# A Nested Decomposition Model for Reliable NFV 5G Network Slicing

Huy Duong and Brigitte Jaumard
Computer Science and Software Engineering, Concordia University
Montreal, QC, Canada
bjaumard@cse.concordia.ca

## ABSTRACT

With the 5th generation of mobile networking (5G) on our doorstep, optical network operators are reorganizing their network infrastructures so that they can deploy different topologies on the same physical infrastructure on demand. This new paradigm, called network slicing, together with network function virtualization (NFV), can be enabled by segmenting the physical resources based on the requirements of the application level.

In this paper, we investigate a nested decomposition scheme for the design of reliable 5G network slicing. It involves revisiting and improving the previously proposed column generation models, and adding in particular the computation of dual bounds with Lagrangian relaxation in order to assess the accuracy of the solutions.

Extensive computational results show that we can get $\varepsilon$-optimal reliable 5G slicing solutions with small $\varepsilon$ (about 1% on average) in fairly reasonable computational times.

## 1 INTRODUCTION

The 5th generation of mobile networking (5G) is based on the key technologies of Software-Defined Networking (SDN) and Network Function Virtualization (NFV) in order to offer multiple services with various performance requirements, e.g., low latency, high throughput, high reliability, or high security. SDN allows network operators to remotely (re)configure the physical network in order to reserve on demand networking resources. Virtual compute nodes (i.e., node with computing resources such as servers or a data center) can enable Virtual Network Functions (VNFs) running on top of general-purpose hardware, such as a cloud infrastructure.

Within the context of 5G networks, network slicing is an end-to-end logical network provisioned with a set of isolated virtual resources on a shared physical infrastructure. Slices are provided as different customized services to fulfill dynamic demands, with flexible resource allocations. In other words, a network slice is a self-contained network with its own virtual resources, topology, traffic flow, and provisioning rules. Network slicing is therefore a key feature of 5G networks, which allows the efficient resource share of a common physical infrastructure and consequently, reduces operators' network construction costs.

An interesting feature of SDN is its ability to process traffic while forwarding it, using "network functions" or "network services". The latter ones can implement header processing and payload processing functions, such as network address translation (NAT), firewall, or domain name system (DNS). They are called Virtual Network Functions (VNFs) and can be implemented in software on conventional processing systems (e.g., servers or data centers) that are co-located with networking equipment. The sequence of functions that need to be set up for a specific flow is referred to as a "service chain."

In this paper, we propose a 5G network slicing design model and algorithm, based on nested column generation. It aims at maximizing the number of granted slices while addressing the reliability requirements of network slices. In order to avoid the costly exact solutions of the sub-problems, we discuss how to compute bounds using Lagrangian relaxation, so that we can assess the accuracy of the output solutions.

The paper is organized as follows. Section 2 contains the literature review. Section 3 provides the detailed problem statement of the design of reliable 5G network slicing. An original nested decomposition model is proposed in Section 4. Algorithmic aspects are covered in Section 5. Numerical results are described in Section 6 and conclusions are drawn in the last section.

## 2 LITERATURE REVIEW

### 2.1 5G Network Slicing

Several papers and surveys have already appeared on 5G network slicing and described their various challenges and opportunities [1, 14]. Similarly, many studies and several surveys have been devoted to Network Function Virtualization (NFV), e.g., [20].

Very few studies look at the combination of reliable 5G slicing and NFV. Tang *et al.* [18] propose an MILP for 5G network slicing that maximizes the number of granted slices while minimizing their failure rate, without providing protection mechanisms.

Some authors looked at network slicing and NFV, more often in the wireless networks than in the wired optical ones. Challenges are discussed in, e.g., [10, 14].

Lin *et al.* [11] propose an exact algorithm using column generation aiming to minimize the total embedding cost in terms of spectrum cost and computation cost for a single virtual network request. Moreover, validation of the exact algorithm is made on a six node network. Large data instances are solved using a heuristic. Destounis *et al.* [3] also propose an exact column generation algorithm for network slicing without the NSF features. They solved data instances up to 200 nodes. Carella *et al.* [2] exemplified Network slicing as an addition to the current Cloud architecture and evaluated on a testbed architecture based on the Fraunhofer FOKUS and TU Berlinopen source Open Baton toolkit.

### 2.2 Nested Column Generation Decomposition

The idea of nested column generation is not new: several authors have already investigated it for various problems, e.g., Song [17] in logistics, Dohn and Mason [4] for staff rostering, Karabuk [8] for scheduling paratransit vehicles and Vanderbeck [19] for two-dimension cutting-stock.

However, most studies did not worry about assessing accurately the quality of the output solutions, except, e.g., [6, 19].

