This is a peer-reviewed, accepted author manuscript of the following conference paper: Murray, A, Arulselvan, A, Roper, M, Cashmore, M, Mohalik, SK, Burdick, I & David, S 2023, The cost of quality of service: SLA aware VNF placement and routing using column generation. in Proceedings of 2023 13th International Workshop on Resilient Networks Design and Modeling (RNDM), IEEE, Piscataway, NJ. https://doi.org/10.1109/RNDM59149.2023

A Column Generation Based VNF Placement and Routing Algorithm for Network Slicing

Andrew Murray*, Ashwin Arulselvan*, Marc Roper*, Michael Cashmore*,
Swarup Kumar Mohalik† and Ian Burdick‡
* University of Strathclyde, Glasgow, {a.murray, ashwin.arulselvan, marc.roper, michael.cashmore}@strath.ac.uk
† Ericsson Research, Bangalore, swarup.kumar.mohalik@ericsson.com
‡Ericsson Inc., Dallas, ian.burdick@ericsson.com

***Abstract*—To provide tailored services for diverse use cases, 5G networks will use network slicing. In network slicing, multiple "slices" corresponding to each use case are hosted on a shared physical infrastructure. In the network function virtualization (NFV) paradigm, these slices are virtualised and the service is provided by routing and processing traffic through an ordered sequence of virtual network functions (VNF). The quality of the service (QoS) depends on the quantity and relative placement of the VNFs, and is quantified by a set of key performance indicators (KPIs), in a service level agreement (SLA): a contract reached between the internet service provider (ISP) and customer. In order to slice the network in line with the SLA, ISPs must consider the SLA constraints directly when placing VNFs and routing service requests. In this paper, we present a VNF placement and routing algorithm based on the column generation method which iterates between generating improving paths, and optimising the placement of the VNFs and routing of traffic given the generated paths. SLA constraints are modelled as soft constraints for which violation incurs a cost, the sum of which is minimised. Unlike prior approaches, we consider the throughput, latency and availability SLA constraints. We validate our approach on a number of mobile edge cloud (MEC) networks using 3 realistic network slice scenarios. We show that our approach can find near optimal solutions ($\pm 10\%$ of optimal value) to realistic sized scenarios (up to 28 vertices, 41 edges and 700 service requests) within a reasonable time-frame ($<$ 1 hour).**

***Index Terms*—Network Function Virtualization, Quality of Service, Column Generation, Service Function Chains, Availability.**

I. Introduction

In contrast to previous generations of mobile networks, 5G networks will use network slicing, with multiple virtual network *slices*, overlaying a shared physical infrastructure [1]. Network slicing enables Internet Service Providers (ISP) to optimise infrastructure usage while offering tailored network services to satisfy diverse use cases such as Internet of Things (IoT), industrial automation, autonomous driving and UHD video streaming [2]. The slice requirements are characterised by varying Key Performance Indicators (KPI) (e.g. latency, throughput, availability, connection density) [3] quantified in the Service Level Agreement (SLA): a contract reached between the ISP and customer. Providing quality of service (QoS) means delivering the end-to-end service subject to the constraints imposed by the SLA.

Each slice is comprised of a number of Service Function Chains (SFC), where each SFC is a request to route and process traffic via an ordered sequence of Network Functions (NF) (e.g. load balancer, traffic monitor, firewall) [5]. Within the Network Function virtualization (NFV) paradigm, NFs are Virtual Network Functions (VNF) implemented in software running on industry standard high volume servers [6]. Finding a suitable placement of the VNFs within the network, and subsequently routing the SFC requests is a hard problem which has attracted significant attention and has been designated the VNF Placement and Routing Problem (VNF-PRP) [7].

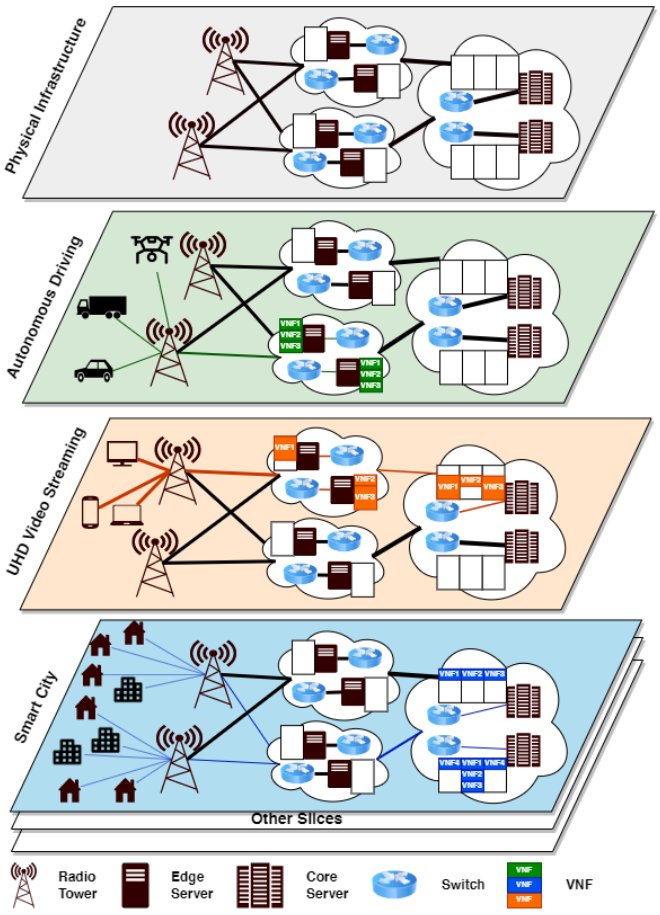

Fig. 1: VNF placement example for 5G network slices (adapted from [4]).

Numerous papers have been presented in recent years

tackling different variations of this problem. However these approaches typically do one of the following: 1) minimise operational expenditure while neglecting QoS, 2) minimise specific QoS terms such as latency or 3) model SLA constraints as hard constraints. When applied to network slicing, the prior may result in a VNF placement guaranteed to violate QoS for some slices, the second offers no distinction between the different QoS requirements of each network slice and the latter will simply result in no solution when it is not possible to satisfy the QoS constraints for a particular SFC.

5G networks use a Mobile Edge Cloud (MEC) architecture [8] with a combination of centralised core data-centers (DC) and localised NFV enabled edge cloud servers. Core DCs have almost infinite compute and memory resources, however they can often be some distance from the user leading to high latency; while edge servers are resource constrained but tend to be placed close to the access points, thus lowering latency [4]. An example of expected 5G network slices is provided in Figure 1. In order to satisfy the low latency constraint for the autonomous driving slice, it may be beneficial to host the VNFs at the edge. The service also requires high availability; replicating the VNFs and hosting them on different servers increases the availability, since in the event of a node failure, traffic can still be processed by the replica. The total data rate required for the service can then be split down two paths, one visiting the master VNFs and one visiting the replicas. The UHD streaming service has high data-rates meaning that a large proportion of the overall network bandwidth must be allocated. Likewise, the VNFs have traffic processing limits and so must be replicated to cope with demand. The data-rate can be split down each replica as per the autonomous driving case. Finally the IoT based smart city service has high connection density but is not as sensitive to QoS and therefore can be placed according to resource availability.

In this paper we present a VNF-PRP algorithm based on column generation, in which we iteratively solve a Restricted Master Problem (RMP), which optimises the placement, replication and routing of the VNFs given the service paths generated so far; and a constrained shortest path Column Generation Problem (CGP) which generates new, improving paths. We treat SLA constraints (data-rate, latency and availability) as soft constraints and then minimise SLA violation cost, meaning that we satisfy all SLA constraints *if possible* otherwise we satisfy *as many of them as possible*.

