Slice-based Column Generation for Network Slicing

Apostolos Destounis, Georgios Paschos, Stefano Paris, Jérémie Leguay, Lazaros Gkatzikis,
Spyridon Vassilaras, Mathieu Leconte, Paolo Medagliani
Mathematical and Algorithmic Science Lab, France Research Center, Huawei Technologies Co. Ltd.
Email: {firstname.lastname}@huawei.com

***Abstract*—We propose an algorithmic framework for network slicing based on column generation. It aims at efficiently maximizing the number of accepted slices while minimizing the cost of embedding virtual networks. It can address key QoS and high reliability constraints for 5G networks and it extensively uses parallelization to quickly solve a relaxed version of the problem and round it to a near-optimal integer solution. This poster presents performance results on small and large problem instances.**

***Index Terms*—Resource Allocation, Network Virtualisation, Combinatorial Optimization.**

I. Introduction

A network slice is a virtual network which is implemented on top of a physical network in a way that creates the illusion of the slice tenant of operating its own dedicated physical network. A virtual link between virtual nodes can be realized as a multi-hop path with reserved bandwidth on all physical links constituting the path. A virtual node can implement specific network functions that can be installed on a physical node (e.g., firewalls, DPI probes). Virtual links and virtual nodes can be easily established by an Software Defined Network (SDN) controller or network orchestrator [1]. Network slicing functionalities are foreseen to be a key component of 5G to provision isolated and personalized network services to different applications (e.g., connected vehicles, smart factories, AR/VR content distribution) [2], [3].

This poster presents an algorithmic framework based on column generation [4] for network slicing which efficiently maps several virtual networks on top of physical resources with QoS and high reliability constraints.

II. Algorithmic Framework

We briefly introduce the multi-slice embedding problem, its extensions for QoS and high-reliability constraints, and the general algorithmic framework we developed.

Multi-slice embedding problem. We consider a set $\mathcal{K}$ of slices to be embedded, each one with a set $\mathcal{S}_k$ of possible embeddings. We want to maximize the number of admitted slices while minimizing the embedding cost. Ideally we must admit all the slices if they fit into the physical network. We denote by c_e and b_e the cost and bandwidth of physical link $e \in \mathcal{E}$, respectively, and $r^k_{e,s}$ the requested bandwidth on physical link e used in candidate embedding s for slice k. Then, the cost of embedding $s \in \mathcal{S}_k$ is $p^k_s := \sum_{e\in\mathcal{E}} r^k_{s,e} c_e$. The optimization variables $x^k_s \in \{0,1\}$ indicate if slice k uses embedding s on the physical network.

We want to solve the Integer Linear Program (ILP) formulated in Fig. 1, where $\beta^k > 0$ is a constant that penalizes not

$$\max_{\mathbf{x}} \sum_{k\in\mathcal{K}} \beta^k \sum_{s\in\mathcal{S}_k} x^k_s - \sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}_k} p^k_s x^k_s \quad (1)$$

$$\text{s.t.} \sum_{s\in\mathcal{S}_k} x^k_s \le 1, \qquad \forall k \in \mathcal{K} \quad (2)$$

$$\sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}_k} r^k_{e,s} x^k_s \le b_e, \qquad \forall e \in \mathcal{E} \quad (3)$$

$$x^k_s \in \{0,1\}, \qquad \forall k \in \mathcal{K}, \forall s \in \mathcal{S}_k$$

Fig. 1: Multi-slice embedding problem.

embedding slice k on the physical network, and signifies the importance of slice k. It must be chosen large enough such that it is always preferable to allocate flows of a slice on the physical network, if at all possible. (Eq. 1)

Slices can be embedded in the network at most once (Eq. 2). In addition, the multiple slices must be embedded in a way that satisfies the capacity constraints of the edges in the physical network (Eq. 3).

QoS constraints and protection schemes. To meet 5G requirements, our framework supports the embedding of slices with latency, capacity and SRLG disjointness constraints on every virtual link. It also supports the following protection mechanisms:

- *1+1 protection:* A primary working path and a backup path are provided for every virtual link.
- *1:N protection:* A primary working path and N sharable backup paths are provided for every virtual link
- *1+1 full protection:* For each slice request, two disjoint network slices are provided.

Algorithmic framework. A naive approach to solve the multi-slice embedding problem could be a greedy algorithm, called Successive Slice Embedding (SSE), which treats the slices one-by-one in a decreasing order of β^k and increasing order of amount of requested traffic, finding minimum-cost single-slice embeddings and reducing the remaining capacity due to each slice embedding. However, such an algorithm does not coordinate the resource allocation among slices. If the load is high, choices made for the first slices might quickly use up the capacity of important links, and may lead to the rejection of remaining slices. Our main objective is to embed the slices for the maximum cumulative value, and hence we need to plan for all the slices jointly. Therefore, we devise a method that solves efficiently the optimization problem presented above.

