A Fast and Scalable Resource Allocation Scheme for End-to-End Network Slices

Panagiotis I. Nikolaidis and John S. Baras
Department of Electrical & Computer Engineering and the Institute for Systems Research
University of Maryland, College Park, MD 20742, USA
Email: {nikolaid, baras}@umd.edu

***Abstract*—We propose an online resource allocation scheme for end-to-end network slices. Our aim is to provide guarantees for the overall delay of the slice users, while minimizing the operational costs of the mobile network operator. Hence, we develop our scheme based on an optimization problem, where we jointly perform bandwidth allocation in the radio access network, and service function chain embedding in the core network. We show that the scheme has polynomial time complexity. The performance of our scheme is evaluated via simulations. Results clearly show that the proposed scheme is fast and highly scalable with respect to the number of users.**

***Index Terms*—network slicing, software defined networking (SDN), network function virtualization (NFV)**

I. Introduction

Next generation mobile networks need to provide services to a diverse set of applications. These applications could range anywhere from Internet of Things (IoT) to mobile cloud gaming. Clearly, the "one size fits all" approach in network design, is no longer efficient when the use cases have such diverse requirements and conflicting performance metrics.

These concerns led to the emergence of network slicing. A network slice is an independent end-to-end virtual network that is tailored to meet the quality of service (QoS) requirements of a single application [1]. Thus, the deployment of multiple network slices on the same physical infrastructure resolves the previous issue. In parallel implementation of network slices is enabled by network function virtualization (NFV) [2] and software defined networking (SDN) [3]. With NFV, mobile network operators (MNOs) can dynamically create multiple service function chains (SFCs) by deploying virtual machines on server stacks in the core network (CN) [2]. Then, the SDN controllers update the forwarding tables of the programmable switches in order to steer the traffic through the newly deployed virtual network functions (VNFs) [2].

There is rich literature on the resource allocation (RA) for the CN component of a slice, which involves variants of the virtual network embedding (VNE) problem [4]. In [5], a log-competitive approximation algorithm is introduced that accepts SFC requests one by one. In [6], the VNF placement problem (VNF-PP) is linked to the generalized assignment problem (GAP). In [7], an efficient heuristic algorithm for the VNF-PP is introduced. Lastly, in [8], a throughput optimal control policy for distributed computing networks is developed.

In all these papers, the radio access network (RAN) component of the slice is missing. However, it is crucial to include it in the analysis, since a network slice is an end-to-end virtual network that provides end-to-end performance guarantees.

In this paper, we develop an end-to-end online RA scheme (RAS) that includes the RAN. We formulate a mixed integer non linear programming (MINLP) problem to bound the end-to-end delay of the slice users, with the objective to minimize the operational costs of the MNO. The problem consists of two smaller problems; the bandwidth allocation in the RAN, and the SFC embedding problem (SFC-EP) in the CN. The two problems are coupled via the end-to-end delay constraint. For this reason, we develop a simple and intuitive decomposition method. Once we decouple the two problems, we solve the MINLP problem in the RAN, optimally with polynomial time complexity. Then, for the INLP problem in the CN, we propose a greedy algorithm with polynomial time complexity.

Consequently, we present a polynomial time RAS that provides end-to-end guarantees in network slicing. We note that to adapt to the changes of the wireless channel conditions, it is important that the RA is fast, which is indeed the case for the proposed RAS.

The remainder of the paper is organized as follows. In Section II, we describe the system model. In Section III, we formulate the end-to-end optimization problem. In Section IV, we develop a decomposition method, solve the RAN problem optimally, and provide a fast greedy algorithm for the CN problem. In Section V, we evaluate the performance of our RAS via simulations. Lastly, Section VI concludes the paper.

II. System Model

In our framework, the virtual MNOs (VMNOs) receive network slice requests from the application service providers (ASPs). The ASPs request a network slice to provide the QoS needed for their end users. We consider that the VMNO receives the slice requests online, one by one. We associate with each request, a maximum allowed end-to-end delay T_d, and a number of users K. Each user produces a stream of packets with constant arrival rate λ, constant packet length L, that need to reach a destination node n_d. Also, each packet flow requires the embedding of a SFC, represented by an ordered set $\mathcal{V}_r$, e.g., $\mathcal{V}_r =$ (firewall, encryption, decryption).

This material is based upon work supported by the Office of Naval Research award no. N000141712622 and the Defense Advanced Research Projects Agency (DARPA), to the University of Maryland College Park.