In Section II we place the contribution of this paper in context with respect to related work. In Section III we formally define the VNF-PRP. In Section IV we present one possible solution approach utilising the column generation method. In Section V we describe the setup and in Section VI the results of our experimental evaluation. We conclude and address avenues for future research in Section VII.

II. Related Work

There has been a plethora of literature in recent years addressing the problem of VNF placement. For a comprehensive overview on the literature we refer the reader to a relevant survey [9], [10]. In this section we address only those which are most relevant. Sun et al. [10] classify the VNF placement problem into 4 distinct sub problems: the *chaining problem* computes the VNFs and outputs a service function chain (SFC) based on demand; the *embedding problem* uses the SFC as input and maps it to the physical links in the network subject to bandwidth resources; the *placement problem* allocates VNFs to the network subject to compute and network resources and the *routing problem* routes flows through the respective VNFs. Past literature has primarily involved: a) solving a combination of these sub problems, b) using an optimisation method, c) for a particular use case, d) optimising some metric.

a) VNF Placement in General: Early work in literature tackled the problem of placement, chaining and embedding but neglected the flow routing. Cohen et al. [11] formalised the VNF placement problem by drawing comparisons between well known Operations Research problems; the facility location problem and the general assignment problem. They present a near-optimal algorithm based on linearly relaxing an ILP to an LP and then rounding the optimal solution so that it is integral. However, the order in which the VNFs are traversed and the bandwidth capacity of the links are not considered. To deal with the transition to NFV, the authors in [12] considered a hybrid scenario composed of a combination of middle-box and VNFs. By taking inspiration from virtual network embedding [13], they develop an ILP which places VNFs and maps service chains onto the physical infrastructure while minimising servers used. While their work considers the bandwidth and latency of network links, they do not consider the ordering of VNFs which is a key feature of VNF placement. The authors in [14] consider the chaining of VNFs and hence maintain the VNF ordering, however they solve the chaining and placement problems sequentially. First they enumerate the SFC's and then use a mixed integer quadratically constrained program (MIQCP) to embed them in the network while optimising numerous objectives.

The problem we address is better compared to the VNF placement and routing problem introduced in [7]. In this problem, service chains are directly encoded in the optimisation as a flow routing problem, where we try to route traffic demands down any number of paths while simultaneously placing VNFs directly on the paths subject to the ordering constraints. In [7], the authors encode this problem as a multi-objective ILP minimising both servers used and maximum network link utilization. They then present a math heuristic based on prioritisation of objectives to solve larger problem instances. However, they do not consider the replication of VNFs; in reality VNFs are often replicated for load balancing, fault tolerance and to cope with demand. Carpio et al. [15] introduced the VNF placement problem with replication which takes this into account. However, their solution method involves three sequential optimisation phases: first they enumerate a set of viable paths for each service, then they find the optimal placement of the VNFs on the enumerated paths and finally, they see if it is possible to improve the solution by introducing replicas. Each sub problem is then solved using a genetic

algorithm.

b) Solution Methods: In terms of approach, VNF placement has predominantly been solved using exact or heuristic methods. The authors in [7], [12], [14], [16]–[28] present ILP's which can exactly compute the optimal solution, however they are incapable of solving problem instances of a practical size. To achieve scalability, heuristics of varying flavours are often presented. The authors in [18], [19], [23], [24], [27] present greedy algorithms, Bari et al. [16] presents a dynamic programming based heuristic, Addis et al. [7] present a math-heuristic, Ghaznavi et al. [21] and Luizelli et al. [28] present a local and binary search based heuristic and Carpio et al. [15] use genetic algorithms. While computationally more efficient, such approaches do not offer guarantees on solution quality. On the other hand column generation can be used as a heuristic to solve practical sized problems while offering bounded optimality. Liu et al. [29] use column generation and exploit it's any-time property to dynamically place VNFs to satisfy new service requests, however their column generation sub problem is an ILP which has an exponential runtime. Huin et al. [30] tackle the VNF placement problem in static scenarios and show that the pricing problem can be formulated as a shortest path problem with polynomial time complexity. However they do not consider the QoS constraints such as latency and availability and assume that each service request is mapped to exactly one path, with the routing demands down each path known apriori. Furthermore they assume that the VNF's instances can be fractionally split in terms of CPU and RAM. This is not the case and can result in infeasible assignments of the VNF's to the computational resources.

c) Use Case: The VNF placement literature can also be categorised into general placement and application specific placement. VNF Placement for 5G has some interesting characteristics which render many of the general placement strategies insufficient. Cao et al. [31] study the problem of VNF Forward Graph (VNF-FG) design and embedding in 5G networks. A two-step method is proposed to generate the VNF-FG graphs according to the service requests, then 4 genetic algorithms are used to embed the graphs and place the VNFs on the network while minimising bandwidth consumption and maximum link utilization. Agarwal et al. [32] present a heuristic approach to solve the problems of VNF placement and allocation in 5G networks while minimising latency. They note that VNF placement ensures that the minimum compute resources required for each VNF are available on the host node; but that additional resources can be provided to allow the VNF to process traffic more quickly. The allocation problem uses a queuing model to schedule the compute resources to individual VNFs hosted on the same node. Both [31] and [32] focus solely on the mobile core; [4] highlight that in order to satisfy the ultra low latency requirement of some 5G services, placing VNFs at the edge is mandatory. They present an Adaptive Interference Aware (AIA) based heuristic which places VNFs on network slices wihin an edge-cloud architecture. They optimise throughput of accepted requests subject to slice-specific latency constraints, however they do not consider availability and solve the VNF-FG design and embedding problems sequentially (similar to [14]). Rather than explicitly modelling availability, [33] introduce a resilient VNF placement ILP for network slicing in which the objective is to minimise the number of SFCs affected given any node was to fail.

d) Optimisation Metric: Prior QoS sensitive approaches to VNF placement have typically handled sub sets of the required KPI's present in the SLA. While most studies consider throughput, Mehraghdam et al. [14] highlighted the importance of considering path latency and performed a Pareto set analysis to show the trade off between latency, number of servers used and link utilization. Following from this, numerous papers have been presented which consider latency as either the objective [24], [34] or as a constraint [7], [16], [28], [35], [36]. Rather than considering latency as a hard constraint which must be satisfied, Ben Jemaa et al. [20] treat it as a soft constraint for which violation incurs a cost. SLA violation cost is then minimised alongside link utilisation and server usage using the weighted sum method. Their approach however does not consider availability and solves the problem using an exact MILP which is not scalable. They consider a small use case with one cloudlet and one cloud server and do not consider flow routing in the optimisation.

More recently, some effort has been placed on incorporating availability in VNF placement algorithms. The authors in [22] present resilient strategies for safe-guarding against specific failures: node failure, link failure and a combination of both. However these strategies do not quantify service availability and therefore are incompatible with the well-defined, numeric SLA requirements. Vizaretta et al. [37] were the first to explicitly model service chain availability as a constraint, using the product of individual VNF, node and link availability. However they model SLA's as hard constraints which simply returns no solution when it is not possible to satisfy all SLA constraints. Furthermore, they do not consider replication, in reality it is impossible to satisfy high availability constraints using only one service chain. Yala et al. [38] solve a weighted bi-objective optimisation problem minimising cost, while maximising availability, where availability is derived using the probability that a node will fail. Similarly, Carpio et al. [39] model availability considering replications in a multi-objective framework. To avoid the non-linearity of the availability function, they choose to optimise a linear inverse penalty function which does not directly quantify availability. Both these approaches consider availability as the objective; in reality different services have significantly different availability requirements which is why we chose to explicitly model availability as a constraint for each service.

