Explanation of parameters file and makefile

Rafael Colares[a,*]

[a] *Orange Innovation, 46 Avenue de la République, 92320 Châtillon, France*

Abstract

This is document provides an explanation of the parameters file (usually referred to as params.txt) used for launching the optimization code. It also contains information on the makefile used for compiling the code.

Keywords: README, Tutorial

1. Input File Paths

This section regroups the addresses of the input files. Four files are required: Node, Link, Demand and VNF.

1.1. Node File : nodeFile

Defines the nodes in the topology. The file should be composed of six columns separated by a semi-colon (;). They are:

- name: Stands for the node name.
- lat: Stands for the node latitude coordinate (it is only there because SNDLib instances have it, but is not used anywhere in the optimization).
- long: Stands for the node longitude coordinate (it is only there because SNDLib instances have it, but is not used anywhere in the optimization).

*Corresponding author
Email address: rafael.colares@gmail.com (Rafael Colares)

- capacity: Stands for the node's capacity.
- availability: Stands for the node's availability (should have a value between 0 and 1).
- cost: Stands for the unitary usage cost of the node.

1.2. Link File: linkFile

Defines the links in the topology. The file should be composed of five columns separated by a semi-colon (;). They are:

- name: Stands for the link's name.
- source: Stands for the name of the source node (*cf.* Node file).
- target: Stands for the name of the target node (*cf.* Node file).
- delay: Stands for the delay consumed to traverse the link.
- bandwidth: Stands for the link's available bandwidth.

1.3. Demand File: demandFile

Defines the Service Function Chains (SFCs) to be considered in the optimization. The file should be composed of seven columns separated by a semi-colon (;). They are:

- name: Stands for the SFC's name.
- source: Stands for the name of the SFC's source node (*cf.* Node file).
- target: Stands for the name of the SFC's target node (*cf.* Node file).
- max_latency: Stands for the maximum delay authorized for the SFC.
- bandwidth: Stands for the SFC's required bandwidth.
- availability: Stands for the SFC's required availability.
- vnf_list: Stands for the SFC's list of VNFs (in order of passage). The list should be composed of the VNF names (*cf.* VNF file) each separated by a colon (,).

1.4. VNF File: vnfFile

Defines the Virtual Network Functions (VNFs) available to be installed throughout the network. The file should be composed of two columns separated by a semi-colon (;). They are:

- name: Stands for the VNF's name.
- consumption: Stands for the amount of resources consumed by the VNF.

2. Optimization Parameters

This section regroups the parameters used for controlling and defining the optimization procedure.

2.1. Linear Relaxation: linearRelaxation

This is a binary parameter: 1 for applying linear relaxation. 0 otherwise.

2.2. Time Limit: timeLimit

Should receive an integer number standing for the number of seconds allowed for carrying the optimization.

2.3. Lazy Constraints: lazy

This is a binary parameter: 1 for activating the avalability lazy constraints (the exponential-sized linear inequalities imposing SFC's availability satisfaction). 0 otherwise.

2.4. Routing Module: routing

This is a binary parameter: 1 for activating the routing module. 0 otherwise. (If set to 0, routing will not be taken into account during optimization: INOC's case).

2.5. Availability Approximations: availability_approx

Should receive an integer number between -1 and 1. -1 for applying piecewise linear approximation to obtain a RESTRICTED problem. 0 for not applying any approximation. 1 for applying piecewise linear approximation to obtain a RELAXED problem.

2.6. Number of Breakpoints: nb_breakpoints

Should receive an integer number greater than or equal to 2. Refers to the number of breakpoints to be used in the piecewise linear approximations (if no approximation is used, this won't be used).

3. Formulation Improvements

This section regroups every improvement proposed to the original formulation. All parameters in this section are binary.

3.1. Disaggregated VNF Placement constraints: disaggregated_VNF_Placement

Stands for the activation of disaggregated VNF placement constraints. (Very basic constraints, should always be activated)

3.2. Strong node capacity constraints: stong_node_capacity

Stands for the inclusion of strong node capacity family of constraints. These constraints are polynomially sized.

3.3. Availability cuts: availability_cuts

States whether the separation problem for the availability constraints should be solved (heuristically) in every node of the Branch-and-Cut tree.

3.4. Node Cover Constraints: node_cover

States whether the separation problem for the subfamily of node cover constraints should be solved in every node of the Branch-and-Cut tree.

3.5. Chain Cover Constraints: chain_cover

States whether the separation problem for the generalized cover and chain cover constraints should be solved (heuristically) in every node of the Branch-and-Cut tree.

3.6. VNF Lower Bound Constraints: vnf_lower_bound

States whether the separation problem for the VNF Lower Bound Constraints should be solved in every node of the Branch-and-Cut tree.

3.7. Section Failure Constraints: section_failure

States whether the separation problem for the section failure constraints should be solved in every node of the Branch-and-Cut tree.

4. Output File Paths

Contains only one parameter (named outputFile) referring to the path of the file to store the results obtained from the optimization. By the end of the optimization, the code will write a new line to the file (if a feasible solution is found within time limit). The line will contain, in order:

- Name of Link file;
- Name of Node file;
- Name of Demand file;
- Name of VNF file;
- Type of approximation;
- Time in seconds consumed by the optimization procedure;
- Upper Bound obtained by the end of the optimization;
- Lower Bound obtained by the end of the optimization;
- Final gap;

- Number of nodes explored;
- Number of nodes left open;
- Number of lazy constraints added;
- Number of user cuts added;
- Time spent during separation problems and heuristics;
- Number of SFC's with availability not respected (for evaluating the relaxation case);
- Max availability violation (for evaluating the relaxation case);

5. Makefile

For adapting the makefile, it suffices to input the right paths of Boost, CPLEX and LEMON libraries on your computer.

The code was designed in Linux, to be executed in Linux.

