P}/\{i,j\}, \tau(\omega, i, j)\Big), \end{aligned} \tag{3}$$

$$\begin{aligned} & o\Big(\mathcal{P}, \omega, j, i\Big) \\ & = g\Big(\omega, j\Big) + g\Big(\tau(\omega, j), i\Big) + o\Big(\mathcal{P}/\{i,j\}, \tau(\omega, j, i)\Big). \end{aligned} \tag{4}$$

And, if we have:

$$o\Big(\mathcal{P}, \omega, i, j\Big) \leq o\Big(\mathcal{P}, \omega, j, i\Big), \tag{5}$$

request $i$ should be served before request $j$ to minimize the total SCT. However, since there is no way to quantify the value of $o(\mathcal{P}, \omega)$, we design the following metrics.

$$e\Big(\omega, i\Big) = g\Big(\omega, i\Big) + \sum_{r \in \mathcal{P}/\{i\}} g\Big(\tau(\omega, i), r\Big), \tag{6}$$

where the second term on the right side qualifies a request's follow-up effect in an ideal manner. If we have $e(\omega, i) \leq e(\omega, j)$, we should serve request $i$ before request $j$ to minimize the follow-up effect, and *vice versa*.

Then, we propose a multi-round request sorting algorithm, as shown in *Algorithm* 1. Here, we store the sorted pending requests in $\mathcal{P}^*$, which is initialized as an empty set (*Line* 1). In each request-sorting round, we select the request $r^*$ with minimum $e(\omega, r^*)$ (*Lines* 3-6). Once the selected request has been served (*Line* 7), we update the network status $\omega$, the pending request set $\mathcal{P}$, and the sorted pending request set $\mathcal{P}^*$ (*Line* 8). Then, the algorithm goes to the next round until all the pending requests in $\mathcal{P}$ have been served (*Lines* 2-9).

**Complexity Analysis:** the time complexity of *Algorithm* 1 is $O(CP(g(\cdot)) \cdot |\mathcal{P}|^3)$, where $CP(g(\cdot))$ is the time complexity of function $g(\cdot)$ and operation $|\cdot|$ returns the element number in a set.

### B. Data-Intensive vNF-SC Orchestration

For function $g(\omega, i)$, we propose a data-intensive vNF-SC orchestration algorithm based on dynamic programming, in which we define the following variables:

- $t(s_i, f_{ij}, v^z_{ij})$: the earliest TS that the data of $R_i$, originating from the source node $s_i$, can be completely processed by vNF $f_{ij}$ on the $z$-th DC node $v^z_{ij} \in V_{ij}$.

**Algorithm 1:** Multi-Round Request Sorting Algorithm

```
    Input: $\mathcal{P}$, $\omega$, $g(\cdot)$, $\tau(\cdot)$;
    Output: $\mathcal{P}^*$;
1   $\mathcal{P}^* = \emptyset$;
2   while $\mathcal{P} \neq \emptyset$ do
3       for any request $r$ in $\mathcal{P}$ do
4           calculate $e(\omega, r)$ with Eq. (6);
5       end
6       select request $r^*$ with the minimum $e(\omega, r^*)$;
7       serve request $r^*$ with the scheme of $g(\omega, r^*)$;
8       $\omega = \tau(\omega, r^*)$, $\mathcal{P} = \mathcal{P}/\{r^*\}$, $\mathcal{P}^* = \mathcal{P}^* \leftarrow r^*$;
9   end
```

- $t(v_{ij}^z)$: the earliest TS that vNF $f_{ij}$ on the $z$-th DC node $v_{ij}^z \in V_{ij}$ can complete the data processing of $R_i$.
- $t(v_{i(j-1)}^{z'}, v_{ij}^z, \zeta_{i(j-1)})$: the earliest TS that the data transfer of $R_i$ can be completed between vNF $f_{i(j-1)}$ on the $z'$-th DC node $v_{i(j-1)}^{z'} \in V_{i(j-1)}$ and vNF $f_{ij}$ on the $z$-th DC node $v_{ij}^z \in V_{ij}$. This definition is also applied to the cases in which the source node $s_i$ or the destination node $d_i$ is an end node.
- $t(s_i, d_i)$: the minimum SCT of $R_i$, *i.e.*, the value of function $g(\omega, i)$.