In short, the slice requests can be expressed as a 6-tuple $(K, \lambda, L, T_d, \mathcal{V}_r, n_d)$.

We assume that the allowed delay, packet arrival rate, packet length, and the required SFC for each user is the same, since a network slice corresponds to a specific application, thus its users have similar traffic statistics and requirements. Lastly, the destination node is fixed given that it corresponds to a data center of the ASP.

The CN consists of programmable switches that are controlled by SDN controllers. Let $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$ denote the directed graph that represents the CN, where $\mathcal{V}_s$ denotes the set of programmable switches, and $\mathcal{E}_s$ the set of edges that are realized by optical fibers connecting the switches. Some switches $\mathcal{V}_{ss} \subset \mathcal{V}_s$ are attached to server stacks, where the placement of VNFs is feasible. In the RAN, we consider a single cell scenario, where one base station (BS) serves the K users. We assume that the 5G New Radio (5G NR) protocol with orthogonal frequency division multiple access (OFDMA) is deployed. The channel conditions between a user and a carrier vary over time.

The RA decisions are computed at the orchestrator, an entity that has a global view of the RAN and the CN. The orchestrator receives the global network view, which contains the channel conditions in the RAN and the queue lengths in the CN, from the BS and the SDN controllers. Once the RA decisions are computed by the orchestrator, they are passed down to the BS and the SDN controllers. Then, the SDN controllers update the flow rules in the switches and the BS updates the scheduling in the RAN. We note the update period of the RAS must be smaller than the coherence time[1] in the RAN but greater than the time required for the SDN controllers to collect the traffic statistics and install the new flow rules.

III. Problem Formulation

In this section, we formulate the optimization problem for the RA process, with the objective to minimize operational costs for the VMNO, while meeting all the end-to-end slice requirements. To formulate the problem, we first need to compute the transmission times in the RAN and CN.

A. Radio Access Network

As mentioned earlier, 5G NR with OFDMA is deployed on the RAN. The bit rate of a user is determined by the number of physical resource blocks (PRBs) allocated to them and the modulation order of the OFDMA symbols. A PRB consists of 12 subcarriers of 15 KHz bandwidth, and its duration is 1 ms. Each of the 12 subcarriers carries 14 OFDMA symbols, and each symbol carries $m = \log_2 M$ bits, where M is the modulation order used in the PRB [9].

The modulation order of each PRB is chosen so that the block error rate (BLER) at the receiver, is less than 10^{-3}. The BLER depends on the received signal to interference and noise ratio (SINR) [9]. In our model, we consider that the transmission power is set to maximum both in downlink and uplink. Thus, the receiver estimates the channel state of the PRB, by detecting the channel state information reference signal (CSI-RS) [9]. Then, the receiver reports back the channel quality indicator (CQI) of the PRB, to the transmitter. Given the CQI, the transmitter chooses the modulation order, based on the predefined modulation and coding scheme (MCS) tables [9]. This process is repeated for all PRBs and all users every T_c seconds. Thus, the modulation order M of each PRB is a known parameter.

[1]By coherence time, we refer to the time period during which the modulation order of each user in the RAN can be assumed to be constant.

Now, we consider that there are at least K available carriers with maximum channel bandwidth $W_{\max}$. Thus, each carrier consists of multiple PRBs with varying modulation orders. To simplify our model, we consider that all the PRBs in the same carrier use the same modulation order. This modulation order is the minimum modulation order among all the PRBs of the carrier. Thus, each carrier is characterized by a single modulation order.

Given the above, if a device is allocated N PRBs of a carrier, where m bits per symbol are transmitted, then the bit rate is $R = 168mN$ Kbps. We also assume that the channel bandwidth W does not need to be a multiple of 180 KHz. Hence, by substituting $N = W/180$, the transmission time for a packet of L bits is $T = L/R = 1.08L/(mW)$ ms.