References

How to model resiliency of Service Function Chains

Rafael Colares[a,*]

[a] *Orange Labs, 46 Avenue de la République, 92320 Châtillon, France*

Abstract

This is document provides some ideas on how to express the availability requirements of a Service Function Chain (SFC) within the SFC routing problem.

Keywords: Virtual Network Functions, Service Function Chain, availability, combinatorial optimization, conflict graphs, lazy constraints

1. Introduction

This document focuses on the availability of SFCs. For this reason let us assume that the Virtual Network Functions (VNFs) are already placed. Notice that all the arguments presented here can be easily extended to the case where this presumption does not hold and the placement of VNFs is to be decided.

2. Modeling availability constraints

Let α_v denote the availability of node $v \in V$, *i.e.*, the probability that node v is operational at a given time. Moreover, let A_k denote the required availability level for SFC $k \in K$. The availability requirement for SFC k is met if

$$\prod_{v \in P_k} \alpha_v \geq A_k,$$

where P_k is the set of nodes in the path used for routing SFC k.

*Corresponding author

Email address: rafael.colares@gmail.com (Rafael Colares)

If x_v^k is a binary variable representing whether or not SFC k is routed through node v, then the following linear inequalities can model the availability of SFCs.

$$\sum_{v \in S} x_v^k \leq |S| - 1 \qquad \forall\, k \in K, S \subseteq V : \prod_{v \in S} \alpha_v < A_k. \tag{1}$$

These inequalities however appear in exponential number and hence the question that naturally emerges is whether or not the associated separation problem can be solved in polynomial time. I still do not have an answer for it but it should be worth looking into it.

In any case, whenever variables x are integer, it is quite simple to check if the given solution satisfies inequalities (1) or not. One simply has to compute

$$\prod_{v \in V} (\alpha_v x_v^k)$$

for each $k \in K$ and check if it violates the proposed availability level A_k. Whenever it does, include the violated constraint

$$\sum_{v \in S} x_v^k \leq |S| - 1$$

where $S = \{v \in V : x_v^k = 1\}$.

3. Strong availability conflicts

A strong availability conflict may appear if two nodes are incompatible for a given SFC. More precisely, two nodes $u \in V$ and $v \in V$ in the network are said to be incompatible for SFC $k \in K$ if $\alpha_u \alpha_v < A_k$. This means that k cannot be routed through both nodes u and v, otherwise its availability requirement would be violated. Considering only incompatible nodes, inequalities (1) become

$$x_u^k + x_v^k \leq 1 \qquad \forall\, k \in K, u \in V, v \in V \text{ s.t. } \alpha_u \alpha_v < A_k \tag{2}$$

Let $G_k = (V'_k, E'_k)$ be the graph of availability conflicts associated with

SFC k, where $V'_k = V$ and E'_k is defined as follows. An edge (u, v) belongs to E'_k if and only if $\alpha_u \alpha_v < A_k$. Then, inequalities (2) can be rewritten as

$$x_u^k + x_v^k \leq 1 \qquad \forall\ k \in K, (u, v) \in E'_k \tag{3}$$

Notice that for each SFC k, inequalities (3) define an independent set problem. Therefore, the following well-known clique inequalities (4) and odd-hole inequalities (5) are valid.

$$\sum_{v \in C} x_v^k \leq 1 \qquad \forall\ C \subseteq V'_k : G[C] \text{ is a clique in } G_k. \tag{4}$$

$$\sum_{v \in H} x_v^k \leq \frac{|H| - 1}{2} \qquad \forall\ H \subseteq V'_k : G[H] \text{ is an odd-hole in } G_k. \tag{5}$$

4. Excessively restricted cases

In some scenarios where the availability required for each SFC is too high, it might be the case that it simply does not exist any routing configuration where the availability requirements are satisfied. In this case, it might be interesting to add some slack variable ϵ to the availability constraints (1) and penalize its value in the objective function.

5. Placing redundancy

In order to meet the availability requirements, one might choose additional backup paths in order to route the SFC. Therefore, for each SFC, the combinatorial structure one should look for is a collection of s-t paths, or more generally, a s-t flow.

Given a s-t flow $F = (V, A)$, the probability that an SFC can be routed from s to $v \in V$ – denoted by π_v – is the probability that v is available *and* there exists at least one available path until v.

$$\pi_v = \alpha_v \left(1 - \prod_{u \in \delta^-(v)} (1 - \pi_u) \right) \tag{6}$$

Therefore, the availability of a s-t flow (*i.e.*, π_t) can be computed in polynomial time (more precisely, in $\mathcal{O}(m+n)$) by computing π_v for each $v \in V$ in a Breadth-First Search (BFS) order. This statement is false. Instead might be able to compute π_t by making use of a Markov chain approach.

If the availability of the s-t flow for a given SFC is not sufficient, then the following infeasibility cut can be added to cutoff such infeasible solution.

$$\sum_{a \in A: \bar{f}_a^k = 0} f_a^k + \sum_{a \in A: \bar{f}_a^k = 1} (1 - f_a^k) \geq 1 \tag{7}$$

6. Routing Module

$$\sum_{a \in \delta^+(v)} r_{kiast} - \sum_{a \in \delta^-(v)} r_{kiast} = \begin{cases} z_{kist}, & \text{if } v = s \\ -z_{kist}, & \text{if } v = t \\ 0 & \text{otherwise} \end{cases} \quad \forall k \in K, i \in I^k, s \in V, t \in V, v \in V,$$

$$\sum_{a \in A} l_a r_{kiast} \leq L_{ki} \qquad \forall k \in K, i \in I^k, s \in V, t \in V$$

$$\sum_{i \in I^k} L_{ki} \leq L^k \qquad \forall k \in K$$

$$z_{kist} \leq x_{k(i-1)s} \qquad \forall k \in K, i \in I^k, s \in V, t \in V$$

$$z_{kist} \leq x_{kit} \qquad \forall k \in K, i \in I^k, s \in V, t \in V$$

$$z_{kist} \geq x_{k(i-1)s} + x_{kit} - 1 \qquad \forall k \in K, i \in I^k, s \in V, t \in V$$

$$r_{kiast} \leq z_{kist} \qquad \forall k \in K, i \in I^k, a \in A, s \in V, t \in V$$

References

How to model resiliency of Service Function Chains

Rafael Colares[a,*]

[a] *Orange Labs, 46 Avenue de la République, 92320 Châtillon, France*

Abstract

This is document provides some ideas on how to describe valid inequalities for the Resilient VNF Placement problem.