Then, we write the recursive relationship as:

$$t\Big(s_i, f_{i1}, v_{i1}^z\Big) = \Big\lceil t\Big(v_{i1}^z\Big)\Big|t\Big(s_i, v_{i1}^z, \zeta_{i0}\Big)\Big\rceil, \tag{7}$$

$$\begin{aligned} t\Big(s_i, f_{ij}, v_{ij}^z\Big) = \min_{v_{i(j-1)}^{z'}} \Big\{ & t\Big(v_{ij}^z\Big)\Big|t\Big(v_{i(j-1)}^{z'}, v_{ij}^z, \zeta_{i(j-1)}\Big)\Big| \\ & t\Big(s_i, f_{i(j-1)}, v_{i(j-1)}^{z'}\Big)\Big\}, \ \forall j \in [2, m_i], \end{aligned} \tag{8}$$

$$t\Big(s_i, d_i\Big) = \min_{v_{im_i}^z} \Big\{ t\Big(v_{im_i}^z, d_i, \zeta_{i(m_i-1)}\Big)\Big|t\Big(s_i, f_{im_i}, v_{im_i}^z\Big)\Big\}, \tag{9}$$

where the expression "$A|B$" means the value of $A$ under the condition $B$, and similarly, the expression "$A|B|C$" means the value of $A$ under the conditions $B$ and $C$.

To calculate $t(v_{i(j-1)}^{z'}, v_{ij}^z, \zeta_{i(j-1)})|t(s_i, f_{i(j-1)}, v_{i(j-1)}^{z'})$, we can always select the best 2D spectrum fragment as:

$$\Big\{k^*, m^*\Big\} = \arg\min_{k,m} \left\{ t_{s,m}^{v_{i(j-1)}^{z'}, v_{ij}^z, k} + \left\lceil \frac{\zeta_{i(j-1)}}{bw_m^{v_{i(j-1)}^{z'}, v_{ij}^z, k}} \right\rceil \right\}. \tag{10}$$

Besides Eq. (1), the select 2D spectrum fragment must satisfy

$$t_{s,m^*}^{v_{i(j-1)}^{z'}, v_{ij}^z, k^*} \geq t\Big(s_i, f_{i(j-1)}, v_{i(j-1)}^{z'}\Big), \tag{11}$$

to ensure sequential vNF-SC service. Then, we can get

$$\begin{aligned} & t\Big(v_{i(j-1)}^{z'}, v_{ij}^z, \zeta_{i(j-1)}\Big)\Big|t\Big(s_i, f_{i(j-1)}, v_{i(j-1)}^{z'}\Big) \\ & = t_{s,m^*}^{v_{i(j-1)}^{z'}, v_{ij}^z, k^*} + \left\lceil \frac{\zeta_{i(j-1)}}{bw_{m^*}^{v_{i(j-1)}^{z'}, v_{ij}^z, k^*}} \right\rceil. \end{aligned} \tag{12}$$

For $t(v_{ij}^z)|t(v_{i(j-1)}^{z'}, v_{ij}^z, \zeta_{i(j-1)})|t(s_i, f_{i(j-1)}, v_{i(j-1)}^{z'})$, we need to perform task scheduling in DC node $v_{ij}^z$ to find the earliest ST-Wnd. To do so, instead of putting the data-processing task right after the earlier tasks, *e.g.*, in Fig. 1(b), $R_1$'s data-processing task in $vNF_5$ is scheduled right after $R_4$'s data-processing task in $vNF_5$ on $DC_3$, we can try to reschedule the earlier tasks' ST-Wnds to squeeze an earlier ST-Wnd for the task. This is because, for those earlier tasks that have "data-to-be-transferred" buffering delay, their scheduled ST-Wnds can be delayed without any adverse influences on their SCTs. For example, in Fig. 1(b), if the data transfer of $R_4$ following its data processing in $vNF_5$ on $DC_3$ can be delayed to provide $R_1$'s data processing in $vNF_5$ an earlier ST-Wnd, we can schedule $R_1$ before $R_4$ as long as $R_4$'s data processing can still be completed before its data transfer.