III. Problem Definition

The VNF-PRP is a tuple: $\langle \mathcal{G}, \mathcal{S} \rangle$, where $\mathcal{G}$ is the network topology graph and $\mathcal{S}$ is the set of network slices. The graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ has a set of vertices $\mathcal{V}$ representing physical locations within the network, and edges $(v_1, v_2) \in \mathcal{E}$ representing physical connections between the locations. The set of vertices

TABLE I: Master Problem Decision Variables

Variables	
$x^p \in \mathcal{R}^+$	Fraction of flow through path p
$\phi_T^s \in \mathcal{R}^+$	Amount by which SFC s violates the SLA throughput constraint.
$y_n^f \in \mathcal{Z}^+$	Number of instances of VNF f installed on node n.
$q_i^{s,f} \in \{0,1\}$	1 if i distinct nodes are hosting VNF f, for SFC s.
$\phi_A^s \in \{0,1\}$	1 if SFC s violates SLA availability, else 0.
$\beta_n^{s,f} \in \{0,1\}$	1 if VNF f for SFC s is hosted on node n, else 0.
Parameters	
W^s	Cost of violating SLA for SFC S.
C^f	CPU requirement for VNF f.
C_n	CPU resources of server node n.
M^f	Memory requirement for VNF f.
M_n	Memory resources of server node n.
T^s	Throughput requirement for SFC s.
T^f	Throughput capacity of each instance of VNF f.
$B_{v_1v_2}$	Bandwidth capacity of edge (v_1, v_2).
$z_{v_1v_2}^p$	Number of times an edge (v_1, v_2) occurs in a path p.
$\alpha_n^{p,f}$	Number of times a VNF f installed on node n processes traffic in path p.
$A_i^{s,f}$	Availability for i replicas of VNF f for SFC s.
A^s	Availability requirement for SFC s.
N	Min fraction of SFC flow required for a path to be considered in the availability calculation.
K_q^f	Max number of distinct nodes that can host a VNF f.
Φ_L^p	1 if path p violates the SLA latency, else 0.

can be classified into distinct subsets: the set of switches $\mathcal{V}_s$, and the set of computational server nodes $\mathcal{V}_n$. Each node $n \in \mathcal{V}_n$, has compute and memory resources C_n, M_n and availability A_n, while each link (v_1, v_2) has an associated bandwidth capacity and latency, $B_{v_1v_2}$ and $L_{v_1v_2}$ respectively.

Within the network, we must host a set of slices $\mathcal{S}$, where each slice is a set of SFC requests and each SFC request $s = \langle \mathcal{F}^s, (v_0, v_d), \mathcal{R}^s \rangle$, is a tuple. The set $\mathcal{F}^s$, is the ordered set of required VNFs for the Slice, where each VNF f, has compute and memory requirements C^f and M^f, availability A^f, processing latency L^f and throughput capacity T^f. The pair (v_0, v_d), represents the source and sink destination of the SFC request, where v_0 is the source node and v_d is the destination. The set $\mathcal{R}^s$, is the set of slice QoS requirements composed of $\langle T^s, L^s, A^s \rangle$, where T^s, L^s and A^s are the throughput, latency and availability requirements respectively. The VNF-PRP problem we seek to solve involves computing: 1) the quantity of replicas of each required VNF, 2) a placement of each VNF onto compute nodes and 3) a number of paths for each SFC and the fraction of flow to send down each path.

IV. Method

We solve the VNF-PRP via a column generation procedure as outlined in Algorithm 1, in which two optimization phases are iteratively solved:

1) The **Restricted Master Problem** (RMP) in line 7, which finds the number of replicas and placement of VNFs and the routing solution given the paths enumerated so far.
2) A **Column Generation Problem** (CGP) in line 10 for each service, in which we find the best new path to include in the RMP.

It should be noted that the RMP in this problem is a MILP, however we solve a linear relaxation (LRMP) by replacing binary variables with continuous ones. This enables us to employ commercial solvers (e.g. Gurobi, Cplex) which do not permit adding variables at local nodes in the search tree. However it does come with some caveats, for example it is only a heuristic. We will show in the coming section that each path is equivalent to a column in the RMP constraint matrix and so from henceforth we can use the terms *column* and *path* interchangeably.

We initialise the set of paths $\mathcal{P}^s$ for each SFC s using a heuristically generated valid path p in lines 1-3. We extract the dual values (Duals) from the solution to the LRMP in line 7, and use them to model the *reduced cost* which we set as the objective to the CGP. Since we are minimising reduced cost, any path whose reduced cost is negative is called an *improving* column. If an improving column can be found (line 11), we set terminate to False (line 12), and add the new path to the set of paths enumerated for that SFC (line 13). The process then repeats until no improving columns can be found, after which we restore the integrality of variables and solve the RMP as a MILP (line 17).

Algorithm 1: Column Generation for VNF-PRP

```
   Input  : A network, G
            A set of SFCs, S
   Output: Placement of VNFs and routing of services.
1  for s in S do
2  |  p := FindInitialPath(s);
3  |  P^s := {p};
4  end
5  Terminate := False;
6  while Terminate := False do
7  |  Cost, Duals := LRMP(G, S);
8  |  Terminate := True;
9  |  for s in S do
10 |  |  p := CGP(Duals);
11 |  |  if p.ReducedCost ≤ 0 then
12 |  |  |  Terminate := False;
13 |  |  |  P^s.insert(p);
14 |  |  end
15 |  end
16 end
17 Cost, Config := RMP(G, S);
   Return: Cost, Config
```

A. Restricted Master Problem

If we were to enumerate the set containing all valid paths $\mathcal{P}^s$ for every SFC, then we can model the optimization problem

$$\min_{x,y,\phi,u,\beta,q} \sum_{s\in\mathcal{S}} W^s \left(\phi_T^s + \phi_A^s + \sum_{p\in\mathcal{P}^s} \phi_L^p x^p\right)$$

s.t:

$$\sum_{f\in\mathcal{F}} y_n^f C^f \le C_n \qquad n\in\mathcal{V}_n \ (1)$$

$$\sum_{f\in\mathcal{F}} y_n^f M^f \le M_n \qquad n\in\mathcal{V}_n \ (2)$$

$$\sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}^s} z^p_{v_1,v_2} T^s x^p \le B_{v_1,v_2} \qquad (v_1,v_2)\in\mathcal{E}\ \langle\mu_{v_1,v_2}\rangle \ (3)$$

$$\sum_{p\in\mathcal{P}^s} x^p + \phi_T^s \ge 1 \qquad s\in\mathcal{S}\ \langle\pi^s\rangle \ (4)$$

$$\sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}^s} \alpha_n^{p,f} T^s x^p \le T^f y_n^f \qquad f\in\mathcal{F},\ n\in\mathcal{V}_n \langle\nu_n^f\rangle \ (5)$$

$$\sum_{f\in\mathcal{F}_s}\sum_{i=1}^{K_q^f} A_i^{s,f} q_i^{s,f} + \phi_A^s \ge \log A^s \qquad s\in\mathcal{S} \ (6)$$

$$\sum_{i=1}^{K_q^f} q_i^{s,f} = 1 \qquad s\in\mathcal{S},\ f\in\mathcal{F}^s \ (7)$$

$$\sum_{i=1}^{K_q^f} i q_i^{s,f} \le \sum_{n\in\mathcal{N}} \beta_n^{s,f} \qquad s\in\mathcal{S},\ f\in\mathcal{F}^s \ (8)$$

$$\sum_{p\in\mathcal{P}^s} N\alpha_n^{p,f} x^p \ge \beta_n^{s,f} \qquad s\in\mathcal{S},\ f\in\mathcal{F}^s\ n\in\mathcal{V}_n\ \langle u_n^{s,f}\rangle \ (9)$$