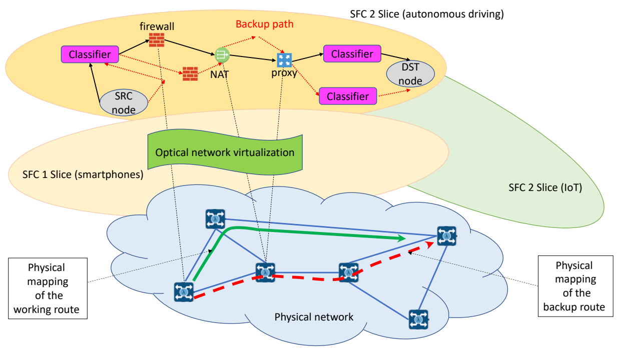

SFC 2 Slice (autonomous driving)
firewall
Backup path
Classifier
NAT
proxy
Classifier
DST node
Classifier
SRC node
Optical network virtualization
SFC 1 Slice (smartphones)
SFC 2 Slice (IoT)
Physical mapping of the working route
Physical network
Physical mapping of the backup route

**Figure 1: 5G Reliable Slicing**

## 3 PROBLEM STATEMENT AND NOTATIONS

### 3.1 Rel_5G_NFV Problem Statement

Consider a physical network $G^{\mathrm{P}}$ and a set $K$ of connections, indexed by $k$. The Reliable 5G NFV Network Slicing (Rel_5G_NFV) problem consists of embedding/mapping the maximum number of slices onto the physical network while ensuring each slice is individually protected against any single link failure. We assume each slice is associated with a given application, that is characterized with the use of a single service function chain.

### 3.2 Notations

**Physical Network.** The physical network $G^{\mathrm{P}} = (V^{\mathrm{P}}, L^{\mathrm{P}})$ is defined by its set of nodes $V^{\mathrm{P}}$, indexed by $v$, set of links $\ell \in L^{\mathrm{P}}$, with capacities $\textsc{cap}_v \geq 0$ and $\textsc{cap}_\ell \geq 0$ on both nodes and links, respectively.

**5G Slicing.** Each slice $S \in \mathcal{S}$ is associated with a virtual network $S = (V^S, L^S, \textsc{cap}^S)$, which is defined by a set of virtual nodes $V^S$ (indexed by $v'$), and virtual links $L^S$ (indexed by $\ell'$), with capacity requirements $\textsc{cap}^S_{v'}$ and $\textsc{cap}^S_{\ell'}$, respectively.

**Virtual Networks.** An embedding of $S$ onto $G^{\mathrm{P}}$ consists of mapping:

- Each virtual node $v' \in V^S$ onto a physical node $v \in V^{\mathrm{P}}$
- Each virtual link $\ell'$ onto a loop-free physical path, connecting two physical nodes $u$ and $v$, to which the virtual nodes $u'$ and $v'$ have been mapped
- Each virtual "path" is protected by a virtual path, whose mapping is physical link-disjoint from the mapping of the first path,

in order to maximize the GoS.

A feasible embedding is an embedding in which all physical link and node capacity constraints are satisfied; that is, the sum of capacity demands of all virtual nodes embedded on a physical node is less than the capacity of this physical node, and the sum of the requests of all the virtual links going through a physical link does not exceed the capacity of this link.

In order to simplify the model and the algorithm, we work directly with the mapping of the virtual nodes/links, i.e., with physical nodes/links, without expressing explicitly the virtual links and nodes.

**Service Function Chaining.** Let $F$ be the set of all services functions, indexed by $f$, and let $C$ be the set of all service function chains, indexed by $c$. Any chain $c$ is defined by an ordered sequence of $n_c$ functions: $c = \{f_0, f_1, .., f_{n_c - 1}\}$. The routing of any demand in a slice governed by SFC $c$ must go through virtual compute nodes hosting the functions of $c$.

**Application (Slice) Demand.** Demands are provided for each slice $S$, with each slice being associated with one particular application, characterized by a given Service Function Chain (SFC) $c_S$. We denote by $K^{sd,c_S}$ the demand for node pair $(v_s, v_d) \in \mathcal{SD}_{c_S}$, i.e., with traffic in slice $S$, subject to the requirement of SFC $c_S$, and by $\Delta^{sd,c}_{f_i}$ the required computational resource of function $f_i$ for demand $K^{sd,c_S}$.

## 4 A NESTED DECOMPOSITION SCHEME

We now present a nested decomposition scheme, in which at the upper layer of the decomposition, we select the slice configurations for each slice demand. Each slice configuration is defined by a virtual network as defined in Section 3.2, which satisfies the demand $K^{c_S}$ associated with its required application and corresponding SFC $c_S$.

Let $\Gamma$, indexed by $\gamma$, be set of all possible slice configurations. Each slice configuration $\gamma$ is characterized by a slice $S$ and its assigned resources. Each slice configuration $\gamma$ is characterized by its slice index $S$, its node assigned resources $R^\gamma_v$, and its link assigned resources $B^\gamma_\ell$. We have $\Gamma = \bigcup\limits_{c \in C} \Gamma_{c_S}$.

In order to simplify the notations, we will simply write $c$ unless there is confusion.

### 4.1 Master Problem

Master problem maximizes the grade of service (GoS) subject to capacity constraints. It requires only one set of variables: $z_\gamma = 1$ if potential slice virtual network $\gamma$ associated with $c$ is selected, 0 otherwise, for $\gamma \in \Gamma_c$ and $c \in C$.