Next, the BS assigns to each user indexed by k, a single carrier indexed by f. Let m_f^k denote the bits per symbol that user k can transmit on carrier f. Let a_f^k denote the decision variable that indicates whether user k is assigned to carrier f. We consider that each user can use only one carrier out of all the $N > K$ available ones. Also, each carrier can be used at most by one user. Therefore, the following constraints are needed:

$$\mathbf{C_1:} \sum_{f=1}^{N} a_f^k = 1, \quad \forall k \in \{1, 2, ..., K\}, \tag{1}$$

$$\mathbf{C_2:} \sum_{k=1}^{K} a_f^k \leq 1, \quad \forall f \in \{1, 2, ..., N\}. \tag{2}$$

We assume that the BS assigns the same bandwidth W to all users. Since we wish to minimize the effect of queueing delays, we require that each packet transmission is completed before the next one arrives, thus:

$$\mathbf{C_3:} \sum_{f=1}^{N} \frac{a_f^k 1.08L}{m_f^k W} \leq \frac{1}{\lambda}, \quad \forall k \in \{1, 2, ..., K\}. \tag{3}$$

Lastly, the assigned bandwidth must be bounded by $W_{\max}$:

$$\mathbf{C_4:} \ 0 \leq W \leq W_{\max}. \tag{4}$$

B. Core Network

The CN is represented by the directed graph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$. We associate a 3-tuple (c_e, T_e, b_e) for each optical fiber $e \in \mathcal{E}_s$, that corresponds to the fiber's capacity, total delay, and cost respectively. Each optical fiber has c_e available wavelength division multiplexing (WDM) channels with equal bit rates R_e. Hence, the total delay is $T_e = L/R_e + t_e$, where t_e is the propagation delay of edge e.

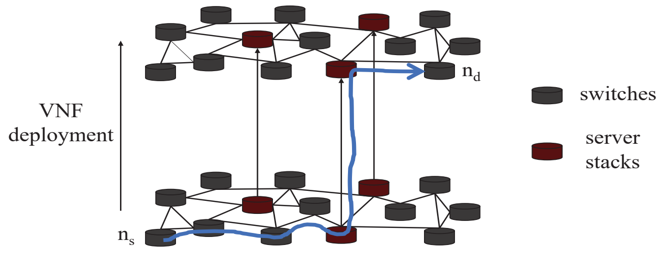

Fig. 1. The multilayered graph $\mathcal{G}_l$ for a single VNF.

We consider that each user can utilize at most one WDM channel in any fiber. Also, the slice users are prioritized in the WDM channels, thus their packets are moved to the front of the queue to avoid qeueuing delays. Given that in each WDM channel, only one user stream can be prioritized, it follows that each WDM channel in an edge can be used at most by one slice user. This prioritization increases the delays for all the other packets in the same queue. Thus, the PIP needs to demotivate any further traffic entering congested edges. This is captured by b_e. The price b_e of edge e is an increasing function of its queue length and capacity. This way, VMNOs are discouraged to steer their traffic through already congested edges that are more expensive to maintain. We note that the queue lengths are available to the PIP and VMNOs, since SDN controllers can collect traffic statistics from the switches.

Similarly for the sever stacks $v \in \mathcal{V}_{ss}$, we denote by c_v the available processing units (PU) at server stack v. Considering that a single VNF can be instantiated at each PU, we similarly associate for each server stack v, a 3-tuple (c_v, T_v, b_v).

As mentioned earlier, we need to decide on which server stacks to deploy the VNFs, then steer all packets through these server stacks as implied by ordered set $\mathcal{V}_r$, while routing them from the BS to the data center. To solve this problem, we construct a multilayered graph as in [5].

First, we create $V_r + 1$ copies of the substrate graph $\mathcal{G}_s$, and denote by $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ the copy i^{th} of graph $\mathcal{G}_s$, where $i \in \{1, ..., V_r + 1\}$. Next, we connect successive copies $\mathcal{G}_i$ and $\mathcal{G}_{i+1}$ by adding edges between the nodes that are copies of the same server stack, for all the server stacks where the i^{th} VNF in $\mathcal{V}_r$ can be deployed. These edges inherit the 3-tuple (c_v, T_v, b_v) of the corresponding server stack. We refer to these edges as processing edges. The last copy $\mathcal{G}_{V_r+1}$ is needed to route the packets from the final VNF to the destination node. We denote the resulting multilayered graph by $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l)$.

Thus, the original SFC-EP is now equivalent to finding a path beginning from the node that represents the BS at the first layer of graph $\mathcal{G}_l$ to the node that represents the destination node at the last layer of graph $\mathcal{G}_l$. We denote these nodes by n_s and n_d respectively. There is a one-to-one relation between the solution space of the SFC-EP and the paths in graph $\mathcal{G}_l$ [8]. All the above are depicted in Fig. 1.

To solve the SFC-EP, we find the set of all paths $\mathcal{P}$, between nodes n_s and n_d in the graph $\mathcal{G}_l$. Finding paths in a graph is computationally demanding. However, this computation can be performed offline for some number of layers, since the network topology does not change. This way we reduce the complexity of the online RAS, by performing the time consuming computations offline.