Keywords: Virtual Network Functions, Service Function Chain, resilience, combinatorial optimization, valid inequalities

1. Introduction

Given a set of nodes $V = \{v_1, \ldots, v_n\}$, let $0 < a(v) < 1$ denote the availability of each node $v \in V$. Notice that the probability that all nodes in $S \subseteq V$ fail simultaneously is given by

$$\prod_{v \in S} (1 - a(v)).$$

Now consider the following problem. Find the minimum number of nodes to be chosen so that the probability that all fail simultaneously is at most B. Let $c(V)$ denote the answer to such problem.

Without loss of generality, assume that

$$a(v_i) \geq a(v_{i+1}) \text{ for } i = 1, \ldots, n-1. \tag{1}$$

The problem can be easily solved by starting from an empty set S and sequentially adding node v_i to set S on the i-th iteration until the availability

*Corresponding author

Email address: rafael.colares@gmail.com (Rafael Colares)

condition is satisfied. Once the condition is satisfied, $c(V) = |S|$ answers the problem since (1) applies.

2. Valid inequalities

2.1. Minimum number of VNFs per section

The availability of a SFC chain is given by the product of its section's availability. It follows that if such availability must meet a certain level of reliability B, then each of its sections must meet the reliability level as well.

Therefore, a lower bound on the minimum number of VNFs that should be placed each section of the SFC chain is $c(V)$, which can be easily computed by considering the problem described in Section 1. The following inequality is hence valid.

$$\sum_{v \in V} x_v^{ik} \geq c(V) \qquad \forall k \in K, i \in I^k. \tag{2}$$

Recall that K is the set of SFCs, I^k is the set of sections of SFC $k \in K$, and the binary variable $x_v^{ik} = 1$ if the i-th VNF of SFC k is placed on node v, and $x_v^{ik} = 0$ otherwise.

Inequalities (2) are quite effective when the node availabilities are relatively uniform. However, it is easy to see that these constraints are of poor-quality when there is a large amplitude within node's availabilities. Take for instance the following trivial case. Let there be a single node $v' \in V$ for which $a(v') \geq A^k$ and for every other node $v \in V \setminus v'$ one has $a(v) << A^k$, where A^k denotes the availability level requested for SFC k. Then, inequalities (2) are the same as the original constraints (3). However if the VNF is not placed on node v', then many more placements are needed in order to ensure availability.

$$\sum_{v \in V} x_v^{ik} \geq 1 \qquad \forall k \in K, i \in I^k. \tag{3}$$

2.2. New valid inequality

Assuming (1) holds, let $V_k = \{v_i \in V : i \geq k\}$ denote the subset of nodes in V that have availability at most $a(v_k)$ for $k = 1, \dots n$. Moreover, considering the problem described in Section 1, let

$$c_{v_i} = \begin{cases} c(V_i), \text{ if } \prod_{v \in V_i}(1 - a(v)) \leq B, \text{ and} \\ c_{v_{i-1}}, \text{ otherwise.} \end{cases}$$

Finally let $C = \max_{v \in V} c_v$. Then the following inequality holds.

$$\sum_{v \in V}(C - c_v + 1)x_v^{ik} \geq C \qquad \forall k \in K, i \in I^k. \tag{4}$$

Proof of validity: Let S denote the set of nodes where a VNF is placed for an arbitrary feasible solution. Let v' be the node in S with smallest c_v. Then, by definition

$$|S| \geq c_{v'}. \tag{5}$$

Since $x_{v'}^{ik} = 1$, the left-hand side of constraints (4) equals

$$(C - c_{v'} + 1) + \sum_{v \in S \setminus v'}(C - c_v + 1).$$

Since $(C - c_v + 1) \geq 1$ for any $v \in V$, and from (5) one has that

$$\sum_{v \in S}(C - c_v + 1) \geq C,$$

which proves the validity of inequalities (4). □

References

How to model resiliency of Service Function Chains II

Rafael Colares[a,*]

[a] *Orange Labs, 46 Avenue de la République, 92320 Châtillon, France*

Abstract

This is document provides some ideas on how to describe valid inequalities for the Resilient VNF Placement problem.

Keywords: Virtual Network Functions, Service Function Chain, resilience, combinatorial optimization, valid inequalities

1. Introduction

Given a set of nodes $V = \{v_1, \ldots, v_n\}$, let $0 < a(v) < 1$ denote the availability of node $v \in V$. Without loss of generality, let us assume that nodes are ordered according to their availability, that is

$$a(v_i) > a(v_{i+1}) \text{ for } i = 1, \ldots, n-1. \tag{1}$$

The probability that all nodes in a subset $S \subseteq V$ fail simultaneously is given by

$$\prod_{v \in S} \Big(1 - a(v)\Big). \tag{2}$$

Consider now a Service Function Chain (SFC) composed of m Virtual Network Functions (VNFs). Let $\mathcal{S} = \{S_1, \ldots, S_m\}$ denote the VNF placement of such SFC, where each subset $S_i \subseteq V$, for $i = 1, \ldots, m$, represents the set of nodes where the i-th VNF of the given SFC can be processed. The probability that the i-th VNF of the considered SFC can be properly processed is the probability that at least one of the nodes in S_i is operational.