Fig. 2: Master Problem MILP

as per Figure 2, with variables and parameters defined in Table I. We refer to this as the Master Problem (MP).

a) Objective: The SLAs are modelled as soft constraints for which violation incurs a cost (denoted W^s), the sum of which is minimised. If it is not possible to satisfy all SLAs, this approach should still satisfy the constraints *as best as possible*. It is trivial to extend this approach to minimise operational cost as a secondary objective. A weighted bi-objective framework could be utilised with the weights selected in accordance with the relative importance of minimising operational costs and satisfying QoS.

b) Compute Constraints: Constraints (1) and (2) ensure that the sum of CPU and memory requirements of the VNFs hosted on each node do not exceed the node resources.

c) Networking Constraints: Constraint (3) says that the sum of all traffic flow passing through each edge must not exceed the bandwidth capacity of the edge. Constraint (4) ensures that as much of the required throughput for each service as possible must be routed down the paths. Constraint (5) says that the flow through all paths which consider a VNF to be installed on a particular node must be zero if that VNF is not installed on the node (i.e. $y_n^f = 0$). Conversely if $y_n^f > 0$, then the flow through all paths which consider a VNF to be installed in that node can be at most $T^f y_n^f$.

d) Availability Constraints: In [15], [40] and [41], availability of a service A^s, is defined as the probability that each sequential VNF works: $A^s = \Pi_{f\in\mathcal{F}} A_n^f$, where A_n^f is simply the sum of the availability of the VNF, and the availability of the node that the VNF is hosted on: $A_n^f = A_n A^f$. The availability of the VNF is the probability that at least one of the replicas works, which is the complement of the probability that all replicas fail: $A^s = \Pi_{f\in\mathcal{F}^s}\left(1 - \Pi_{n\in\mathcal{N}}\left(1 - A_n^f\right)\right)$. A more accurate way to model service availability is by calculating the probability of at least one valid path being available as per Yang et al. [42]. However this results in non-linear constraints which consequently renders the MP intractable. Since there is typically a rich path diversity between any nodes in modern networks [40], it should always be possible to reroute the traffic through the remaining instances of the VNFs in the event of simultaneous node failures. We assume that the availability of each server and VNF is the same, such that we can write the availability of the service as:

$$A^s = \Pi_{f\in\mathcal{F}^s}\left(1 - \left(1 - A_n^f\right)^{q^f}\right) \qquad (1)$$

where q^f is the number of different nodes hosting a VNF. It's worth mentioning that we can have multiple replicas of a VNF running on one node. In which case, the availability $A_n^f = A_n\left(1-(1-A^f)^i\right)$, where i is the number of replicas running on node n. However, we note that $A_n\left(1-(1-A^f)^i\right) \ge A_n A^f$, and so equation (1) is conservative. A similar justification can be made for assuming that the availability of all nodes/VNFs is the same.

Constraint (6) ensures that the SLA availability is satisfied and comes from Equation (1). If A^s is the required availability of the service, then we can rewrite Equation (1) as:

$$\sum_{f\in\mathcal{F}_s} \log\left(1 - \left(1 - A_n^f\right)^{q^f}\right) \ge \log A_s \qquad (2)$$

where q^f is the number of nodes hosting VNF f. We make an assumption here that the number of different nodes hosting a VNF can be at most K_q^f. We can then enumerate the function: $A_i^{s,f} = \log\left(1-\left(1-A_n^f\right)^i\right.$ for $i = 1, 2, .., K_q^f$ and introduce the binary variables $q_i^{s,f}$ which take a value of 1 if i nodes are hosting VNF f for a service s. If the LHS terms are not sufficient to satisfy the availability, then $\phi_A^s = 1$ and the SLA violation cost is incurred. Constraint (7) says that exactly one of these variables must have a value of 1 for each VNF. Constraint (8) forces the number of replicas to be equal to the number of distinct nodes that host the VNF. Constraint (9) forces the variable $\beta_n^{s,f}$ to take a value of 0 if the service flow through the node is less than $1/N$ of the total service flow. This forces the solution away from assigning arbitrarily small flows to paths in order to satisfy the availability. For example if $N = 10$, then each path must have at least $0.1T^s$ flow through it. If one path is insufficient to satisfy availability, a valid solution may be to use two paths, one with $x^{p_1} = 0.9$ and one with $x^{p_2} = 0.1$.

B. Column Generation

Note that there are exponentially many paths in the graph, and therefore enumerating all possible paths can be computationally prohibitive. In addition, we are only interested in the paths from our base that are in the vicinity of the optimal

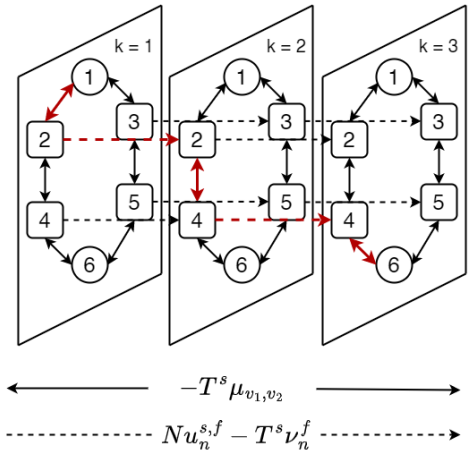

Fig. 3: Diagram showing multi-layered graph.

$$\min_{z,\alpha,\phi} W^s\phi_L^p + \pi^s + \sum_{(v_1^l,v_2^l)\in\hat{\mathcal{E}}} (-T^s\mu_{v_1,v_2}) z_{v_1^l,v_2^l} + \sum_{n^l\in\hat{\mathcal{V}}_n} (Nu_n^{s,f} - T^s\nu_n^f)\alpha_{n^l}$$

s.t:

$$\sum_{(v^l,v_2^l)\in\hat{\mathcal{E}}} z_{v^l,v_2^l} - \sum_{(v_1^l,v^l)\in\hat{\mathcal{E}}} z_{v_1^l,v^l} = \begin{cases} 0 & v^l \in \hat{\mathcal{V}}_s \setminus \hat{\mathcal{V}}_x \quad (1)\\ 1 & v^l = v_0^1 \quad (2)\\ -1 & v^l = v_d^K \quad (3)\end{cases}$$

$$\sum_{(n^l,v_2^l)\in\hat{\mathcal{E}}} z_{n^l,v_2^l} - \sum_{(v_1^l,n^l)\in\hat{\mathcal{E}}} z_{v_1^l,n^l} + \alpha_{n^l} - \alpha_{n^{l-1}} = \begin{cases} 0 & n^l \in \hat{\mathcal{V}}_n \setminus \hat{\mathcal{V}}_x \quad (4)\\ 1 & n^l = v_0^1 \quad (5)\\ -1 & n^l = v_d^K \quad (6)\end{cases}$$

$$\sum_{(v_{1^l},v_{2^l})\in\hat{\mathcal{E}}} z_{v_{1^l},v_{2^l}} L_{v_1,v_2} + \sum_{n^l\in\hat{\mathcal{V}}_n} \alpha_{n^l} L^f - M\phi_L^p \le L^s \quad (7)$$

Fig. 4: Column Generation Problem ILP. Note that $\hat{\mathcal{V}}_x = \{v_0^1, v_d^K\}$

solution. Instead, we iteratively solve a restricted version of the MP, that we refer to as the **Restricted Master Problem** (RMP), using a subset of the total paths. We can imagine that there are many other paths, whose variables x^p take a zero value in the RMP simplex basis. Given a linear program $\min_{\mathrm{x}}\{c^T\mathrm{x} \mid A\mathrm{x} \le b,\ \mathrm{x}_k \ge 0\}$, any variable x_k that takes a zero value in the simplex is known as a *non-basic variable*. The *reduced cost* of introducing a non-basic variable x_k into the simplex basis is: $c_k - A^{k^T}y$, where A^k refers to column k of matrix A and y is the dual vector associated with the constraint $A\mathrm{x} \le b$. For minimisation problems, a column is said to be an *improving* column if the reduced cost is negative [43].