For each path $p \in \mathcal{P}$, we compute the pair (T_p, b_p), where T_p is the total delay and b_p is the total price of path p. These quantities are the sum of all the delays and all the prices of the edges that compose path p. Now, we denote by x_p^k the decision variable that indicates whether the packets of user k use path $p \in \mathcal{P}$. Then, to route all K packet flows from the n_s to n_d, we have:

$$\mathbf{C_5:} \sum_{p \in \mathcal{P}} x_p^k = 1 \quad \forall k. \tag{5}$$

Let $e^i \in \mathcal{E}_l$ denote the copy of the substrate edge $e \in \mathcal{E}_s$ in the i^{th} layer of graph $\mathcal{G}_l$. We note that all the capacities of edges e^i, $\forall i$, are always equal to c_e since they are copies of the same substrate edge e. Therefore, if any edge e^i, $\forall i$, is utilized, then this implies a reduction to the capacity of every other edge e^j, $\forall j$, that represents the same physical edge e. Also, if a user stream visits two or more copies of the same physical edge, the capacity c_e should not be decreased any further, since a user stream can utilize at most one WDM channel in a fiber as mentioned before. Thus, to correctly count the number of WDM channels utilized by user streams, and bound them by the capacity of the physical edges, the following constraint is required:

$$\mathbf{C_6:} \sum_{k=1}^{K} (1 - \prod_{i=1}^{V_r+1} (1 - \sum_{p \in \mathcal{P} | e^i \in p} x_p^k)) \leq c_e, \quad \forall e \in \mathcal{E}_s. \tag{6}$$

Note that for a given user k, the quantity in the sum is 1 if the user stream k visits edge e at least once, and 0 otherwise. Then, we sum over all users to calculate the total number of used WDM channels in edge e. The same restrictions are needed for the processing edges. Let processing edge $v^i \in E_l$ denote the deployment of the i^{th} VNF on server stack $v \in \mathcal{V}_{ss}$. The resulting constraint is:

$$\mathbf{C_7:} \sum_{k=1}^{K} (1 - \prod_{i=1}^{V_r} (1 - \sum_{p \in \mathcal{P} | v^i \in p} x_p^k)) \leq c_v, \quad \forall v \in \mathcal{V}_{ss}. \tag{7}$$

C. Optimization Problem

Now, we present the end-to-end optimization problem. The total delay of user k is the sum of the delays in the RAN and CN. Consequently, since the end-to-end delay of all users needs be less than T_d, it follows:

$$\mathbf{C_8:} \sum_{f=1}^{N} \frac{a_f^k 1.08 L}{m_f^k W} + \sum_{p \in \mathcal{P}} x_p^k T_p \leq T_d, \quad \forall k. \tag{8}$$

Lastly, we consider a unit cost b_w of normalized bandwidth in the RAN. Therefore, the overall problem is:

$$\begin{aligned} \min_{W, a_f^k, x_p^k} \quad & b_w \frac{KW}{W_{\max}} + \sum_{k=1}^{K} \sum_{p \in P} x_p^k b_p \\ \text{s.t.} \quad & \mathbf{C_1} - \mathbf{C_8} \\ & \mathbf{C_9:} \; a_f^k, \, x_p^k \in \{0, 1\} \quad \forall k, \, \forall f, \, \forall p. \end{aligned} \tag{9}$$

IV. Solution Approach

Given that the MINLP (9) has to be solved within the coherence time to adapt to the wireless channel conditions, we decompose it into two smaller problems, one for the RAN and one for the CN. Then, we provide polynomial time algorithms for each problem.

A. Decomposition

First, we identify that the RA in the RAN and CN are coupled due to constraint $\mathbf{C_8}$. Consequently, decomposition methods for coupling constraints can be used. Many such methods exist in literature [10], [11]. However, these methods do not guarantee the convergence to a globally optimal solution of (9), since the duality gap is positive in most MINLP problems. Therefore, we propose a simpler, yet again sub-optimal, decomposition method. We notice that constraint $\mathbf{C_7}$ is equivalent to the following:

$$\begin{aligned} &\sum_{f=1}^{N} \frac{a_f^k 1.08L}{m_f^k W} \leq \mu^k T_d, \quad \forall k, \\ &\sum_{p\in\mathcal{P}} x_p^k T_p \leq (1-\mu^k) T_d, \quad \forall k, \\ &0 \leq \mu^k \leq 1, \quad \forall k. \end{aligned} \tag{10}$$

In (10), the variables μ^k are also optimization variables. Clearly, the optimal solution of (9) satisfies the constraints in (10) for some optimal vector $\boldsymbol{\mu}^*$. The components of $\boldsymbol{\mu}^*$ determine the percentage of the total delay allowed in the RAN, i.e., they balance the load between the RAN and CN.