However, the problem of finding the earliest ST-Wnd with task rescheduling is also strong $\mathcal{NP}$-hard [14]. Based on binary search and a well-verified minimum late task scheduling (MLTS) algorithm in [15], we propose a task rescheduling algorithm to search the earliest ST-Wnd for the target task. The detailed procedure is shown in *Algorithm* 2. We define the following notations:

- $J$: set of former tasks for rescheduling, each task $J_i \in J$ has a specific tuple $(r_i, p_i, d_i)$, where $r_i$ is the task's ready time, $p_i$ is the processing time, $d_i$ is the deadline.
- $J_\dagger = \{r_\dagger, p_\dagger, d_\dagger\}$: the target task, in which $r_\dagger$ is the task's ready time calculated with Eq. (12), $p_\dagger$ is the processing time calculated as the right side of Eq. (2), and $d_\dagger$ is the deadline, the value of which is changed iteratively and finally equals the end-time of the earliest ST-Wnd.
- $d_{\dagger,upper}$: the upper bound of the target task's deadline.
- $d_{\dagger,lower}$: the lower bound of the target task's deadline.
- $C(J)$: the maximum task completion time using the MLTS algorithm to schedule tasks in $J$.
- $N(J)$: the number of late tasks using the MLTS algorithm to schedule tasks in $J$.

*Line* 1 calculates $C(J)$ using the MLTS algorithm to get the upper bound of the target task's deadline. *Line* 2 initializes the values of $d_{\dagger,upper}$ and $d_{\dagger,lower}$. *Line* 3 sets the target task's deadline as $d_{\dagger,upper}$ and puts the target task into set $J$ for task scheduling. The while loop of *Lines* 4-12 first uses the MLTS algorithm to get a valid scheduling scheme for the tasks in $J$ with the minimum number of late tasks, *i.e.*, the ones that have missed their deadlines (*Line* 5). If the number of late tasks $N(J)$ is 0, the value of $d_{\dagger,upper}$ is updated as $d_\dagger$ (*Line* 7). Otherwise, the value of $d_{\dagger,lower}$ is updated as $d_\dagger$ (*Line* 9). *Line* 11 updates the value of $d_\dagger$ with binary search to prepare for the next round. When $d_{\dagger,upper} = d_{\dagger,lower}$, the while loop stops and the value of $d_\dagger$ becomes the end-time of the earliest ST-Wnd. Then, we can get

$$t\Big(v_{ij}^z\Big)\Big|t(v_{i(j-1)}^{z'}, v_{ij}^z, \zeta_{i(j-1)})\Big|t\Big(s_i, f_{i(j-1)}, v_{i(j-1)}^{z'}\Big) = d_\dagger. \tag{13}$$

Finally, based on the recursive relation described in Eqs. (7)-(9), we can get the value of $t(s_i, d_i)$, and the associated data-intensive vNF-SC orchestration for $R_i$.

**Complexity Analysis:** the time complexity of *Algorithm* 2 is $O(|J|^2 \cdot \log(d_{\dagger,upper}))$, where $d_{\dagger,upper}$ is calculated in *Line* 2. Then, the time complexity of the data-intensive vNF-SC orchestration algorithm based on dynamic programming is $O(N_i \cdot |V| \cdot \overline{|J|^2 \cdot \log(d_{\dagger,upper})})$, where $\overline{|J|^2 \cdot \log(d_{\dagger,upper})}$ is the average value of $|J|^2 \cdot \log(d_{\dagger,upper})$. Hence, $CP(g(\omega, i))$ equals to $O(N_i \cdot |V| \cdot \overline{|J|^2 \cdot \log(d_{\dagger,upper})})$.