*Corresponding author

Email address: rafael.colares@gmail.com (Rafael Colares)

Such probability, denoted by $\alpha(S_i)$, can hence be calculated as follows:

$$\alpha(S_i) = 1 - \prod_{v \in S_i} \Big(1 - a(v)\Big). \tag{3}$$

For a SFC to be operational, all its VNFs must be properly processed. It follows that the availability $A(\mathcal{S})$ of a SFC whose VNF placement is $\mathcal{S}$ is given by

$$A(\mathcal{S}) = \prod_{S \in \mathcal{S}} \alpha(S) = \prod_{S \in \mathcal{S}} \left(1 - \prod_{v \in S} \Big(1 - a(v)\Big)\right). \tag{4}$$

Remark 1. *If a certain availability level B is required to be achieved by the SFC (i.e., $A(\mathcal{S}) \geq B$), then*

$$\alpha(S) \geq B \qquad \forall S \in \mathcal{S}$$

is a necessary condition that must be satisfied, since by definition $0 \leq \alpha(S) \leq 1$ for any $S \subseteq V$. Moreover, this can be extended to case where a subset of sections is considered, that is,

$$\prod_{S \in \mathcal{S}'} \alpha(S) \geq B \qquad \forall \mathcal{S}' \subseteq \mathcal{S}.$$

2. Minimum number of VNFs to be placed

In this section, let us consider the following combinatorial problem. Given a network $G = (V, E)$ and a SFC k composed of m VNF types, find the placement $\mathcal{S}$ inducing the highest possible availability while placing exactly $\eta \in \mathbb{N}$ VNFs.

Next we show that this problem can be solved in linear time. For this, let us first analyze the structure of an optimal placement $\mathcal{S}$.

Theorem 1. *If $\mathcal{S} = \{S_1, \ldots, S_m\}$ is an optimal placement, then every placement $\mathcal{S}'$ built from the permutation of sets $S_1, \ldots, S_m$ is also optimal.*

Proof. The availability function $A(\mathcal{S})$ defined by (4) is commutative over the elements of $\mathcal{S}$. □

Theorem 2. *If $\mathcal{S} = \{S_1, \ldots, S_m\}$ is an optimal placement, then each subset S_i is composed by the $|S_i|$ most available nodes, that is,*

$$S_i = \{v_1, \ldots, v_{|S_i|}\} \qquad \forall i \in \{1, \ldots, m\}.$$

Proof. Suppose there exists a set $S_i \in \mathcal{S}$ for which $S_i \neq \{v_1, \ldots, v_{|S_i|}\}$. Then let us construct a VNF placement with the same number of VNFs as follows. Let $\mathcal{S}^* = \{S_1^*, \ldots, S_m^*\}$, where $S_j^* = S_j$ for any $j \neq i$ and $S_i^* = \{v_1, \ldots, v_{|S_i|}\}$. Since by definition $\alpha(S_i) < \alpha(S_i^*)$, one has $A(\mathcal{S}) < A(\mathcal{S}^*)$ and hence $\mathcal{S}$ is not optimal. □

Theorem 3. *If $\mathcal{S} = \{S_1, \ldots, S_m\}$ is an optimal placement, then difference between the number of VNFs in any two sets S_i and S_j within $\mathcal{S}$ is at most one, that is,*

$$|S_i| - |S_j| \leq 1 \qquad \forall i, j \in \{1, \ldots, m\}.$$

Proof. The proof is done by contradiction. Suppose that $\mathcal{S} = \{S_1, \ldots, S_m\}$ is an optimal placement where there exists i and j for which $|S_i| - |S_j| \geq 2$. From Theorem 2, S_i and S_j are composed of the most available nodes and hence $S_j \subset S_i$. Let $v' \in S_i$ be the least available node in S_i. By definition, $v' \notin S_j$. Let us construct a VNF placement $\mathcal{S}^*$ with the same number of VNFs as follows. Let $\mathcal{S}^* = \{S_1^*, \ldots, S_m^*\}$, where $S_k^* = S_k$ for any $k \in \{1, \ldots, m\} \setminus \{i, j\}$, $S_j^* = S_j \cup v'$ and $S_i^* = S_i \setminus v'$. Notice that since v' is the least available node in S_i,

$$\alpha(S_i) > \alpha(S_i^*) > \alpha(S_j^*) > \alpha(S_j).$$

Moreover,

$$\begin{aligned}\alpha(S_i^*) =& 1 - \frac{\prod_{v\in S_i}(1-a(v))}{1-a(v')}\\ =& \frac{1-a(v')-\prod_{v\in S_i}(1-a(v))}{1-a(v')}\\ =& \frac{\alpha(S_i)-a(v')}{1-a(v')},\end{aligned} \tag{5}$$

and

$$\begin{aligned}\alpha(S_j^*) =& 1 - \left[\left(\prod_{v\in S_j}(1-a(v))\right)\big(1-a(v')\big)\right]\\ =& \alpha(S_j) + \left[a(v')\left(\prod_{v\in S_j}(1-a(v))\right)\right]\\ =& \alpha(S_j) + \big[a(v')(1-\alpha(S_j))\big].\end{aligned} \tag{6}$$

Next we show that $A(\mathcal{S}^*) > A(\mathcal{S})$, a contradiction since $\mathcal{S}$ is supposed to be optimal. By definition,

$$A(\mathcal{S}^*) = A(\mathcal{S})\frac{\alpha(S_j^*)\alpha(S_i^*)}{\alpha(S_j)\alpha(S_i)}.$$