Knowing $\boldsymbol{\mu}^*$ would allow us to decompose (9), however it is hard to compute it. Instead, we solve (9) for a particular set of vectors $\boldsymbol{\mu}$. To do so, we first assume that $\mu^k = \mu$, $\forall k$. This assumption expresses our anticipation that the distribution of the load between the RAN and CN primarily depends on the parameters b_w and b_p in the objective function, and less on the channel conditions of each user.

Now, by solving (9) for various values of μ, such as $\{0.1, 0.2, ..., 0.9\}$, the problem decomposes. We note that the problem can be solved for the various values of μ in parallel. Thus, the computational burden is not severe.

B. Radio Access Network Problem

The resulting problem in the RAN is:

$$\begin{aligned} \min_{W, a_f^k} \quad & W \\ \text{s.t.} \quad & \mathbf{C_1} - \mathbf{C_4} \\ & \mathbf{C_{8a}}\text{:} \sum_{f=1}^{N} \frac{a_f^k 1.08L}{m_f^k} \leq W\mu T_d, \quad \forall k, \\ & \mathbf{C_{9a}}\text{:}\ a_f^k \in \{0,1\} \quad \forall k, \forall f. \end{aligned} \tag{11}$$

Now, suppose that the carrier assignment has taken place, and let f_k denote the carrier assigned to user k. Since constraints $\mathbf{C_3}$ and $\mathbf{C_{8a}}$ hold for every user, it follows:

$$W^* = \frac{1.08L}{\min\limits_k m_{f_k}^k} \max\{\lambda, \frac{1}{\mu T_d}\}. \tag{12}$$

From (12), it is clear that to minimize bandwidth W^*, we need to maximize $\min\limits_k m_{f_k}^k$. Equivalently, we need to find the assignment of the K users to the N carriers, such that the minimum modulation order used in the assignment is not smaller than the minimum modulation order of any other possible assignment.

Consequently, the carrier assignment problem in the RAN is an unbalanced version of the Linear Bottleneck Assignment Problem (LBAP) [12]. We complete the transformation by letting $c_{k,f} = 1/m_f^k$, and adding $N-K$ dummy users with $c_{k,f} = 0$, $\forall f$. Now, by solving the LBAP for the resulting $N \times N$ cost matrix, we obtain the optimal assignment variables a_f^k. Then, we compute the optimal bandwidth by (12). Lastly, we mention that the LBAP can be solved using the Threshold Algorithm (TA) with time complexity $\mathcal{O}(N^{2.5}/\sqrt{\log N})$. All the above are summarized in Algorithm 1.

Algorithm 1: Carrier and Bandwidth Assignment

```
Result: Solution of the RAN problem (11);
1 Initialize: $c_{k,f} = 1/m_f^k$; $c_{k,f} = 0, \forall K < k \leq N, \forall f$;
2 For matrix $\mathbf{C}$, solve LBAP using the Threshold Algorithm;
3 Compute optimal bandwidth $W^*$ using (12);
4 return table $\mathbf{A}$ that contains all $a_f^k$ values, and bandwidth W;
```

C. Core Network Problem

For the CN, the problem is:

$$\begin{aligned} \min_{x_p^k} \quad & \sum_{k=1}^{K} \sum_{p\in\mathcal{P}} x_p^k b_p \\ \text{s.t.} \quad & \mathbf{C_5} - \mathbf{C_7} \\ & \mathbf{C_{8b}}\text{:} \sum_{p\in\mathcal{P}} x_p^k T_p \leq (1-\mu) T_d, \quad \forall k, \\ & \mathbf{C_{9b}}\text{:}\ x_p^k \in \{0,1\} \quad \forall p \in \mathcal{P}. \end{aligned} \tag{13}$$

Now, we note that constraint $\mathbf{C_{8b}}$ expresses that the selected paths should have total delay T_p less than $(1-\mu)T_d$. Therefore, we can first find the set of all paths $\mathcal{P}' = \{p \in \mathcal{P} : T_p \leq (1-\mu)T_d\}$. Then, we can solve (13) only over the set $\mathcal{P}'$, and without constraint $\mathbf{C_{8b}}$.