Each non-basic path variable x^p at index k in decision vector x has an associated *column* of coefficients A^k:

$$A^k = \left(\mathbf{0\ 0\ \ z^p_{v_1,v_2}T^s\ -1\ \alpha^{p,f}_n T^s\ 0\ 0\ 0\ -N\alpha^{p,f}_n}\right)^T$$

Where each value in A^k is a vector of coefficients from constraints (1-9). Each path is directly encoded in the column and is defined by precisely 2 variables: $\alpha_n^{p,f}$, which VNFs are installed on which nodes in the path and $z^p_{v_1,v_2}$, how many times each edge occurs in the path.

The dual variables for each constraint are shown in Figure 2. Since constraints (3, 4, 5 and 9) are the only constraints containing the path variable x^p (all other coefficients in A^k are 0), we denote $y = \begin{pmatrix}\mu_{v_1,v_2} & \pi^s & \nu_n^f & u_n^{s,f}\end{pmatrix}$, the dual vector. The reduced cost is given by:

$$\rho = W^s\phi_L^p + \pi^s - \sum_{(v_1,v_2)\in\mathcal{E}} T^s\mu_{v_1,v_2} z^p_{v_1,v_2} - \sum_{n\in\mathcal{V}^n}\sum_{f\in\mathcal{F}} \left(T^s\nu_n^f - Nu_n^{s,f}\right)\alpha_n^{p,f}$$

The best new path is the one that minimises the reduced cost: $\min_{z,\alpha,\phi}(\rho)$. For a comprehensive overview on column generation we refer the reader to a relevant paper [44].

C. Network Transformation

The purpose of this section is to highlight that the CGP can be solved as a constrained shortest path problem on a transformed network. For each SFC, we construct an augmented network $\mathcal{G}^S = \langle\hat{\mathcal{V}},\hat{\mathcal{E}}\rangle$. This network is a K-layered graph, where $K = |\mathcal{F}^s|+1$, and each layer is a copy of the network $\mathcal{G}$. We use $v^l \in \hat{\mathcal{V}}$ to refer to the vertex $v \in \mathcal{V}$ from the original graph on layer l. As per in $\mathcal{G}$, we split the set of vertices into the set of switches $\hat{\mathcal{V}}_s$ and compute nodes $\hat{\mathcal{V}}_n$, The binary variable, $z_{v_1^l,v_2^l}$ says that an edge $(v_1,v_2) \in \mathcal{E}$, in layer l of the graph is used in the path. From each node $n \in \mathcal{V}_n$ in layer l, we add an edge to the equivalent node in layer $l+1$. We have binary variables α_{n^l} that says that one of these edges from layer l is traversed. Traversing this edge represents an assignment of the VNF at index l of the set $\mathcal{F}^s$, to node n: i.e. if $\alpha_{n^l} = 1$, $\alpha_n^{p,f} = 1$. The problem is then to route flow from the service source node v_0 on layer 1 to the destination node v_d on layer K (referred to as v_{0^1} and v_{d^K} respectively). An example is shown for a small network with 2 switches and 4 nodes in Figure 3. Consider that there is an SFC with source $v_0 = 1$, sink $v_d = 6$ and two VNFs. The sub problem would be to find a path from node 1 on layer 1 to node 6 on layer 3. A valid path: $p = \{(1^1,2^1),\ (2^1,2^2),\ (2^2,4^2),\ (4^2,4^3).\ (4^3,6^3)\}$ is highlighted in red in Figure 3 and is equivalent to the column with non-zero terms: $z^p_{1,2} = 1$, $z^p_{2,4} = 1$, $z^p_{4,6} = 1$, $\alpha_2^{p,1} = 1$ and $\alpha_4^{p,2} = 1$. This transformation was first characterised in [30].

An ILP to solve this problem is provided in Figure 4. The objective is analogous to minimising the reduced cost. Constraints (1) and (4) say that the flow is conserved at every vertex that is neither the source or sink (for switch and compute node respectively). Constraints (2), (5) and (3), (6) say that exactly one edge must be active out of the source and into the destination node respectively. Constraint (4) constrains the latency of the path based on the SLA. If the latency is not satisfied, the big M term corresponding to ϕ_L^p is activated and the SLA violation cost is incurred.

Note that the dual values are constants in this problem and the $z_{v_1^l,v_2^l}$ and α_{n^l} variables encode which links in Figure 3 are used. We can consider the terms $-T^s\mu_{v_1,v_2}$ and $Nu_n^{s,f} - T^s\nu_n^f$ as the *edge weights*. Since $\mu_{v_1,v_2} \le 0$ the edge weights

TABLE II: Networks Used

Name	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	No. Core DC's	No. Edge DC's
Abilene	12	15	2	6
Nobel EU	28	41	5	15

$-DR^s\mu_{v_1,v_2}$ are all positive. Only the edges representing the assignment of a node to a VNF can be negative. Since these edges can only be traversed in one direction, the graph has no negative cycles.

If we were to remove constraint (7), along with the term $W^s\phi_L^p + \pi^s$ then what we are left with is just a shortest path problem with positive and negative edge weights but no negative cycles. Such a problem can be solved efficiently using the Bellman-Ford algorithm [45]. The approach we take in this paper is to initially use Bellman-Ford to solve the CGP, check the reduced cost and if it results in an improving column then we add the column and continue. If on the other hand, Bellman-Ford fails to result in an improving column, we resort to solving the ILP from Figure 4.

V. Experimental Setup

As per prior studies [30], network topologies were taken from SNDlib [46]. Of the networks available, Abilene and Nobel-EU were selected as they gave a broad range of network size. In order to simulate a realistic edge-cloud network, a subset of nodes were selected to be core DC's, another subset were selected to be edge DC's and the remaining nodes were set as simple switches which can be access or destination points for SFC requests but have no NFV functionality. Betweenness centrality, a common measure of node importance within a graph, was used to select the most important nodes which were set as the core DC's. A summary of the graphs used is provided in Table II.

Since the edge DC's are more computationally constrained versus the larger core DC's, we set each edge DC as a single NFV node with 40 cores and 40GB of RAM. The Core DC's on the other hand were given 3 NFV nodes, each with 100 cores and 100GB of RAM. This was achieved by minor modification of the network; the node which was selected to be the core DC was set as the DC gateway switch, and was connected to three additional nodes. Service requests could therefore access the gateway and then reach the required VNF on whichever node was hosting it. Apart from this, and due to the significant path diversity in the DCs and the fact that the internal DC latency is negligible, we did not model the internal DC topology. However, a separate routing problem could be solved to route traffic between the DC gateways and the nodes if necessary. Using the coordinates of each node from SNDlib, we computed the length of each link as the distance between nodes, and then used this to compute the relative link latency. Latencies were then scaled in the range $[0, 2]$ in line with prior studies [4], [47]. Bandwidth of each link was randomly sampled from $[10, 40, 100]$ Gbps.