**Objective**:

$$\max \sum_{c \in C} \sum_{\gamma \in \Gamma_c} \sum_{(s,d) \in \mathcal{SD}_c} K^{sd,c} z_\gamma \tag{1}$$

subject to:

$$\sum_{\gamma \in \Gamma_c} z_\gamma \leq 1 \qquad c \in C \tag{2}$$

$$\sum_{c \in C} \sum_{\gamma \in \Gamma_c} R^\gamma_v z_\gamma \leq \textsc{cap}_v \qquad v \in V^{\mathrm{P}} \tag{3}$$

$$\sum_{c \in C} \sum_{\gamma \in \Gamma_c} B^\gamma_\ell z_\gamma \leq \textsc{cap}_\ell \qquad \ell \in L^{\mathrm{P}} \tag{4}$$

$$z_\gamma \in \{0, 1\} \qquad \gamma \in \Gamma \tag{5}$$

Constraints (2) impose to select at most one virtual network (slice) for demand associated with $c \in C$. Constraints (3) enforce the compute node capabilities, while constraints (4) enforce the link transport capacities.

### 4.2 Slicing Pricing Problem ($\mathrm{PP}_{\textsc{slice}}$)

In order to be able to compute the required node and link resource for a given slice, the pricing problem, or equivalently, the slice configuration generator, needs to provision the demand $K^c$. We define the following parameters.

**Parameters:**

- $\pi \in \Pi$: a logical path that defines a service path with chain $c$ from $s$ to $d$. Note that a logical path may go through a given physical link several times due to the sequence of functions in $c$.
- $\Pi^c_{sd} \subseteq \Pi$: set of all potential paths for service chain $c$ from $s$ to $d$.

- $a_v^{i,\pi} = 1$ if, on path $\pi$, function $f_i$ is hosted on physical node $v$, 0 otherwise.
- $\delta_\ell^\pi$ = number of times path $\pi$ goes through link $\ell$
- $x_\ell^\pi = 1$ if logical path $\pi$ goes through physical link $\ell$ at least once, 0 otherwise.

**Variables:**

- $y_{\pi,p}^{sd,c} = 1$ if path $\pi$ is the primary path to provision traffic from $s$ to $d$, 0 otherwise.
- $y_{\pi,b}^{sd,c} = 1$ if path $\pi$ is the backup path to provision traffic from $s$ to $d$, 0 otherwise.

**Objective:**

$$\max \quad \mathrm{RC}_{\mathrm{PP_{SLICE}}} = \sum_{(s,d)\in\mathcal{SD}} K^{sd,c} - u_c^{(2)}$$
$$- \sum_{v\in V^{\mathrm{P}}} u_v^{(3)} \sum_{(s,d)\in\mathcal{SD}} \sum_{i=0}^{n_c-1} \sum_{\pi\in\Pi_{sd}^c} \Delta_{f_i}^{sd} a_v^{i,\pi}(y_{\pi,p}^{sd,c} + y_{\pi,b}^{sd,c})$$
$$- \sum_{\ell\in L^{\mathrm{P}}} u_\ell^{(4)} \sum_{(s,d)\in\mathcal{SD}} \sum_{\pi\in\Pi_{sd}^c} K^{sd,c}\delta_\ell^\pi(y_{\pi,p}^{sd,c} + y_{\pi,b}^{sd,c}) \quad (6)$$

**Constraints:**

One primary path per demand:

$$\sum_{\pi\in\Pi_{sd}^c} y_{\pi,p}^{sd,c} = 1 \qquad (v_s, v_d) \in \mathcal{SD} \qquad (7)$$

One backup path per demand:

$$\sum_{\pi\in\Pi_{sd}^c} y_{\pi,b}^{sd,c} = 1 \qquad (v_s, v_d) \in \mathcal{SD}. \qquad (8)$$

Link disjoint primary and backup paths:

$$\sum_{\pi\in\Pi_{sd}^c} x_\ell^\pi(y_{\pi,p}^{sd,c} + y_{\pi,b}^{sd,c}) \le 1 \qquad (v_s, v_d) \in \mathcal{SD}, \ell \in L^{\mathrm{P}}. \quad (9)$$

Link and node capacities:

$$(R_v =) \quad \sum_{(v_s,v_d)\in\mathcal{SD}} \sum_{i=0}^{n_c-1} \sum_{\pi\in\Pi_{sd}^c} \Delta_{f_i}^{sd} a_v^{i,\pi}(y_{\pi,p}^{sd,c} + y_{\pi,b}^{sd,c})$$
$$\le \mathrm{CAP}_v \qquad v \in V^{\mathrm{P}} \quad (10)$$
$$(B_\ell =) \quad \sum_{(v_s,v_d)\in\mathcal{SD}} \sum_{\pi\in\Pi_{sd}^c} K^{sd,c}\delta_\ell^\pi\, (y_{\pi,p}^{sd,c} + y_{\pi,b}^{sd,c})$$
$$\le \mathrm{CAP}_\ell \qquad \ell \in L^{\mathrm{P}}. \quad (11)$$

## 4.3 Path Pricing Problem (PP$_{sd}$): Service path for Demand from $v_s$ to $v_d$

For a given $(v_s, v_d) \in \mathcal{SD}$, we look for the generation of a path $\pi$ from $v_s$ to $v_d$, which can improve the linear programming relaxation of PP$_{\mathrm{SLICE}}$.