We notice that constraint $\mathbf{C_5}$ implies that we need to select exactly K paths. Also, the parameters b_p and the set of acceptable paths $\mathcal{P}'$, do not depend on the index k . Therefore, the order that paths are assigned to users, does not affect the solution. Consequently, we need to find the cheapest combination of K paths that satisfy constraints $\mathbf{C_6}$ and $\mathbf{C_7}$. These constraints imply that each path reduces the capacity of a physical edge e or server stack v at most by 1.

Given these observations, we propose a greedy algorithm to solve (13). In each iteration, we select the cheapest path p that can be used without violating constraints $\mathbf{C_6}$ and $\mathbf{C_7}$. Then, we assign it to a user. Once all users have been assigned a path, we return table $\mathbf{X}$, which contains the values of all x_p^k variables. This procedure is described in Algorithm 2.

Regarding the time complexity of Algorithm 2, the first for-loop is executed K times and the while-loop is executed at

Algorithm 2: Greedy Path Selection

```
Result: Solution of CN problem (13);
1  Initialize: $x_p^k = 0, \forall k, \forall p$;
2  for $k \in \mathcal{K}$ do
3      assignment = FALSE;
4      while assignment = FALSE do
5          Select first path $p$ from ordered set $\mathcal{P}$;
6          if $T_p > (1-\mu)T_d$ then
7              delete path $p$;
8              continue;
9          end
10         for $(i,j) \in p$ do
11             Find substrate edge $e \in \mathcal{G}_s$ of edge $(i,j) \in \mathcal{G}_l$;
12             if $c_e < 1$ then
13                 delete path $p$;
14                 continue;
15             end
16         end
17         Set $x_p^k = 1$;
18         assignment = TRUE;
19         counted(e) = FALSE, $\forall e \in \mathcal{E}_s$;
20         for $(i,j) \in p$ do
21             Find substrate edge $e \in \mathcal{G}_s$ of edge $(i,j) \in \mathcal{G}_l$;
22             if counted(e) = FALSE then
23                 $c_e = c_e - 1$;
24                 counted(e) = TRUE;
25             end
26         end
27     end
28 end
29 return table $\mathbf{X}$ that contains all $x_p^k$ values;
```

most P times. Moreover, let $l_{\max}$ denote the number of edges of the longest path in graph $\mathcal{G}_l$. Since there is no point in selecting paths with cycles in $\mathcal{G}_l$, then $l_{\max} \leq E_l$. Therefore, the for-loops in steps 7 and 17 run at most E_l times. Thus, the complexity of Algorithm 2, can be described by the loose upper bound $\mathcal{O}(KPE_l)$. Hence, the algorithm scales linearly with respect to the number of users K.

D. Complete Resource Allocation Scheme

The complete RAS is composed of two parts, the offline path discovery and the online RA. The offline algorithm accepts the network topology graph $\mathcal{G}_s$, and produces the multilayered graph $\mathcal{G}_s$ for predetermined numbers of layers. Then, all simple paths p in graph $\mathcal{G}_s$ are obtained and sorted in increasing order of prices b_p, using Yen's algorithm [13]. The offline algorithm is summarized in Algorithm 3. The online RA is summarized in Algorithm 4. As mentioned earlier, the complete RAS is computed at the orchestrator. Its update period is approximately equal to the coherence time and its overall time complexity is $\mathcal{O}(\max\{N^{2.5}/\sqrt{\log N}, KPE_l\})$.

Algorithm 3: Offline Path Discovery

```
Result: Graph $\mathcal{G}_s$ and sorted paths $\mathcal{P}$;
1  Create $V_r + 1$ copies of graph $\mathcal{G}_s$ to obtain $\mathcal{G}_l$;
2  Add edges between succeeding copies of server stacks;
3  Run Yen's Algorithm on graph $\mathcal{G}_l$;
4  return graph $\mathcal{G}_l$, and sorted paths $\mathcal{P}$;
```

Algorithm 4: Online Resource Allocation

```
Result: End-to-end resource allocation of the requested slice;
1  Solve the LBAP of (11) as in Algorithm 1;
2  for $\mu \leftarrow 0.1$ to $0.9$ by $0.1$ do
3      Solve the CN problem (13) using Algorithm 2;
4      Compute optimal bandwidth $W^*$ using 10;
5  end
6  Select the best solution;
7  return table $\mathbf{A}^*$, bandwidth $W^*$, and table $\mathbf{X}^*$;
```

TABLE I
DEFAULT SIMULATION PARAMETERS.