The VNF data used is provided in Table III and was extracted from [4], [36], [48]–[50]. We assume an availability of 0.9999 and 0.999 for servers and VNFs respectively as per Wang et al. [51]. The network slice SFC requirements used are provided in Table IV and have been extracted from prior studies [4], [52], and industrial technical specifications [2], [53]. The cost of violating each SFC was set according to the priority ranking from [52].

TABLE III: VNF Requirements

VNF	C (cores)	M (GB)	T (Mbps)	L (ms)
FW	4	4	600	0.8
TM	10	10	2000	0.1
IDS	8	8	600	0.01
NAT	16	16	3200	0.1
VOC	8	8	2320	0.25
ADNF	8	8	1500	0.1

FW: Firewall, TM: Traffic Monitor, IDS: Intrusion Detection System, NAT: Network Address Translator, WOC: WAN Optimization Controller, VOC: Video Optimization Controller, ADNF: Autonomous Driving Network Function

TABLE IV: SFC Requirements for Different Slices

Network Slice	Required VNFs	T (Mbps)	L (ms)	A (%)	Prob	W^s
Autonomous Driving	NAT-FW-TM-ADNF	10	1	99.999	0.1	3
UHD Streaming	NAT-FW-TM-VOC-IDS	200	100	-	0.4	2
Smart City	NAT-FW-IDS	0.1	-	-	0.5	1

The number of SFC requests were varied, with each request being randomly assigned to slices from Table IV and source and sink node. In order to simulate realistic traffic scenarios, the probability of sampling each service type was estimated from [54] (shown as Prob in Table IV). In order to scale the load on the network, we introduced a "load factor". This scaled the number of users accessing each SFC; for example a load factor 5 meant that every SFC in the network was being accessed by 5 users and therefore the total throughput would be $5T^s$.

Since there are no comparable models incorporating all the features modelled in our approach, we compare our solutions to the LRMP LP solution after column generation, which gives a lower bound on the optimal solution [1].

VI. Results

Experimental results are provided in Table V.

a) SLA Violation Cost: The total SLA violation cost is provided in the objective column (Obj.) which shows the objective of the MILP RMP. Note that since we are minimising, this is an upper bound on the optimal solution. Comparatively, the LP objective column (LP Obj.) shows the optimal solution to the LRMP which is a lower bound on the optimal value. The difference between the upper and lower

[1]Source code and benchmark problems can be accessed online at: https://anonymous.4open.science/r/VNFPP-CG-813B

Network	No. SFCs	Load Factor	No. Nodes Used	LP Obj.	Obj.	Gap (%)	Total Pens.	Av. Pens.	Lt. Pens.	Tp. pens.	Runtime (s)	MILP Runtime (s)	CG Runtime (s)
abilene	20	1	8	3.00	3.00	0.00	1.00	0.00	1.00	0.00	0.58	0.00	0.58
abilene	20	2	9	6.00	6.81	11.87	2.40	0.00	2.00	0.40	0.60	0.03	0.57
abilene	20	3	9	9.00	15.08	40.30	6.01	0.00	3.00	3.01	0.87	0.10	0.77
abilene	20	4	7	12.00	12.00	0.00	4.00	0.00	4.00	0.00	0.84	0.01	0.83
abilene	20	5	9	15.00	15.81	5.11	5.40	0.00	5.00	0.40	0.99	0.21	0.78
abilene	50	1	10	11.40	17.50	34.87	8.02	0.00	5.00	3.02	2.54	0.41	2.13
abilene	50	2	10	22.80	26.91	15.27	12.03	0.00	10.00	2.03	3.16	0.93	2.23
abilene	50	3	11	34.20	64.90	47.30	30.12	0.00	10.20	19.92	3.82	1.23	2.59
abilene	50	4	12	24.00	39.17	38.74	27.51	0.00	20.00	7.51	5.80	2.94	2.86
abilene	50	5	12	29.99	61.76	51.44	40.73	0.00	25.00	15.72	9.05	6.04	3.01
abilene	100	1	10	44.40	50.89	12.75	19.16	0.00	12.80	6.36	5.48	0.41	5.07
abilene	100	2	12	38.39	42.40	9.46	33.98	0.00	32.00	1.98	7.47	2.31	5.16
abilene	100	3	12	50.15	66.31	24.37	57.06	0.00	48.00	9.06	113.30	104.35	8.95
abilene	100	4	12	149.76	165.22	9.36	114.11	0.00	64.00	50.11	702.23	693.77	8.46
abilene	100	5	12	250.22	263.00	4.86	170.10	0.00	80.00	90.10	1005.35	997.26	8.09
abilene	200	1	12	19.20	51.57	62.78	32.02	0.00	16.00	16.02	17.55	7.99	9.56
abilene	200	2	12	140.37	153.04	8.28	91.73	0.00	32.00	59.73	2188.42	2161.83	26.59
abilene	200	3	12	335.08	345.79	3.10	195.42	0.00	48.00	147.42	1494.65	1476.66	17.99
abilene	200	4	12	530.74	540.29	1.77	299.39	0.00	64.00	235.39	424.90	405.50	19.40
abilene	200	5	12	720.69	739.48	2.54	406.48	0.00	80.00	326.48	1788.73	1771.30	17.43
nobeleu	300	1	24	89.40	113.70	21.37	41.53	3.00	27.80	10.73	68.99	34.06	34.93
nobeleu	300	2	28	74.39	118.43	37.19	83.80	0.00	62.00	21.80	3655.43	3600.05	55.38
nobeleu	300	3	30	153.47	185.99	17.48	139.03	0.00	93.00	46.03	3782.73	3600.15	182.58
nobeleu	300	4	30	416.87	437.73	4.77	279.31	0.00	124.00	155.31	3711.25	3600.23	111.02
nobeleu	300	5	30	677.80	692.53	2.13	421.10	0.00	155.00	266.10	3715.92	3600.23	115.69
nobeleu	400	1	27	44.39	44.40	0.02	37.00	0.00	37.00	0.00	66.68	6.18	60.50
nobeleu	400	2	30	86.53	139.39	37.92	106.37	0.00	74.00	32.37	3706.32	3600.02	106.30
nobeleu	400	3	30	422.89	455.51	7.16	280.36	0.00	111.00	169.36	3774.22	3600.03	174.19
nobeleu	400	4	30	778.02	799.28	2.66	470.42	0.00	148.00	322.42	3791.14	3600.05	191.09
nobeleu	400	5	30	1115.80	1141.52	2.25	658.52	0.00	185.00	473.52	3763.31	3600.03	163.28
nobeleu	500	1	27	50.39	65.18	22.69	49.31	0.00	42.00	7.31	3343.36	3246.97	96.39
nobeleu	500	2	30	290.86	321.99	9.67	201.82	0.00	84.00	117.82	3919.13	3600.15	318.98
nobeleu	500	3	30	748.22	778.64	3.91	449.09	0.00	126.00	323.09	3845.18	3600.06	245.12
nobeleu	500	4	30	1216.61	1234.36	1.44	695.50	0.00	168.00	527.50	3820.54	3600.05	220.49
nobeleu	500	5	30	1681.21	1707.05	1.51	949.63	0.00	210.00	739.63	3794.92	3600.05	194.87
nobeleu	600	1	28	72.99	136.58	46.56	91.99	1.00	61.00	29.99	3712.75	3600.07	112.68
nobeleu	600	2	30	419.74	449.87	6.70	284.31	0.00	122.00	162.31	3932.37	3600.07	332.30
nobeleu	600	3	30	947.34	966.76	2.01	571.00	0.00	183.00	388.00	3836.44	3600.05	236.39
nobeleu	600	4	30	1471.87	1493.62	1.46	862.62	0.00	244.00	618.62	3880.90	3600.04	280.86
nobeleu	600	5	30	1998.42	2016.98	0.92	1151.03	0.00	305.00	846.03	3861.30	3600.04	261.26
nobeleu	700	1	30	71.00	106.97	33.63	88.61	0.00	71.00	17.61	3787.39	3600.07	187.32
nobeleu	700	2	30	669.45	706.27	5.21	421.34	0.00	142.00	279.34	4058.44	3600.06	458.38
nobeleu	700	3	30	1329.44	1347.26	1.32	774.52	0.00	213.00	561.52	3908.69	3600.08	308.61
nobeleu	700	4	30	1982.60	2004.39	1.09	1134.89	0.00	284.00	850.89	3882.50	3600.13	282.37
nobeleu	700	5	30	2673.25	2711.54	1.41	1495.86	0.00	355.00	1140.86	3878.93	3600.06	278.87

TABLE V: Table showing experimental results.

bound $(UB - LB)/UB$, is shown as a percentage in the Gap column.