Using (5) and (6),

$$\begin{aligned}A(\mathcal{S}^*) =& A(\mathcal{S})\frac{\alpha(S_j^*)\Big[\alpha(S_i)-a(v')\Big]}{\alpha(S_j)\alpha(S_i)(1-a(v'))}\\ =& A(\mathcal{S})\frac{\alpha(S_i)\alpha(S_j)+\alpha(S_i)\Big(a(v')\big(1-\alpha(S_j)\big)\Big)-a(v')\alpha(S_j^*)}{\alpha(S_j)\alpha(S_i)(1-a(v'))}\\ =& A(\mathcal{S})\left(1+\frac{\alpha(S_i)\alpha(S_j)a(v')+\alpha(S_i)\Big(a(v')\big(1-\alpha(S_j)\big)\Big)-a(v')\alpha(S_j^*)}{\alpha(S_j)\alpha(S_i)(1-a(v'))}\right)\\ =& A(\mathcal{S})\left(1+\frac{\alpha(S_i)a(v')-a(v')\alpha(S_j^*)}{\alpha(S_j)\alpha(S_i)(1-a(v'))}\right).\end{aligned}$$

Since $\alpha(S_i) > \alpha(S_j^*)$ and $0 < a(v') < 1$, one has

$$A(\mathcal{S}^*) > A(\mathcal{S}),$$

which concludes the proof. □

2.1. Chain Cover Inequalities

Let $\eta(V, m, B) \in \mathbb{N}$ denote the minimum number of VNFs required to be installed within the node-set V so that a SFC composed of m sections can meet the availability requirement B. Next, we show how to compute $\eta(V, m, B)$.

For any integer $0 \leq j \leq n$, let $S^j = \{v_1, \ldots, v_j\}$ denote the set composed by the j most available nodes in V. Consider the VNF placement $\mathcal{S}^j = \{S_1^j, \ldots, S_m^j\}$ obtained by assigning the j most available nodes to each VNF. Finally, let j^* be the smallest integer for which $\mathcal{S}^{j^*}$ satisfies a given availability requirement $0 < B < 1$. It follows from Theorem 2 and Theorem 3 that

$$m(j^* - 1) \leq \eta(V, m, B) \leq mj^*.$$

Using Theorem 1, $\eta(V, m, B)$ can now be easily computed with the following algorithm.

Algorithm 1: Computation of $\eta(V, m, B)$

Result: $\eta(V, m, B)$

Let $\mathcal{S} = \mathcal{S}^{j^*-1}$, and $i = 0$;

while $A(\mathcal{S}) < B$ **do**

  $i \leftarrow i + 1$;

  Add the j^*-th most available node in V to S_i;

end

Return $\eta(V, m, B) = m(j^* - 1) + i$;

It follows from Algorithm 1 that the following inequalities are valid.

$$\sum_{i \in I^k} \sum_{v \in V} x_v^{ik} \geq \eta(V, |I^k|, A^k) \qquad \forall k \in K.$$

This can be further extended using Remark 1, which gives rise the following Chain Cover Inequalities.

$$\sum_{i \in Q} \sum_{v \in V} x_v^{ik} \geq \eta(V, |Q|, A^k) \qquad \forall k \in K, Q \subseteq I^k. \tag{7}$$

Theorem 4. *The Chain Cover inequalities* (7) *are valid.*

Proof. From the definition of $\eta(V, |Q|, A^k)$. □

2.2. Node Cover Inequalities

Given a SFC k requiring a certain availability level B, we know from Remark 1 that the availability of each section $i \in I^k$ must meet the required availability B, that is, $\alpha(S_i) \geq B$. Moreover, from Algorithm 1 we know how to compute the minimum number of nodes required to receive the i-th VNF of SFC k in order to achieve the requested availability level B, that is $\eta(V, 1, B)$. Notice however that $\eta(V, 1, B)$ is computed following the principles of Theorem 2, that is, only the most available nodes in V are actually taken into account. If for any particular reason (*e.g.*, elevated cost, insufficient capacity, etc) some of these nodes are banned from being used, such lower bound on the number of nodes receiving the VNF might increase. We next focus on this special case in order to derive a new family of valid inequalities.

Let $U \subseteq V$ denote the subset of nodes that are not banned from receiving a given VNF. Then, $\eta(U, 1, B)$ refers to the minimum number of nodes required to receive the VNF so that the availability level B is reached. With this in mind, consider the following Node Cover inequalities.

$$\sum_{v \in V \setminus U} \eta(U, 1, A^k) x_v^{ik} + \sum_{v \in U} x_v^{ik} \geq \eta(U, 1, A^k) \qquad \forall k \in K, i \in I^k, U \subseteq V. \tag{8}$$

Theorem 5. *The Node Cover inequalities* (8) *are valid.*

Proof. Let S_i denote the set of nodes where the i-th VNF of SFC k is placed for an arbitrary feasible solution. That is, $S_i = \{v \in V : x_v^{ik} = 1\}$. If

there exists a node $v \in S_i$ such that $v \in V \setminus U$, then the inequality is clearly satisfied. Hence, suppose all nodes in S_i belong to U, that is, $S_i \cap U = S_i$. In this case, $|S_i| \geq \eta(U, 1, A^k)$ and therefore the inequality is also verified. $\square$

Notice that even if the bound provided by Algorithm 1 is tight, such inequalities might become quite loose when there exists a node $v \in S_i$ that belongs to $V \setminus U$. For this reason, we next propose a lifted version of the Node Cover inequalities (8).

For this, let us first consider the following problem. Knowing that a VNF is placed on a given node $v' \in V$, find the minimum number of backup replicas, denoted by $\beta(v', V, B)$, that need to be installed in parallel within the node-set V so that a certain availability requirement B is reached. Remark that such problem is only slightly different than the previously studied computation of $\eta(V, 1, B)$. Indeed, it suffices to oblige node v' to be part of the placement in the Algorithm 1. Algorithm 2 details the procedure.