K	500 users	b_w	5
λ	10 pps	V_s	30 switches
L	6 Mb	c_e	$randInt(64, 128)$
T_d	50 ms	R_e	$rand(10, 80)$ Gbps
V_r	3 VNFs	b_e	$10\mathcal{N}(R_e^{'}, R_e^{'}/4)$
W_{max}	50 MHz	μ	$\{0.1, 0.2, ..., 0.9\}$
N	$1.2K$ carriers	V_{ss}	$0.2Vs$ server stacks

V. PERFORMANCE EVALUATION

We test the efficiency of the decomposition method by comparing our RAS to a serial scheme via simulations. The serial scheme decomposes the problem for the largest possible values of μ_k. To do so, the serial scheme uses the same greedy algorithm to solve the CN problem but with the objective to minimize delay. Then, the allowed RAN delays are computed by $T_a^k = T_d - T_c^k$. Next, the RAN problem is solved using the Threshold Algorithm, where the cost matrix is now $c_{k,f} = (1/m_f^k)\max(\lambda, 1/T_a^k)$.

We use the Erdős–Rényi random graph model [14], to generate a strongly connected graph. Although these graphs may not appear frequently in practice, they can be used as worst case examples for algorithm evaluations. To reduce the complexity of the offline algorithm, we find only the 10 000 cheapest paths, as in [15]. Similarly to [5], the default simulation parameters are presented in Table I. Since fibers with higher rates R_e are more expensive in practice, the prices b_e are generated using the Gaussian distribution as shown in Table I, where $R_e^{'}$ is the normalized rate. All the algorithms were implemented using the NetworkX package in Python [16]. Lastly, the simulations were performed on an Intel i7 CPU at 3.8 GHz, with 16 GB of memory.

In Fig. 2, we compare the total cost of the two schemes for varying number of users and b_w. The proposed scheme outperforms the serial one. The gap decreases as b_w increases, since as bandwidth becomes more expensive, the CN delays should be reduced to give more leeway in the RAN. This effect is also depicted in Fig. 3. We note that a large μ may result in infeasibility in the CN, thus increases of b_w do not always increase the optimal value of μ.

In Fig. 4, we compare the schemes for various delay requirements. Again, the proposed scheme outperforms the serial one. In Fig. 5, we compute the average running time of 20 executions of each scheme for varying number of users. Clearly, both schemes are fast and scalable. The RA

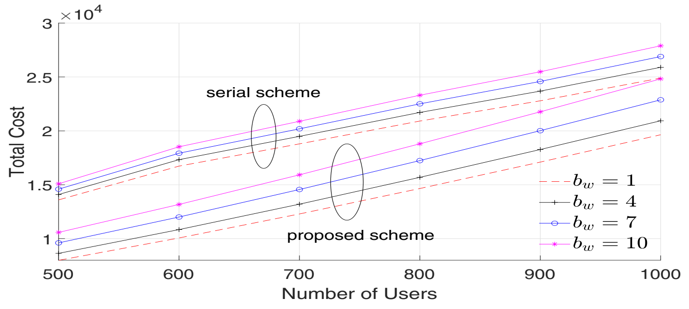

Fig. 2. Effect of users and b_w on cost function.

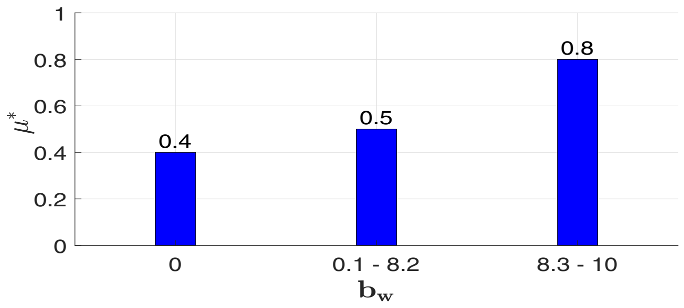

Fig. 3. Effect of b_w on μ^*.

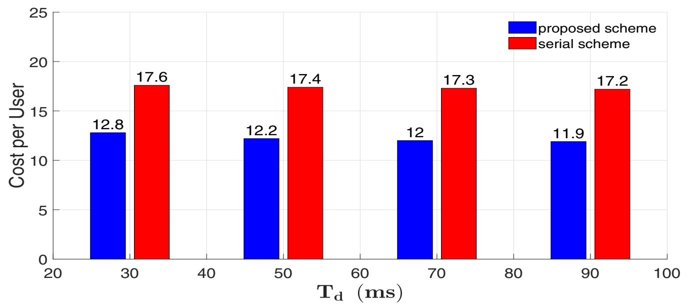

Fig. 4. Effect of delay T_d on cost function.