**Algorithm 2:** Task Rescheduling for Earliest Service Time Window

```
    Input: $J$, $J_\dagger = \{r_\dagger, p_\dagger, d_\dagger\}$;
    Output: $d_\dagger$;
1   calculate $C(J)$ with the MLTS algorithm;
2   $d_{\dagger,upper} = C(J) + p_\dagger$, $d_{\dagger,lower} = r_\dagger + p_\dagger$;
3   $d_\dagger = d_{\dagger,upper}$, $J = J \leftarrow J_\dagger$;
4   while $d_{\dagger,upper} \neq d_{\dagger,lower}$ do
5       calculate $N(J)$ with the MLTS algorithm;
6       if $N(J) = 0$ then
7           $d_{\dagger,upper} = d_\dagger$;
8       else
9           $d_{\dagger,lower} = d_\dagger$;
10      end
11      $d_\dagger = \left\lceil \frac{d_{\dagger,upper} + d_{\dagger,lower}}{2} \right\rceil$, update $J_\dagger$ in $J$;
12  end
```

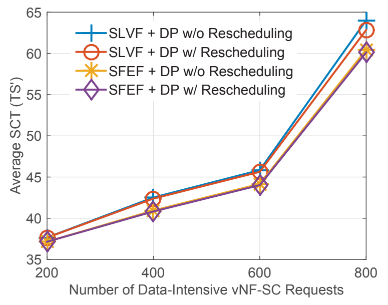
## IV. Performance Evaluation

We conduct simulations using Matlab R2013a to evaluate the proposed algorithms under both light and heavy background traffic scenarios. In detail, we use the 14-node NSFNET topology as an inter-DC EON, in which each node has several bandwidth-variable optical switches (BV-OXCs) and bandwidth-variable transponders (BV-Ts) and connects with a local DC. We assume that 10 types of vNFs are available and each DC node has randomly deployed $[2, 4]$ of those vNF types. Between each DC pair, there are two established lightpaths at most, each of which occupies 11 frequency slots (FS'). We make the background traffic use the lightpaths' bandwidth along the time axis and leave $12.14\%$ and $2.67\%$ bandwidth on average as the 2D spectrum fragments in the light and heavy background traffic scenarios, respectively. Each simulation covers around 2000 TS', in which the data-intensive vNF-SC requests are generated with the Poisson process and the service provision period is 40 TS'. For each data-intensive vNF-SC, the number of requested vNFs is 5 on average, and the initial data volume is uniformly distributed within $[2, 6]$ FS·TS. Regarding the vNFs' processing rates, we set their values in a range of $[0.56, 1.12]$ times of the transmission rate of an FS. And we set the data change ratios within $[0.7, 1.3]$.

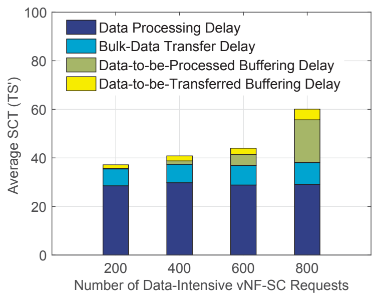
To evaluate the performance of our proposed request sorting and task rescheduling algorithms, we use a request sorting algorithm that serves the requests that have less requested vNFs and smaller data volume earlier as the benchmark, and compare the cases with and without task rescheduling. For the sake of distinction, we name our proposed multi-round request sorting algorithm as the smallest follow-up effect first (SFEF), while call the benchmark as the smallest vNF-SC length and data volume first (SLVF). For the proposed data-intensive vNF-SC orchestration algorithm based on dynamic programming, we name it as "DP w/ Rescheduling" and "DP w/o Rescheduling" for with and without task rescheduling. Hence, by combining request sorting and data-intensive vNF-SC orchestration, we have four different service provisioning schemes in the simulations.