Algorithm 2: Computation of $\beta(v', V, B)$

Result: $\beta(v', V, B)$
Let $S = \{v'\}$, and $i = 0$;
while $\alpha(S) < B$ **do**
    $i \leftarrow i + 1$;
    Add the i-th most available node in $V \setminus v'$ to S;
end
Return $\beta(v', V, B) = i$;

Finally, let us denote by $U_v \subseteq V$, for each node $v \in V$, the subset of nodes in V that are at most as available as v, that is,

$$U_v = \Big\{ u \in V : a(u) \leq a(v) \Big\}.$$

The Lifted Node Cover inequalities are defined as follows.

$$\sum_{v \in V \setminus U} c_v x_v^{ik} + \sum_{v \in U} x_v^{ik} \geq \eta(U, 1, A^k) \qquad \forall k \in K, i \in I^k, U \subseteq V, \tag{9}$$

where $c_v = \max\Big(\eta(U, 1, A^k) - \beta(v, U \cup U_v, A^k), 1\Big)$.

Theorem 6. *The Lifted Node Cover inequalities* (9) *are valid.*

Proof. Let S_i denote the set of nodes where the i-th VNF of SFC k is placed for an arbitrary feasible solution. That is, $S_i = \{v \in V : x_v^{ik} = 1\}$. If $S_i \subseteq U$, then the inequality is satisfied since, by definition, $|S_i| \geq \eta(U, 1, A^k)$. Hence, let us focus on the case where $S_i \nsubseteq U$. For this, let v' denote the most available node in S_i that does not belong to U. Then, $S_i \subseteq U \cup U_{v'}$ and hence $|S_i \setminus v'| \geq \beta(v', U \cup U_{v'}, A^k)$. Since $c_{v'} \geq \eta(U, 1, A^k) - \beta(v, U \cup U_v, A^k)$ and every other coefficient is at least 1, the inequality is verified. □

2.3. Generalized Cover Inequalities

Chain Cover inequalities (7) concern the non-linearity of the SFC's availability function that appears from the chaining of VNFs. Node Cover inequalities (8), (9) treat the availability non-linearity arising rather from the parallel placement of VNFs over heterogeneous nodes. Next we combine these two ideas into a single large family of valid inequalities. The Generalized Cover inequalities are defined as follows.

$$\sum_{i \in Q} \sum_{v \in V \setminus U} \eta(U, |Q|, A^k) x_v^{ik} + \sum_{i \in Q} \sum_{v \in U} x_v^{ik} \geq \eta(U, |Q|, A^k) \qquad \forall k \in K, Q \subseteq I^k, U \subseteq V. \tag{10}$$

Theorem 7. *The Generalized Cover inequalities* (10) *are valid.*

Proof. Let $\mathcal{S} = \{S_1, \ldots, S_m\}$ denote the VNF placement of SFC k for an arbitrary feasible solution. That is, $S_i = \{v \in V : x_v^{ik} = 1\}$, for $i = 1, \ldots, m$. If there exists a node $v \in S_i$, for any $i \in Q$, such that $v \in V \setminus U$, then the inequality is clearly satisfied. Hence, suppose all nodes in S_i, for

$i \in Q$, belong to U. In this case, at least $\eta(U, |Q|, A^k)$ VNFs must be placed and hence $\sum_{i \in Q} \sum_{v \in U} x_v^{ik} \geq \eta(U, |Q|, A^k)$. It follows that the inequality is valid. □

A question that immediately arises is whether or not such inequalities can be lifted like the Node Cover inequalities...

3. Section failures

If all the VNFs installed in a section of a given SFC fail simultaneously, the whole SFC crashes. Therefore, the availability of each section should reach at least the SFC's required availability (*c.f.*, Remark 1). Based on such statement, consider the following Section Failure Inequalities:

$$\sum_{v \in V} \Big(-\log\big(1 - a(v)\big)\Big) x_v^{ik} \geq -\log\big(1 - A^k\big) \qquad \forall k \in K, i \in I^k. \tag{11}$$

Theorem 8. *The Section Failure inequalities* (11) *are valid.*

Proof. The proof is done by contradiction. Suppose there exists a feasible solution $\bar{x}$ that violates the Section Failure inequalities for some $k \in K, i \in I^k$, that is

$$\sum_{v \in V} \Big(-\log\big(1 - a(v)\big)\Big) \bar{x}_v^{ik} < -\log\big(1 - A^k\big). \tag{12}$$

Let $S = \{v \in V : \bar{x}_v^{ik} = 1\}$. Then, the violated constraint (12) reduces to

$$\sum_{v \in S} \Big(-\log\big(1 - a(v)\big)\Big) < -\log\big(1 - A^k\big).$$

From the fundamental property of logarithms that states $\log(a) + \log(b) = \log(ab)$, the inequality can be rewritten as

$$\log\Big(\prod_{v \in S} \big(1 - a(v)\big)\Big) > \log\big(1 - A^k\big),$$

which is only true if

$$\prod_{v \in S} \Big(1 - a(v)\Big) > 1 - A^k.$$

This is a contradiction since $\bar{x}$ is said to be feasible. □

References

Nested Column Generation Approach

Rafael Colares[a,*]

[a] *Orange Labs, 46 Avenue de la République, 92320 Châtillon, France*

Abstract

This is document provides some ideas on how to model the Resilient VNF Placement and SFC Routing problem through a Nested Column Generation Framework.

Keywords: Virtual Network Functions, Service Function Chain, resilience, combinatorial optimization, column generation

1. Instance description

Let $G = (V, A)$ be a directed, loopless, connected graph. Each node $v \in V$ has a capacity $c_v \in \mathbb{R}$, a unitary processing cost $p_v \in \mathbb{R}$, and an availability $0 < a_v \leq 1$, (*i.e.*, a failure probability of $1 - a_v$). Each each link $(u, v) \in A$ has a length d_{uv} and a capacity B_{uv}. Moreover let $\mathcal{F}$ be the set of VNF types, where each VNF $f \in \mathcal{F}$ has a resource consumption $r_f \in \mathbb{R}$. Finally, let K be the set of SFC demands, where each demand $k \in K$ is defined by (i) an origin $o_k \in V$ and a destination $d_k \in V$, (ii) a bandwidth $b_k \in \mathbb{R}$, (iii) a required availability A^k, and (iv) an ordered set of VNFs $F^k \subseteq \mathcal{F}$.