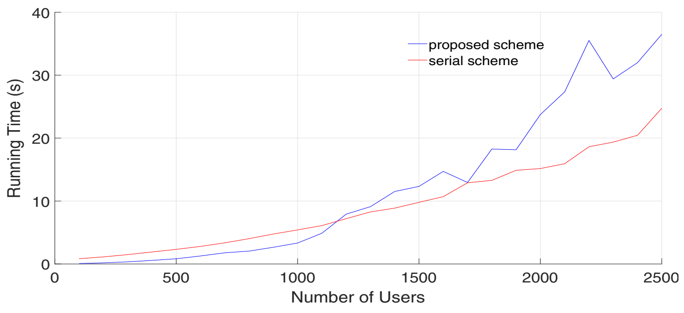

Fig. 5. Effect of users on average running time.

is completed in less than 1 second for 100 users and less than 45 seconds for 2500 users. As expected, the serial scheme is faster, since it solves the overall problem once. We note that the proposed scheme's loops were not parallelized, thus the running time can be reduced even more.

VI. Concluding Remarks

In this paper, we proposed a complete RAS for network slices, that provides end-to-end guarantees, by including the RAN. We solved the overall problem by decomposing it into its RAN and CN components. The former was solved optimally, and for the latter, we followed a greedy approach. The end-to-end RAS has $\mathcal{O}(\max\{N^{2.5}/\sqrt{\log N}, KPE_l\})$ time complexity. Lastly, the simulation results verified that indeed the RAS is very fast and scalable with respect to the users.

We note that our scheme provides end-to-end guarantees, by preventing queueing delays. However, better resource utilization can be achieved, if queueing delays are allowed to some extent. Thus, various models of queuing theory could be useful in network slicing. Also, in practice, perfect channel state information and traffic statistics are not readily available. Therefore, leveraging SDN to quickly adapt in stochastic settings is a direction worth exploring.

References

[1] P. Rost, C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega, D. Aziz, and H. Bakker, "Network slicing to enable scalability and flexibility in 5g mobile networks," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 72–79, 2017.

[2] J. Gil Herrera and J. F. Botero, "Resource allocation in nfv: A comprehensive survey," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 518–532, 2016.

[3] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and openflow: From concept to implementation," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 2181–2206, 2014.

[4] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, p. 17–29, Mar. 2008. [Online]. Available: https://doi.org/10.1145/1355734.1355737

[5] Z. Cao, S. S. Panwar, M. Kodialam, and T. V. Lakshman, "Enhancing mobile networks with software defined networking and cloud computing," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1431–1444, 2017.

[6] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *2015 IEEE Conf. Comput. Commun. (INFOCOM)*, 2015, pp. 1346–1354.

[7] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *2015 11th Int. Conf. Netw. Service Manag. (CNSM)*, 2015, pp. 50–56.

[8] J. Zhang, A. Sinha, J. Llorca, A. Tulino, and E. Modiano, "Optimal control of distributed computing networks with mixed-cast traffic flows," in *IEEE INFOCOM 2018 - IEEE Conf. Comput. Commun.*, 2018, pp. 1880–1888.

[9] S. Ahmadi, *5G NR: Architecture, Technology, Implementation, and Operation of 3GPP New Radio Standards*. Academic Press, 2019.

[10] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, p. 1–122, Jan. 2011. [Online]. Available: https://doi.org/10.1561/2200000016

[11] A. Conejo, E. Castillo, R. Mínguez, and R. García-Bertrand, *Decomposition Techniques in Mathematical Programming: Engineering and Science Applications*. Springer, 2006.

[12] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*. USA: SIAM, 2009.

[13] J. Y. Yen, "Finding the k shortest loopless paths in a network," *Manag. Sci.*, vol. 17, no. 11, pp. 712–716, 1971. [Online]. Available: https://doi.org/10.1287/mnsc.17.11.712

[14] P. Erdős and A. Rényi, "On random graphs i," *Publ Math Debrecen*, vol. 6, pp. 290–297, 1959.

[15] N. Torkzaban and J. S. Baras, "Trust-aware service function chain embedding: A path-based approach," in *2020 IEEE Conf. NFV-SDN*, 2020, pp. 31–36.

[16] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proc. 7th Python Sci. Conf.*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11 – 15.