Typically the model performs poorer on cases in which the number of SFCs and load factor is low. This is thought to be attributed to the high fractionality of the assignment variables y_n^f for these cases. In a number of instances, since the flow was low, the throughput constraint (constraint (7) in the LRMP) could be satisfied by setting $y_n^f \approx 0$. Paths were thus generated assuming this was a valid configuration. When integrality was restored and the variables y_n^f were forced to be integer, the ILP tried to set $y_n^f = 1$, but this violated the CPU and RAM constraints for that particular node. Since the y_n^f variables were then required to be zero, and no paths had been generated for other configurations, a number of cases ended up paying some quantity of throughput penalties which were not paid in the LRMP. This is evident for example in Abilene with the number of SFCs 20 and the load factor 3. Nonetheless, the more realistic sized cases tended to be solved to within 10% of optimality.

In the cases we ran, the model managed to satisfy availability for all SFCs. This is due to there being sufficient resources available to at least duplicate the VNFs required for the autonomous driving slice. For every autonomous driving SFC, the model will always choose to split the flow down at least two paths. For example the model could select one path with low latency to send 90% of the flow down, and then another with high latency to send 10% of the flow down. Since the latency penalty is fractional, this would contribute to a cost of $0.1W^s$. Conversely, since the availability penalty is binary, if we were to route all the flow down the low latency path, the model would pay a cost of W^s. Of course a different tradeoff could be achieved through a different parameterisation of the

availability penalty versus latency. It is expected that as the number of network slices increases and the resources become more scarce, the model would be forced to sacrifice availability to some extent.

b) Runtime: The runtime for the algorithm is highlighted in the column "Runtime". We also show a breakdown of the column generation runtime versus the MILP runtime, shown in the columns "CG Runtime" and "MILP Runtime" respectively. Note that the column generation runtime corresponds to the total runtime over the course of all CGP subproblems and LRMP LPs. Since the CGPs are predominantly solved using Bellman-Ford (which has polynomial time complexity $O(|\mathcal{V}||\mathcal{E}|)$), and the LRMP is just solving an LP, most of the runtime is composed of solving the MILP which is NP-hard.

Despite having significantly less variables than if we were to enumerate all paths, the RMP still has an insignificant number of variables (17440 continuous and 1723 integer for nobeleu with 700 services and load factor 5). As such solving the RMP MILP was still impractical for the larger instances. Instead, we set a time-limit of 1 hour (3600s) for Gurobi. When the problem could not be solved in that time, the best incumbent solution was returned which still gives a valid upper bound on the optimal solution. Hence the solutions in which the MILP runtime is quoted as 3600s are those in which the solver failed to find the optimal solution within the time-limit. Despite not finding the optimal solution, the solver is typically able to find solutions that are within 10% of the optimal value.

VII. Conclusion

We present a QoS sensitive VNF-PRP algorithm which satisfies SLA constraints (latency, data-rate and availability) *if possible*, otherwise it minimises the cost of SLA violations. This allows us to generate solutions which are compliant with the diverse requirements of 5G network slices. We introduce one solution method using column generation, in which we iterate between generating *improving* paths and optimising the placement and routing of the VNFs given the generated paths.

We experimentally validated our approach on a realistic MEC architecture generated using SNDlib benchmarks. We show that for realistic sized instances (28 vertices, 41 edges, 700 SFCs), our approach is typically able to find near-optimal solutions within a practical time-frame ($\pm 10\%$ of optimal value within 1 hour).

We conclude by addressing some caveats and avenues for future work. First, we found that when the number of SFC's and the network load is low, the solution can be quite poor ($\pm 62.78\%$ of optimal value for the worst case). This is a result of the fractionality of the VNF assignment variables as explained in Section VI. One solution could be to add cover cuts corresponding to the knapsack constraints (constraints (1) and (2) in Figure 2), which could tighten the LP approximation. Another more radical solution would be to employ a branch and price framework [55] to add columns at local nodes in the search tree. This would permit finding the optimal solution but would require using specialised solvers which may lead to significant increase in runtime. Another potentially interesting avenue is a more comprehensive study of the trade-off between satisfying QoS and minimising operational cost. Operational cost is typically a function of the bandwidth and number of nodes used. One could model the problem as a multi-objective optimization problem as explained in Section IV-A. By varying the weights, a Pareto front of solutions could be generated.

References

[1] H. Yang, T. So, and Y. Xu, "5g network slicing," in *5G NR and Enhancements*. Elsevier, 2022, pp. 621–639.

[2] 3GPP, "Service requirements for the 5g system, rel. 15," Tech. Rep. 22.261, 2016.

[3] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5g: survey and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[4] Q. Zhang, F. Liu, and C. Zeng, "Adaptive interference-aware vnf placement for service-customized 5g network slices," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 2449–2457.

[5] ETSI, "Network functions virtualisation (nfv) architectural framework," Tech. Rep. NFV-002, 2014.

[6] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: challenges and opportunities for innovations," *IEEE communications magazine*, vol. 53, no. 2, pp. 90–97, 2015.

[7] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *IEEE International Conference on Cloud Networking (CloudNet)*. IEEE, 2015, pp. 171–177.

[8] H. Liu, F. Eldarrat, H. Alqahtani, A. Reznik, X. De Foy, and Y. Zhang, "Mobile edge cloud system: Architectures, challenges, and approaches," *IEEE Systems Journal*, vol. 12, no. 3, pp. 2495–2508, 2017.

[9] A. Laghrissi and T. Taleb, "A survey on the placement of virtual resources and virtual network functions," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1409–1434, 2018.

[10] J. Sun, Y. Zhang, F. Liu, H. Wang, X. Xu, and Y. Li, "A survey on the placement of virtual network functions," *Journal of Network and Computer Applications*, vol. 202, p. 103361, 2022.

[11] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 1346–1354.

[12] H. Moens and F. De Turck, "Vnf-p: A model for efficient placement of virtualized network functions," in *International Conference on Network and Service Management (CNSM) and Workshop*. IEEE, 2014, pp. 418–423.

[13] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: a survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[14] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *IEEE International Conference on Cloud Networking (CloudNet)*. IEEE, 2014, pp. 7–13.

[15] F. Carpio, S. Dhahri, and A. Jukan, "Vnf placement with replication for loac balancing in nfv networks," in *IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–6.

[16] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *International conference on network and service management (CNSM)*. IEEE, 2015, pp. 50–56.

[17] A. Baumgartner, V. S. Reddy, and T. Bauschert, "Mobile core network virtualization: A model for combined virtual core network function placement and topology optimization," in *IEEE conference on Network Softwarization (NetSoft)*. IEEE, 2015, pp. 1–9.