2. Solution Description

2.1. Path configuration definition

Let $P = (\pi, S)$ denote a *path configuration* for a given SFC $k \in K$, where π is an o_k-d_k path within which the SFC can be routed and $S \subseteq V$ is its

*Corresponding author
Email address: rafael.colares@gmail.com (Rafael Colares)

VNF placement, *i.e.*, the set of nodes processing the SFC's VNFs along path π. The availability of a path configuration $P = (\pi, S)$, denoted by a_P, is given by

$$a_P = \prod_{v \in S} a_v.$$

Path configuration feasibility. A path configuration $P = (\pi, S)$ is said to be feasible for an SFC $k \in K$ if and only if (i) the length of path π, denoted by $l(\pi)$, is at most the SFC's maximum latency L^k, and (ii) each of the SFC required VNFs is placed along the path π, that is, $S \subseteq V(\pi)$ and $|S| = |F^k|$.

2.2. SFC configuration definition

Let $C = \{P_1, \ldots, P_q\}$ denote an *SFC configuration* for a given SFC $k \in K$, where $P_i = (\pi_i, S_i)$ is a path configuration. An SFC configuration is a set of path configurations that can route the SFC and satisfy its latency and VNF requirements.

SFC Configuration feasibility. An SFC configuration $C = \{P_1, \ldots, P_q\}$ is said to be feasible for SFC k if and only if (i) every path configuration in C is feasible, (ii) the path configurations are *VNF-disjoint*, that is, sets S_i are disjoint for $1 \leq i \leq q$, (iii) the SFC required availability A^k is satisfied, that is,

$$1 - \prod_{P \in C} \Big(1 - a_P\Big) \geq A^k.$$

2.3. Network deployment definition

A network deployment consists of finding a feasible SFC configuration for each SFC $k \in K$ such that link and node capacities are respected.

3. Nested Column Generation

3.1. First level (Master problem)

Let $\mathcal{C}$ denote the set of available SFC configurations. For each SFC configuration $C \in \mathcal{C}$, each node $v \in V$ and each VNF $f \in \mathcal{F}$, let α_C^{fv} be a binary parameter stating whether or not the VNF f is placed on node v

within the SFC configuration C. For each SFC configuration $C \in \mathcal{C}$, and each arc $a \in A$, let β_C^a be a binary parameter stating whether or not arc a is used within the SFC configuration C. Finally for each $C \in \mathcal{C}$ and each $k \in K$, let λ_C^k be a binary variable stating whether or not the SFC configuration C is applied to SFC k.

The first level optimization program can be described as follows.

$$\min \sum_{v \in V} \sum_{f \in \mathcal{F}} (r_f p_v) y_v^f \tag{1}$$

subject to

$$\sum_{C \in \mathcal{C}} \lambda_C^k = 1 \qquad \forall k \in K, \tag{2}$$

$$\sum_{C \in \mathcal{C}} \sum_{k \in K} \sum_{f \in F^k} \alpha_C^{fv} b^k r_f \lambda_C^k \leq C_v \qquad \forall v \in V, \tag{3}$$

$$\sum_{C \in \mathcal{C}} \sum_{k \in K} \beta_C^a b^k \lambda_C^k \leq B_a \qquad \forall a \in A, \tag{4}$$

$$\alpha_C^{fv} \lambda_C^k \leq y_v^f \qquad \forall k \in K, f \in F^k, C \in \mathcal{C}, v \in V, \tag{5}$$

$$\lambda_C^k \in \{0,1\} \qquad \forall C \in \mathcal{C}, k \in K. \tag{6}$$

Inequalities (2) state that exactly one SFC configuration should be assigned to each SFC. Inequalities (3) and (4) ensure that node and link capacities are, respectively, satisfied. Inequalities (5) impose that a VNF can only be assigned to an SFC if it is already placed.

3.2. Second level (Subproblem I)

The second level consists of finding the best SFC configuration for an SFC $k \in K$ to be included in the first level.

Let $\mathcal{P}$ denote the set of available path configurations that can be part of the SFC configuration. For each path configuration $P \in \mathcal{P}$, let β_P^v be a binary parameter stating whether or not a VNF is processed in node v. Finally for each $P \in \mathcal{P}$, let γ_P be a binary variable stating whether or not the path configuration P belongs to the SFC configuration.

The second level optimization program associated with a given SFC $k \in K$ can be described as follows.

$$\min g(\gamma) \tag{7}$$

subject to

$$\sum_{P \in \mathcal{P}} \gamma_P \geq 1, \tag{8}$$

$$\sum_{P \in \mathcal{P}} \beta_P^v \gamma_P \leq 1 \qquad \forall v \in V \tag{9}$$

$$1 - \prod_{P \in \mathcal{P}} \Big(1 - a_P \gamma_P\Big) \geq A^k, \tag{10}$$

$$\gamma_P \in \{0, 1\} \qquad \forall P \in \mathcal{P} \tag{11}$$

Inequalities (8) states that at least one path configuration should be assigned the SFC configuration. Inequalities (9) ensure that the paths in the SFC configuration are VNF-disjoint. Inequalities (10) guarantee that the required availability level is reached.