Fig. 2 shows the results in the light background traffic scenario. Fig. 2(a) compares the results on average SCT. We can see that the schemes with SFEF can significantly reduce the average SCT when being compared with the schemes with the SLVF. This observation verifies the effectiveness of the proposed multi-round request sorting algorithm. However, when comparing the schemes with/without task rescheduling, we notice that the task rescheduling can only slightly reduce the average SCT if the request sorting scheme is the same. This is mainly because in the light background traffic scenario, there are sufficient 2D spectrum fragments on the established lightpaths and thus the data processed in DCs has a lot of opportunities to be transferred immediately. Therefore, the number of vNF-SCs that have a "data-to-be-transferred" buffering delay (*i.e.*, the value of $|J|$ in *Algorithm* 2) is relatively small. To prove this, we show the distributions of the average SCT from "SFEF + DP w/ Rescheduling" and "SLVF + DP w/o Rescheduling" in Figs. 2(b) and 2(c), respectively. In both figures, the "data-to-be-processed" and "data-to-be-transferred" buffering delays are much shorter than the data processing and bulk-data transfer delays. Furthermore, comparing Figs. 2(b) and 2(c), we find that, the "data-to-be-processed" and "data-to-be-transferred" buffering delays in Fig. 2(b) are shorter than those in Fig. 2(a), which verifies the effectiveness of the proposed request sorting and task rescheduling algorithms again.

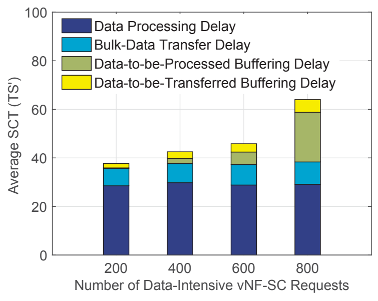
Fig. 3 shows the results in the heavy background traffic scenario. Fig. 3(a) compares the average SCT obtained by the four algorithms. Again, we can clearly observe the advantage of the proposed request sorting algorithm. Moreover, it is interesting to notice that this time, the schemes with task rescheduling achieve much shorter average SCT than those without task rescheduling. This observation verifies the effectiveness of the proposed task rescheduling. The reason behind this is that the number of vNF-SCs that have a "data-to-be-transferred" buffering delay is relatively large when the background traffic load is heavy. Basically, in the heavy background traffic scenario, there are only a few 2D spectrum fragments on the established lightpaths and hence the processed data in DCs has very few opportunities to be transferred immediately, which increases the "data-to-be-transferred" buffering delay, as shown in Figs. 3(b) and 3(c). Similarly, by comparing Figs. 3(b) and 3(c), we can see that "SFEF + DP w/ Rescheduling" provides shorter "data-to-be-processed" and "data-to-be-transferred" buffering delays than "SLVF + DP w/o Rescheduling".



(a) Average SCT of data-intensive vNF-SCs

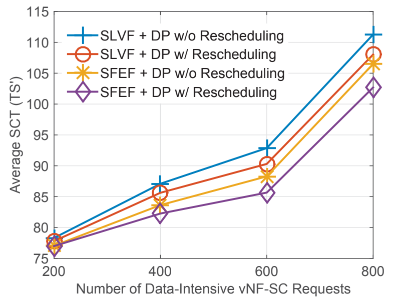


(b) Distribution of average SCT when using "S-FEF + DP w/ Rescheduling"

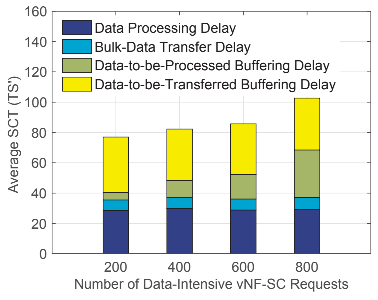


(c) Distribution of average SCT when using "S-LVF + DP w/o Rescheduling"

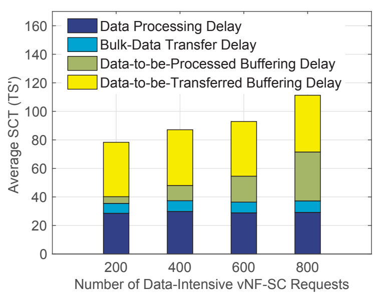
Fig. 2. Results in light background traffic scenario.