[18] A. Mohammadkhan, S. Ghapani, G. Liu, W. Zhang, K. Ramakrishnan, and T. Wood, "Virtual function placement and traffic steering in flexible and dynamic software defined networks," in *IEEE International Workshop on Local and Metropolitan Area Networks*. IEEE, 2015, pp. 1–6.

[19] R. Riggio, A. Bradai, T. Rasheed, J. Schulz-Zander, S. Kuklinski, and T. Ahmed, "Virtual network functions orchestration in wireless networks," in *International conference on network and service management (CNSM)*. IEEE, 2015, pp. 108–116.

[20] F. B. Jemaa, G. Pujolle, and M. Pariente, "Qos-aware vnf placement optimization in edge-central carrier cloud architecture," in *IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–7.

[21] M. Ghaznavi, N. Shahriar, R. Ahmed, and R. Boutaba, "Service function chaining simplified," *arXiv preprint arXiv:1601.00751*, 2016.

[22] A. Hmaity, M. Savi, F. Musumeci, M. Tornatore, and A. Pattavina, "Virtual network function placement for resilient service chain provisioning," in *International Workshop on Resilient Networks Design and Modeling (RNDM)*. IEEE, 2016, pp. 245–252.

[23] Q. Sun, P. Lu, W. Lu, and Z. Zhu, "Forecast-assisted nfv service chain deployment based on affiliation-aware vnf placement," in *Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–6.

[24] D. Bhamare, M. Samaka, A. Erbad, R. Jain, L. Gupta, and H. A. Chan, "Optimal virtual network function placement in multi-cloud service function chaining architecture," *Computer Communications*, vol. 102, pp. 1–16, 2017.

[25] D. Dietrich, C. Papagianni, P. Papadimitriou, and J. S. Baras, "Network function placement on virtualized cellular cores," in *International Conference on Communication Systems and Networks (COMSNETS)*, 2017, pp. 259–266.

[26] A. Gupta, M. F. Habib, U. Mandal, P. Chowdhury, M. Tornatore, and B. Mukherjee, "On service-chaining strategies using virtual network functions in operator networks," *Computer Networks*, vol. 133, pp. 1–16, 2018.

[27] A. Ali, C. Anagnostopoulos, and D. P. Pezaros, "On the optimality of virtualized security function placement in multi-tenant data centers," in *IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.

[28] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2015, pp. 98–106.

[29] J. Liu, W. Lu, F. Zhou, P. Lu, and Z. Zhu, "On dynamic service function chain deployment and readjustment," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 543–553, 2017.

[30] N. Huin, B. Jaumard, and F. Giroire, "Optimal network service chain provisioning," *IEEE/ACM Transactions on Networking*, vol. 26, no. 3, pp. 1320–1333, 2018.

[31] J. Cao, Y. Zhang, W. An, X. Chen, J. Sun, and Y. Han, "Vnf-fg design and vnf placement for 5g mobile networks," *Science China Information Sciences*, vol. 60, pp. 1–15, 2017.

[32] S. Agarwal, F. Malandrino, C.-F. Chiasserini, and S. De, "Joint vnf placement and cpu allocation in 5g," in *IEEE INFOCOM 2018-IEEE conference on computer communications*. IEEE, 2018, pp. 1943–1951.

[33] P. M. Mohan and M. Gurusamy, "Resilient vnf placement for service chain embedding in diversified 5g network slices," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[34] A. Laghrissi, T. Taleb, M. Bagaa, and H. Flinck, "Towards edge slicing: Vnf placement algorithms for a dynamic & realistic edge cloud environment," in *IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2017, pp. 1–6.

[35] S. Yang, F. Li, S. Trajanovski, X. Chen, Y. Wang, and X. Fu, "Delay-aware virtual network function placement and routing in edge clouds," *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, pp. 445–459, 2019.

[36] A. Varasteh, B. Madiwalar, A. Van Bemten, W. Kellerer, and C. Mas-Machuca, "Holu: Power-aware and delay-constrained vnf placement and chaining," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1524–1539, 2021.

[37] P. Vizarreta, M. Condoluci, C. M. Machuca, T. Mahmoodi, and W. Kellerer, "Qos-driven function placement reducing expenditures in nfv deployments," in *IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–7.

[38] L. Yala, P. A. Frangoudis, G. Lucarelli, and A. Ksentini, "Cost and availability aware resource allocation and virtual function placement for cdnaas provision," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1334–1348, 2018.

[39] F. Carpio and A. Jukan, "Improving reliability of service function chains with combined vnf migrations and replications," *arXiv preprint arXiv:1711.08965*, 2017.

[40] D. Li, P. Hong, K. Xue, and J. Pei, "Availability aware vnf deployment in datacenter through shared redundancy and multi-tenancy," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1651–1664, 2019.

[41] Y. Alahmad, A. Agarwal, and T. Daradkeh, "Cost and availability-aware vnf selection and placement for network services in nfv," in *2020 International Symposium on Networks, Computers and Communications (ISNCC)*. IEEE, 2020, pp. 1–6.

[42] S. Yang, S. Trajanovski, and F. A. Kuipers, "Availability-based path selection," in *International Workshop on Reliable Networks Design and Modeling (RNDM)*. IEEE, 2014, pp. 39–46.

[43] G. B. Dantzig, *Linear programming and extensions*. Princeton: Princeton University Press, 1963.

[44] M. E. Lübbecke and J. Desrosiers, "Selected topics in column generation," *Operations research*, vol. 53, no. 6, pp. 1007–1023, 2005.

[45] A. Goldberg and T. Radzik, "A heuristic improvement of the bellman-ford algorithm," STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, Tech. Rep., 1993.

[46] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "Sndlib 1.0—survivable network design library," *Networks: An International Journal*, vol. 55, no. 3, pp. 276–286, 2010.

[47] N. Promwongsa, A. Ebrahimzadeh, R. H. Glitho, and N. Crespi, "Joint vnf placement and scheduling for latency-sensitive services," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 4, pp. 2432–2449, 2022.

[48] S. M. Araujo, F. S. de Souza, and G. R. Mateus, "A demand aware strategy for a machine learning approach to vnf-pc problem," in *IEEE International Conference on Cloud Networking (CloudNet)*. IEEE, 2022, pp. 211–219.

[49] Y. Gu, Y. Hu, Y. Ding, J. Lu, and J. Xie, "Elastic virtual network function orchestration policy based on workload prediction," *IEEE Access*, vol. 7, pp. 96 868–96 878, 2019.

[50] J. Cai, K. Qian, J. Luo, and K. Zhu, "Sarm: service function chain active reconfiguration mechanism based on load and demand prediction," *International Journal of Intelligent Systems*, vol. 37, no. 9, pp. 6388–6414, 2022.

[51] M. Wang, B. Cheng, and J. Chen, "Joint availability guarantee and resource optimization of virtual network function placement in data center networks," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 821–834, 2020.

[52] A. Jalalian, S. Yousefi, and T. Kunz, "Network slicing in virtualized 5g core with vnf sharing," *Journal of Network and Computer Applications*, p. 103631, 2023.

[53] 3GPP, "Service requirements for cyber-physical control applications in vertical domains, rel. 16," Tech. Rep. 22.104, 2018.

[54] U. Cisco, "Cisco annual internet report (2018–2023) white paper," *Cisco: San Jose, CA, USA*, vol. 10, no. 1, pp. 1–35, 2020.

[55] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance, "Branch-and-price: Column generation for solving huge integer programs," *Operations research*, vol. 46, no. 3, pp. 316–329, 1998.