**Variables:**

- $x_\ell^\pi = 1$ if path $\pi$ uses $\ell$, 0 otherwise.
- $\delta_\ell^\pi$ = number of times path $\pi$ goes through $\ell$.
- $\varphi_\ell^{sd,c,i} = 1$ if, for service chain $c$, the path from $v_s$ to $v_d$ uses link $\ell$ to go from the location of function $f_{i-1}$ to the location of function $f_i$, 0 otherwise. Note that, when $i = 0$, $\varphi_\ell^{sd,c,i}$ represents the path from the source to the first function, when $i = n_c$, it is the path from the last function to the destination.
- $a_v^i = 1$ if the $i$th function ($f_i$) of chain $c$ is installed on node $v$, 0 otherwise.

**Objective:**

$$\max \quad \Big( - \sum_{v\in V^{\mathrm{P}}} u_v^{(3)} \sum_{i=0}^{n_c-1} \Delta_{f_i}^{sd} a_v^{i,\pi} - \sum_{\ell\in L^{\mathrm{P}}} u_\ell^{(4)} K^{sd,c}\delta_\ell^\pi \Big)$$
$$- u_{sd,p}^{(7)} - \sum_{\ell\in L^{\mathrm{P}}} x_l^\pi u_{sd}^{(9)}$$
$$- \sum_{v\in V} \sum_{i=0}^{n_c-1} \Delta_{f_i}^{sd} a_v^i u_v^{(10)} - \sum_{\ell\in L} u_\ell^{(11)} K^{sd,c}\delta_\ell^\pi \quad (12)$$

**Constraints:**

Aggregation of link usage:

$$\delta_\ell^\pi = \sum_{i=0}^{n_c} \varphi_\ell^{sd,c,i} \qquad \ell \in L^{\mathrm{P}}. \qquad (13)$$

Multiple usage of a link:

$$\varphi_\ell^i \le x_\ell^\pi \qquad \ell \in L^{\mathrm{P}}, i \in 0,..,n_c - 1. \qquad (14)$$

This set of constraints ensures that $x_\ell$ keeps track of physical link $\ell$ if it is used by any logical link. Indeed, a link can be used multiple times by a given path, this set of constraints result $x_\ell$ as used links, no matter how many times they are used. These variables play the role in the upper pricing where backup path and primary path must be disjoint.

Flow Conservation constraints

$$\sum_{\ell\in\omega^+(v)} \varphi_\ell^{sd,c,0} - \sum_{\ell\in\omega^- v} \varphi_\ell^{sd,c,0} + a_v^{sd,c,0}$$
$$= \begin{cases} 1 \text{ if } v = v_s \\ 0 \text{ else} \end{cases} \qquad v \in V^{\mathrm{P}} \qquad (15)$$
$$\sum_{\ell\in\omega^+(v)} \varphi_\ell^{sd,c,n_c} - \sum_{l\in w^- v} \varphi_\ell^{sd,c,n_c} - a_v^{sd,c,n_c-1}$$
$$= \begin{cases} -1 & \text{if } v = v_d \\ 0 & \text{else} \end{cases} \qquad v \in V^{\mathrm{P}} \qquad (16)$$
$$\sum_{\ell\in\omega^+(v)} \varphi_\ell^{sd,c,i} - \sum_{\ell\in\omega^-(v)} \varphi_\ell^{sd,c,i} + a_v^{sd,c,i} - a_v^{sd,c,i-1} = 0$$
$$v \in V^{\mathrm{P}}, 0 < i < n_c. \qquad (17)$$

Constraints (15) ensure that demand starts at the source node, then is transferred through a path to the location of first function (unless first function is located at the source node). Similarly, constraints (16) make sure that the demand is delivered to the destination after it is processed by the last function (unless the last function is installed at the destination node). From the location of function $i - 1$ to the location of function $i$, constraints (17) define a path to connect them.

We next use constraints to eliminate the ineffective solutions and, as a consequence, those constraints help to improve the quality of the columns, i.e., slice configurations.

A unique node location for each function occurrence in the service chain:

$$\sum_{v\in V^{\mathrm{P}}} a_v^i = 1 \qquad i = 0, 1, \ldots, n_c. \qquad (18)$$

If a link is not used, its corresponding $x_\ell$ can be set to zero:

$$x_\ell \le \delta_\ell^\pi \qquad \ell \in L^{\mathrm{P}} \qquad (19)$$

Domain constraints:

$$x_\ell^\pi, \varphi_\ell^\pi, a_v^i \in \{0, 1\}; \qquad \delta_\ell^\pi \in \mathbb{Z}^+ \qquad (20)$$

Node capacity constraints:

$$\sum_{i=0}^{n_c-1} \Delta_{f_i} a_v^i \leq \text{CAP}_v \qquad v \in V^{\text{P}} \tag{21}$$

Link capacity constraints:

$$\delta_\ell^\pi K^{sd,c} \leq \text{CAP}_\ell \qquad \ell \in L^{\text{P}}. \tag{22}$$

We will discuss in the next section how to solve efficiently the path pricing problems, without requiring the solution of ILP programs at each iteration of the column generation algorithm.