(a) Average SCT of data-intensive vNF-SCs



(b) Distribution of average SCT when using "S-FEF + DP w/ Rescheduling"



(c) Distribution of average SCT when using "S-LVF + DP w/o Rescheduling"

Fig. 3. Results in heavy background traffic scenario.

## V. Summary

In this paper, we studied how to serve the data-intensive vNF-SCs in an inter-DC EON to minimize their average SCT. We proposed a request sorting algorithm to minimize the follow-up effect and a data-intensive vNF-SC orchestration algorithm based on dynamic programming. Simulation results verified that the proposed request sorting algorithm can significantly reduce the average SCT when being compared with a benchmark that considers no follow-up effect and indicated that task rescheduling is only helpful in the heavy background traffic scenario in light of the much longer "data-to-be-transferred" buffering delay caused by insufficient 2D spectrum fragments.

## Acknowledgments

This work was supported in part by the NSFC Project 61371117, Natural Science Research Project for Universities in Anhui (KJ2014ZD38), the Strategic Priority Research Program of the CAS (XDA06011202), and the China Postdoctoral Science Foundation (2016M602031).

## References

[1] Network functions virtualization. [Online]. Available: http://www.etsi.org/technologies-clusters/tech-nologies/nfv
[2] Q. Sun, P. Lu, W. Lu, and Z. Zhu, "Forecast-assisted NFV service chain deployment based on affiliation-aware vNF placement," in *Proc. of GLOBECOM 2016*, pp. 1–6, Dec. 2016.
[3] Z. Zhu, W. Lu, L. Zhang, and N. Ansari, "Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing," *J. Lightw. Technol.*, vol. 31, pp. 15–22, Jan. 2013.
[4] P. Lu *et al.*, "Highly-efficient data migration and backup for big data applications in elastic optical inter-datacenter networks," *IEEE Netw.*, vol. 29, pp. 36–42, Sept./Oct. 2015.
[5] W. Lu, Z. Zhu, and B. Mukherjee, "Optimizing deadline-driven bulk-data transfer to revitalize spectrum fragments in EONs," *J. Opt. Commun. Netw.*, vol. 7, pp. B173–B183, Dec. 2015.
[6] M. Xia *et al.*, "Network function placement for NFV chaining in packet/optical datacenters," *J. Lightw. Technol.*, vol. 33, pp. 1565–1570, Apr. 2015.
[7] J. Herrera and J. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Trans. Netw. Serv. Manag.*, vol. 13, pp. 518–532, Aug. 2016.
[8] T. Kuo, B. Liou, K. Lin, and M. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," in *Proc. of INFOCOM 2016*, pp. 1–9, Apr. 2016.
[9] I. Jang *et al.*, "Optimal network resource utilization in service function chaining," in *Proc. of NetSoft 2016*, pp. 11–14, Jun. 2016.
[10] T. Lin, Z. Zhou, M. Tornatore, and B. Mukherjee, "Demand-aware network function placement," *J. Lightw. Technol.*, vol. 34, pp. 2590–2600, Jun. 2016.
[11] W. Fang *et al.*, "Joint spectrum and IT resource allocation for efficient VNF service chaining in inter-datacenter elastic optical networks," *IEEE Commu. Lett.*, vol. 20, pp. 1539–1542, Aug. 2016.
[12] M. Zeng, W. Fang, J. Rodrigues, and Z. Zhu, "Orchestrating multicast-oriented NFV trees in inter-DC elastic optical networks," in *Proc. of ICC 2016*, pp. 1–6, May 2016.
[13] M. Zeng, W. Fang, and Z. Zhu, "Orchestrating tree-type VNF forwarding graphs in inter-DC elastic optical networks," *J. Lightw. Technol.*, vol. 34, pp. 3330–3341, Jul. 2016.
[14] P. Brucker, *Scheduling Algorithms*. Springer, 2006.
[15] S. Dauzère-Pérès, "Minimizing late jobs in the general one machine scheduling problem," *Eur. J. Oper. Res.*, vol. 81, pp. 134–142, Feb. 1995.