However, due to the integral nature of the problem, in medium to large problem instances running-time explodes. An

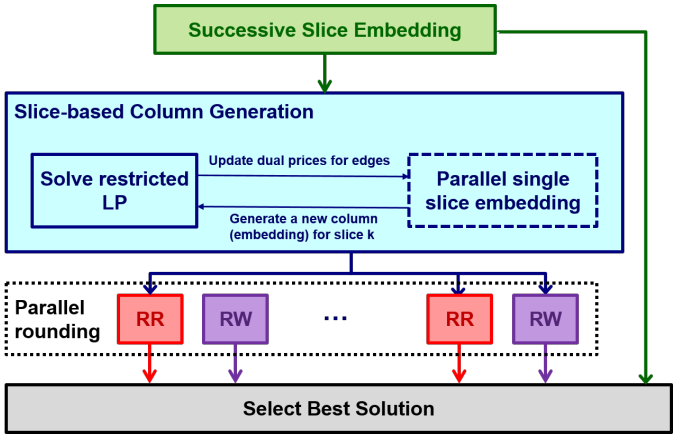

Fig. 2: Slice-based column generation framework.

efficient approach to tackle this issue is to relax integrality, find a coordinated fractional routing for all the slices, and correct it to finally obtain an integral routing. From properties of LPs (Linear Programs), few slices actually use non-integral routing and the fractional solution is generally easy to patch.

Our slice-based column generation algorithm is depicted in Fig. 2. It aims at efficiently solving the relaxed problem and correct it to an integral solution. It first warm starts from an initial solution using SSE. If some slices cannot be embedded at this stage, they are put on dummy embeddings with very high costs β^k. Then, column generation starts by solving at each iteration a restricted LP on the current set of embeddings, updating the dual prices of the physical edges, and generating new columns (possible single-slice embeddings). Columns are generated in parallel and may replace existing ones to keep the master problem small while accelerating convergence. All constraints related to QoS and high reliability are addressed in the generation of columns. At the end of each iteration, we monitor the improvement of the master problem. If no improvements have been made for several iterations, we exit and keep the last fractional solution. We also monitor execution time and estimate if we have time for one more iteration. Once the solution to the relaxed problem has been found, several instances of rounding methods are instantiated with different random seeds to increase diversity. Methods like Randomize Rounding (RR) [5] or Re-Weighting (RW)-based rounding are used.

III. Numerical results

We present simulation results obtained on a server with 40 cores on small (10 nodes, 30 slices) and large (200 nodes, 7 slices) instances (20 trials in each case). We compare our slice generation algorithm with Successive Slice Embedding (SSE) and the optimal solution obtained with CPLEX (for small instances). We considered random graphs with representative distributions of node degrees and SRLGs. Tab. I provides average properties of the instances. Results are shown as a function of congestion, i.e., a multiplication factor that scales up or down the size of virtual links in each slice. Each instance is generated in such a way that it is marginally possible to admit all the slices (with their respective protection mechanism) under a congestion level of 1.

Phy. Nodes	Phy. Links	SRLGs	Virt. links	Slices
10	38	43	43	30
200	4005	1195	403	7

TABLE I: Average properties of instances.

Results on small instances. We can see that the number of accepted slices is nearly optimal and 25% higher than for SSE. The execution time of ILP explodes when the congestion increases as the problem becomes harder while it remains very low for our algorithm.

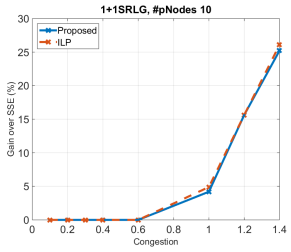

Fig. 3: Gain over SSP in % of accepted slices vs congestion.

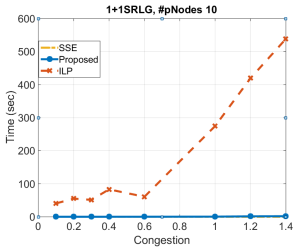

Fig. 4: Execution time vs congestion.

Results on large instances. On networks with 200 nodes, the gain is up to 20% in terms of accepted slices. The execution time of slice generation reaches a plateau as the algorithm is designed and parametrized to meet an execution time requirement of 1500s. When the congestion increases, admission control dominates the minimization of embedding cost which makes the problem easier to solve, hence reducing the execution time.

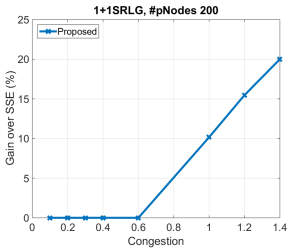

Fig. 5: Gain over SSP in % of accepted slices vs congestion.

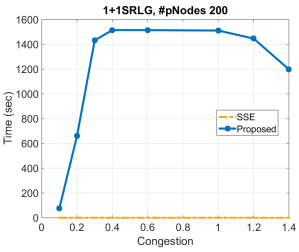

Fig. 6: Execution time vs congestion.

References

[1] D. Kreutz, F. M. V. Ramos, P. E. Verssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, Jan 2015.

[2] S. Vassilaras, L. Gkatzikis, N. Liakopoulos, I. N. Stiakogiannakis, M. Qi, L. Shi, L. Liu, M. Debbah, and G. S. Paschos. The algorithmic aspects of network slicing. *IEEE Communications Magazine*, 2017.

[3] 5G Service-Guaranteed Network Slicing White Paper. Huaweis whitepaper, February 2017.

[4] G. Desaulniers, J. Desrosiers, and M.M. Solomon. *Column Generation*. GERAD 25th anniversary series. Springer US, 2006.

[5] P. Raghavan and C. D. Tompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, Dec 1987.