## 5 NESTED COLUMN GENERATION ALGORITHM

Column generation [9] is based on the fact that, in the simplex method, the solver does not need to simultaneously access all variables of the problem. In fact, a solver can start working only with the basis (a particular subset of the constrained variables), then use a reduced cost to choose the other variables to access, as needed. It is today a very well known and powerful technique [5, 12], while column generation modeling remains an art when the decomposition is not deduced from the application of the Dantzig-Wolfe decomposition.

We next provide the details of our nested column generation algorithm and how we estimated the accuracies of the resulting solutions.

### 5.1 Nested CG and ILP Solution

The conceptual column generation scheme alternates between solving a restriction of the original problem, usually called restricted master problem, and a column generation phase which is used to augment the set of variables/columns of the restricted master problem using a so-called pricing problem. Here, the pricing problem can be decomposed into $|\mathcal{S}|$ slice pricing subproblems.

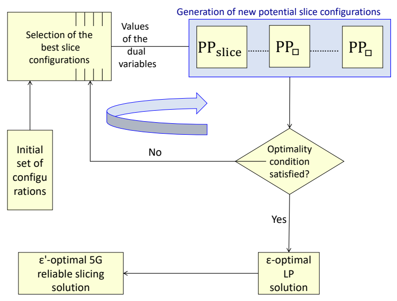
In order to guarantee reaching an optimal LP solution, it is required to solve at least once the pricing problem. In this study, we propose to solve the slice pricing problem, indeed, the slicing pricing subproblems using again a column generation algorithm. As these last subproblems are Integer Linear Programs (ILPs), and as we did not develop any branch-and-price algorithms to solve them, they are never solved optimally, and therefore we need to derive a linear relaxation bound in order to get upper bounds, see next section for the details.

In any case, at both decomposition levels, we use the column generation algorithm as long as we can derive new improving columns. For integer solutions, when we cannot improve anymore the LP solution, we use an ILP solver on the current constraint matrix, i.e., the constraint matrix made of all the columns generated so far, and deduce an ILP solution.

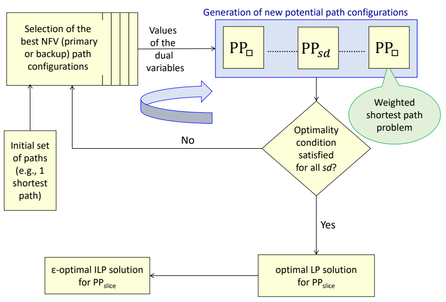
Flowcharts in Figure 2 summarize the algorithm. Accuracy of the output solutions is assessed with $\varepsilon$, which is defined as follows:

$$\varepsilon = \frac{\overline{z}_{\text{LP}} - \tilde{z}_{\text{ILP}}}{\tilde{z}_{\text{LP}}},$$

where $\overline{z}_{\text{LP}}$ is an upper bound the LP solution of problem (1)-(5), whose calculation is developed in Section 5.2. $\tilde{z}_{\text{ILP}}$ is the best found ILP solution (hence a lower bound on the ILP solution), as derived by the solution of the ILP solver on the constraint matrix of (1)-(5) when no more improved column can be generated by the solution of the slice pricing problem (6)-(11).


Generation of new potential slice configurations
Selection of the best slice configurations
Values of the dual variables
$\text{PP}_{\text{slice}}$
$\text{PP}_\square$
$\text{PP}_\square$
Initial set of configu rations
No
Optimality condition satisfied?
Yes
ε'-optimal 5G reliable slicing solution
ε-optimal LP solution

(a) Upper level flowchart


Generation of new potential path configurations
Selection of the best NFV (primary or backup) path configurations
Values of the dual variables
$\text{PP}_\square$
$\text{PP}_{sd}$
$\text{PP}_\square$
Weighted shortest path problem
Initial set of paths (e.g., 1 shortest path)
No
Optimality condition satisfied for all sd?
Yes
ε-optimal ILP solution for $\text{PP}_{\text{slice}}$
optimal LP solution for $\text{PP}_{\text{slice}}$

(b) Lower level flowchart

**Figure 2: Flowcharts**

In order to speed-up the solution of the path pricing subproblems, we first use a shortest path algorithm after noting that all the link costs are positive, taking into account the values of the dual variables. It is worth noting that the usage of a shortest path algorithm does not necessarily guarantee the generation of feasible lightpaths with respect to link and node capacities. However, those capacities are enforced in the slice pricing subproblems, and therefore taken care. When the path pricing subproblems are not able to generate improving paths (i.e., with a negative reduced cost), then we use an ILP solver to solve them, with the guarantee to satisfy all node and link capacities.

### 5.2 Solution Accuracy

The nested column generation framework allows the efficient exploitation of the substructures of a problem at the expense of a more difficult exact solution of the linear programming relaxation as it a priori requires the exact solution of the upper level pricing problem (here the slice $\text{PP}_{\text{SLICE}}$ pricing problem), i.e., a branch-and-price algorithm. In order to overcome that difficulty, we propose to compute an upper bound on the objective (i.e., reduced cost) of the $\text{PP}_{\text{SLICE}}$ problem, and then deduce an upper bound on the optimal LP solution of the Rel_5G_NFV master problem (1)-(5). It then allows the evaluation of the accuracy (gap) of output ILP solutions using the algorithm described in the previous section.

Consider the compact formulation associated with (1)-(5), i.e., the COMPACT model such that when applying a Dantzig-Wolfe decomposition to it, we derive model (1)-(5). Let

$$[\text{COMPACT}] \qquad \max\{cx : Ax \leq b, x \in X\}.$$

Using the Dantzig-Wolfe decomposition of Model COMPACT, the slicing pricing problem, $\text{PP}_{\text{SLICE}}$, can be written as follows:

$$\text{RC}^{\star}_{\text{PP}_{\text{SLICE}}} = \max \left\{ \overline{c}\, x : x \in X^{\text{PRICING}} \right\}. \qquad (23)$$

We simply write RC to shorten $\text{RC}_{\text{PP}_{\text{SLICE}}}$ when there is no ambiguity so that $\text{RC}^{\star}_{\text{PP}_{\text{SLICE}}} = \text{RC}^{\star}$.

In Figure 3, we rank the relative positions of the various values that we discuss below. Question marks indicate values that are not computed accurately, and that are upper/lower bounded.

The Lagrangian relaxation of the COMPACT Model can be written:

$$\text{LR}(u) = \max_{x \in X} \left\{ L(u, x) = ub + \underbrace{(c - uA)x}_{\text{RC}(u,x)} \right\}. \qquad (24)$$

Following Vanderbeck [19] and Pessoa *et al.* [15], a valid upper bound for the COMPACT problem can be computed using Lagrangian Relaxation (LR). At any iteration $\tau$ of the column generation algorithm, i.e., when we re-optimize the linear relaxation of the master problem (1)-(5), the optimal $x_{\text{RC}^{\star}}$ that maximizes $L(u_\tau, x_{\text{RC}^{\star}})$ can be written:

$$\begin{aligned} x_{\text{RC}^{\star}} &= \arg\max_{x \in X} L(u_\tau, x) \quad = \arg\max_{x \in X} \text{RC}(u_\tau, x) \\ &= \arg\max_{i \in I} \text{RC}(u_\tau, x^i) = \arg \max_{x \in X^{\text{PRICING}}} \text{RC}(u_\tau, x), \end{aligned}$$

where $x^i, i \in I$ denote the extreme points of $X$, see [13], Section II.3.6.

As $x_{\text{RC}^{\star}}$ is known only if we solve $\text{PP}_{\text{SLICE}}$ exactly, we can bound it in order to get an upper bound, $\overline{z}_{\text{LP}}$, on the optimal value of the linear programming relaxation. Indeed, $\text{RC}^{\star,\tau}_{\text{ILP}} \leq \text{RC}^{\star,\tau}_{\text{LP}}$, where $\text{RC}^{\star,\tau}_{\text{LP}}$ is the optimal value of the LP relaxation of $\text{PP}_{\text{SLICE}}$ at iteration $\tau$ of the column generation algorithm.

Consequently, $L(u_\tau, x_{\text{RC}^{\star}}) = u_\tau b + \text{RC}^{\star,\tau}_{\text{ILP}} \leq u_\tau b + \text{RC}^{\tau}_{\text{LP}} = \overline{z}^{\tau}_{\text{LP}}$.

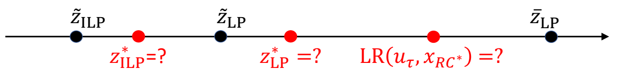


**Figure 3: Ranking of the various LP, LR and ILP values.**

At each iteration $\tau$ of the column generation algorithm, each pricing problem is decomposed into $|\mathcal{S}|$ elementary slice pricing problems of the type $\text{PP}_{\text{SLICE}}$. It implies:

$$\overline{z}^{\tau}_{\text{LP}} = u_\tau b + \sum_{S \in \mathcal{S}} \text{RC}^{\star}_{\text{LP}}(\text{PP}_{\text{SLICE}}(S)).$$

Note that the Lagrangian relaxation upper bound does not improve monotonically [15], thus, in order to derive the best possible upper bound, the algorithm must compute

$$\overline{z}_{\text{LP}} = \min_{\tau} \overline{z}^{\tau}_{\text{LP}} = \min_{\tau} \max_{S \in \mathcal{S}} \text{RC}^{\star,\tau}_{\text{LP}}(\text{PP}_{\text{SLICE}}(S)).$$

It remains possible to add several columns (i.e., slices) at a time (whose $\widetilde{\text{RC}}^{\tau}_{\text{ILP}}(\text{PP}_{\text{SLICE}}(S)) > 0$) to the master problem (1)-(5) in one iteration, as long as they are generated with the same set of dual values. Note that output ILP solutions of $\text{PP}_{\text{SLICE}}(S)$ are not guaranteed to be optimal, hence the notation $\widetilde{\text{RC}}$ to denote a heuristic solution of the slice pricing problem. Indeed, the algorithm has to go through all slice subproblems in each iteration to ensure the correctness of the Lagrangian bound.

## 6 NUMERICAL RESULTS

We implemented the model and algorithm described in the previous sections with a C++ program on a Linux computer with 773727 MB RAM and Intel Xeon E5-2687W v3 @ 3.10 GHz 2 processors, 20 cores. We first describe the data sets, and then we report on the performance of the algorithm.

### 6.1 Data Sets

We considered two topologies from SNDLib [16] and their characteristics are described in Table 1. We re-use the traffic matrix of [7] with four SFCs. In order to derive slice demand, for each original SFC in [7], we divided the overall traffic in 4 subsets, resulting into traffic demands for 16 slices. Transport capacities were set with the optimal solution when allowing only one NFV node.

**Table 1: Data sets**

| Topologies | # nodes | # links | # connections per slice | # slices | Offered load |
|---|---|---|---|---|---|
| INTERNET2 | 10 | 34 | 90 | 16 | 1Tb |
| ATLANTA | 15 | 44 | 210 | 16 | 1Tb |

### 6.2 Model and Algorithm Efficiency and Accuracy

We conducted experiments with the same link transport capacities, and increased node capacities as we increase the number of NFV (compute) nodes. Corresponding accuracies and computational times (seconds) are reported in Table 2. We observe that resulting accuracies are less than 3% except for 4 cases where the gap can reach up to 5.6%. Data Instances are easier to solve as the number of NFVs is increasing, and computational times are fairly reasonable taking inot account the accuracies and the complexity of the design problem of reliable 5G network slicing.

**Table 2: Nested CG performance**

| # NFV | INTERNET2 | | ATLANTA | |
|---|---|---|---|---|
| nodes | gap (%) | CPU | gap (%) | CPU |
| 1 | 3.8 | 454.9 | 5.6 | 991.8 |
| 2 | 3.4 | 574.2 | 4.3 | 4,215.9 |
| 3 | 0.4 | 89.4 | 2.9 | 1,040.2 |
| 4 | 0.4 | 65.7 | 2.9 | 1,010.1 |
| 5 | 0.4 | 89.9 | 2.9 | 578.3 |
| 6 | 0.4 | 36.3 | 2.9 | 582.9 |
| 7 | 0.4 | 36.6 | 0.0 | 651.5 |
| 8 | 0.1 | 158.8 | 2.9 | 758.5 |
| 9 | 0.1 | 34.8 | 0.1 | 601.3 |
| 10 | 0.1 | 35.1 | 0.0 | 561.5 |
| 11 | - | - | 0.0 | 442.5 |
| 12 | - | - | 0.0 | 572.6 |
| 13 | - | - | 0.0 | 524.8 |
| 14 | - | - | 0.1 | 554.3 |
| 15 | - | - | 0.0 | 557.8 |

## 6.3 Network Spectrum Usage

We investigated how the network spectrum is used when the number of nodes with compute capacities is increasing, i.e., when there are more network functions distributed all over the network. We provide the results for the ATLANTA topology in Figure 4.

Plots of Figure 4 show that it is more or less the same subset of links which are the most loaded, but their load vary with the number and location of the NFVs, and the increase of the overall network load when the number of NFVs is increasing. Sometimes we see a drop in the load of a link, which is explained by the increase and position of more NFVs. In conclusion, dimensioning of the link is very dependent on the number and location of the NFVs.

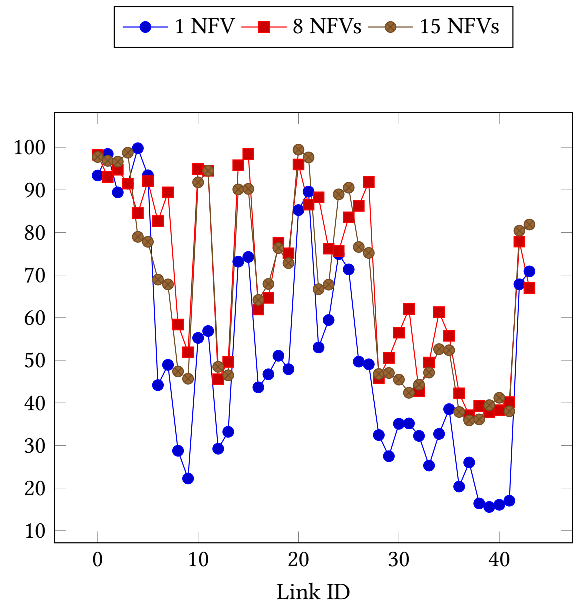


**Figure 4: Physical Link Load - ATLANTA Topology**

We also investigated the throughput evolution when the number of NFV nodes increases and results are depicted in Figure 5 for the ATLANTA network. We observe that as soon as we reach four or five NFV nodes, then the throughput does not increase significantly anymore.

## 7 CONCLUSIONS

We designed a first efficient nested decomposition scheme for reliable 5G slicing. Future work will include several algorithmic enhancements such as parallel solutions of pricing problems and greedy heuristics to generate an initial solution (i.e., initial columns at both decomposition levels).

## ACKNOWLEDGMENTS

B. Jaumard has been supported by a Concordia University Research Chair (Tier I) on the Optimization of Communication Networks and by an NSERC (Natural Sciences and Engineering Research Council of Canada) grant.

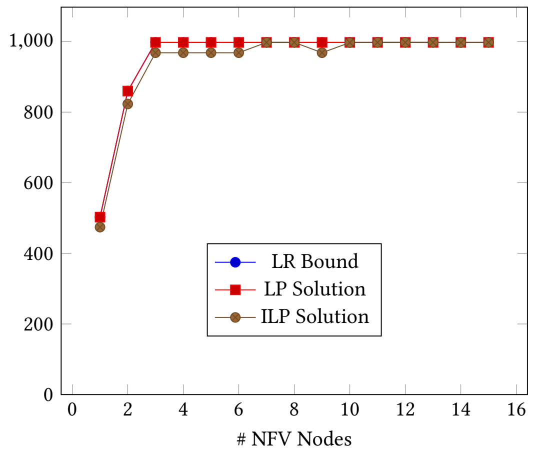


**Figure 5: Throughput evolution with an increasing number of NFV nodes**

## REFERENCES

[1] M.S. Bonfim, K.L. Dias, and S.F.L. Fernandes. 2019. Integrated NFV/SDN Architectures: A Systematic Literature Review. *Journal ACM Computing Surveys (CSUR)* 51, 6 (2019), xxx – xxx.

[2] G. Carella, M. Pauls, A. Medhat, L. Grebe, and T. Magedanz. 2017. A Network Function Virtualization framework for Network Slicing of 5G Networks. In *Mobilkommunikation–Technologien und Anwendungen*. ITG-Fachtagung, Osnabrück, Deutschland, 1–7.

[3] A. Destounis, G. Paschos, S. Paris, J. Leguay, L. Gkatzikis, S. Vassilaras, M. Leconte, and P. Medagliani. 2018. Slice-based column generation for network slicing. In *Annual Joint Conference of the IEEE Computer and Communications Societies - INFOCOM*. IEEE, Honolulu, HI, USA, 1–2.

[4] A. Dohn and A. Mason. 2013. Branch-and-price for staff rostering: An efficient implementation using generic programming and nested column generation. *European Journal of Operational Research* 230 (2013), 157–169.

[5] J.B. Gauthier, J. Desrosiers, and M.E. Lübbecke. 2018. Vector Space Decomposition for Solving Large-Scale Linear Programs. *Operations Research* 66, 5 (2018), 1376–1389.

[6] F. Hennig, B. Nygreen, and M.E. Lübbecke. 2012. Nested Column Generation Applied to the Crude Oil Tanker Routing and Scheduling Problem with Split Pickup and Split Delivery. *Naval Research Logistics* 59 (April âĂŘ June 2012), 298–310. Issue 3âĂŘ4.

[7] N. Huin, B. Jaumard, and F. Giroire. 2018. Optimal Network Service Chain Provisioning. *IEEE/ACM Transactions on Networking* 26, 3 (June 2018), 1320–1333.

[8] S. Karabuk. 2009. A nested decomposition approach for solving the paratransit vehicle scheduling problem. *Transportation Research Part B* 43 (2009), 448–465.

[9] L.S. Lasdon. 1970. *Optimization Theory for Large Systems*. MacMillan, New York.

[10] X. Li, M. Samaka, H.A. Chan, D. Bhamare, L. Gupta, C. Guo, and R. Jain. 2017. Network Slicing for 5G: Challenges and Opportunities. *IEEE Internet Computing* 21, 5 (2017), 20–27.

[11] R. Lin, S. Luo, J. Zhou, S. Wang, B. Chen, X. Zhang, A. Cai, W.-D. Zhong, and M. Zukerman. 2018. Column generation algorithms for virtual network embedding in flexi-grid optical networks. *Optics Express* 26, 8 (Apr 2018), 10898–10913.

[12] M.E. Lübbecke and J. Desrosiers. 2005. Selected Topics in Column Generation. *Operations Research* 53 (2005), 1007–1023. Issue 6.

[13] George L. Nemhauser and Laurence A. Wolsey. 1988. *Integer and Combinatorial Optimization*. Wiley, New York.

[14] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J.J. Ramos-Muñoz, J. Lorca, and J. Folgueira. 2017. Network Slicing for 5G with SDN/NFV: Concepts, Architectures and Challenges. *IEEE Communications Magazine* 55 (2017), 80–87. Issue 5.

[15] A. Pessoa, R. Sadykov, E. Uchoa, and F. Vanderbeck. 2018. Automation and Combination of Linear-Programming Based Stabilization Techniques in Column Generation. *INFORMS Journal on Computing* 30, 2 (2018), 339–360.

[16] SNDlib. 2005. Germany50 Problem. http://sndlib.zib.de/home.action/. (October 2005).

[17] S.H. Song. 2009. A nested column generation algorithm to the meta slab allocation problem in the steel making industry. *Journal International Journal of Production Research* 47, 13 (2009), 3625–3638.

[18] L. Tang, G. Zhao, C. Wang, P. Zhao, and Q. Chen. 2018. Queue-aware reliable embedding algorithm for 5G network slicing. *Computer Networks* 146 (December 2018), 138 – 150.

[19] F. Vanderbeck. 2001. A Nested Decomposition Approach to a Three-Stage, Two-Dimensional Cutting-Stock Problem. *Management Science* 47, 6 (2001), 864–879.

[20] B. Yi, X. Wang, K. Li, S.K. Das, and M. Huan. 2018. A comprehensive survey of Network Function Virtualization. *Computer Networks* 133 (2018), 212 – 262.