Inequality (10) can clearly be rewritten as

$$\prod_{P \in \mathcal{P}} \Big(1 - a_P \gamma_P\Big) \leq 1 - A^k. \tag{12}$$

Consider now the linearized version of such constraint as follows:

$$\sum_{P \in \mathcal{P}} \log(1 - a_P) \gamma_P \leq \log(1 - A^k). \tag{13}$$

Theorem 1. *The inequality* (13) *is satisfied if and only if inequality* (12) *is also satisfied.*

Proof. Let $C = \{P \in \mathcal{P} : \gamma_P = 1\}$ be an SFC configuration. Suppose C satisfies the availability inequality (12). Then,

$$\prod_{P \in C} \Big(1 - a_P\Big) \leq 1 - A^k,$$

and hence,

$$\prod_{P \in C} \log\Big(1 - a_P\Big) \leq \log(1 - A^k).$$

From the fundamental property of logarithms that states $\log(a)+\log(b) = \log(ab)$, one has that

$$\sum_{P \in C} \log(1 - a_P) = \prod_{P \in C} \log\Big(1 - a_P\Big) \leq \log(1 - A^k),$$

and therefore inequality (13) is satisfied.

Suppose now that C does not satisfy the availability inequality (12). Then,

$$\prod_{P \in C} \Big(1 - a_P\Big) > 1 - A^k,$$

and the same arguments applied in the first part of this proof hold for proving that inequality (13) is not satisfied. □

Corollary 1. *The non-linear inequality* (10) *can be replaced by the linear inequality* (13).

Remarks:

- Odd-hole inequalities may be applied here for reinforcing disjunction.
- Cover inequalities may be applied here for reinforcing availability.

3.3. Third level (Subproblem II)

The third level consists of finding the best path configuration to be included in the available path configurations $\mathcal{P}$ for the second level.

Let $V' = V \setminus \{o_k, d_k\}$. Moreover, let $I = \{0, \ldots, |F^k| + 1\}$ and $I' = \{1, \ldots, |F^k| + 1\}$. For each arc $a \in A$ and each $i \in I'$, let x_a^i be a binary variable stating whether or not the arc a belongs to the path between the $(i-1)$-th VNF and the i-th VNF. For each $v \in V$ and each $i \in I$, let y_v^i be a binary variable stating whether or not the i-th VNF is processed on node v along the path. The artificial VNFs 0 and $|F^k| + 1$ are, respectively, placed on nodes o_k and d_k.

The third level optimization program associated with a given SFC $k \in K$ can be described as follows.

$$\min h(x, y) \tag{14}$$

subject to

$$\sum_{a \in \delta^+(v)} x_a^i - \sum_{a \in \delta^-(v)} x_a^i = y_v^{i-1} - y_v^i \qquad \forall v \in V, i \in I', \tag{15}$$

$$\sum_{i \in I'} \sum_{a \in A} x_a^i \leq L^k, \tag{16}$$

$$\sum_{v \in V} y_v^i = 1 \qquad \forall i \in I, \tag{17}$$

$$y_{o_k}^0 = 1, \tag{18}$$

$$y_{d_k}^{|F^k|+1} = 1, \tag{19}$$

$$x_a \in \{0, 1\} \qquad \forall a \in A, \tag{20}$$

$$y_v^i \in \{0, 1\} \qquad \forall v \in V, i \in I. \tag{21}$$

Inequalities (15) state that a path between every two consecutive VNFs must be selected. Inequality (16) ensures that the SFC maximum latency is verified. Inequalities (17) specify that each required VNF must be placed on a node. Inequalities (18) and (19) fix the artificial VNFs on the origin and destination nodes, respectively.

4. 'Standard' Column Generation

Next we derive an standard column generation approach by combining the first and second levels presented previously.

For this, let $\mathcal{P}^k$ denote the set of available path configurations that can route SFC k. For each path configuration $P \in \mathcal{P}^k$ and each node $v \in V$, let β_P^v be a binary parameter stating whether or not a VNF is processed on node v along the path configuration P. For each path configuration $P \in \mathcal{P}^k$ and each arc $a \in A$, let γ_P^a be a binary parameter stating whether or not arc a is part of the path configuration P. For each path configuration $P \in \mathcal{P}^k$,

each node $v \in V$ and each VNF $f \in \mathcal{F}$, let α_P^{fv} be a binary parameter stating whether or not VNF f can be processed on node v within the path configuration P.

Finally for each $P \in \mathcal{P}^k$, let λ_P^k be a binary variable stating whether or not the path configuration P belongs to the configuration of SFC k. For each node $v \in V$ and each VNF $f \in \mathcal{F}$, let y_v^f be a binary variable stating whether or not the VNF f is placed on node v. The path formulation is described below.

$$\min \sum_{v \in V} \sum_{f \in \mathcal{F}} (r_f p_v) y_v^f \tag{22}$$

subject to

$$\sum_{P \in \mathcal{P}^k} \lambda_P^k \geq 1 \qquad \forall k \in K, \tag{23}$$

$$\sum_{P \in \mathcal{P}^k} \beta_P^v \lambda_P^k \leq 1 \qquad \forall k \in K, v \in V, \tag{24}$$

$$\sum_{P \in \mathcal{P}^k} \log(1 - a_P) \lambda_P^k \leq \log(1 - A^k) \qquad \forall k \in K, \tag{25}$$

$$\sum_{k \in K} \sum_{P \in \mathcal{P}^k} \gamma_P^a b^k \lambda_P^k \leq B_a \qquad \forall a \in A, \tag{26}$$

$$\sum_{k \in K} \sum_{P \in \mathcal{P}^k} \sum_{f \in F^k} \alpha_P^{fv} b^k r_f \lambda_P^k \leq C_v \qquad \forall v \in V, \tag{27}$$

$$\alpha_P^{fv} \lambda_P^k \leq y_v^f \qquad \forall k \in K, f \in F^k, P \in \mathcal{P}^k, v \in V, \tag{28}$$

$$\lambda_P^k \in \{0, 1\} \qquad \forall k \in K, P \in \mathcal{P}^k, \tag{29}$$

$$y_v^f \in \{0, 1\} \qquad \forall v \in V, f \in \mathcal{F}. \tag{30}$$

References

An arc formulation for VNF-disjoint resilient SFC routing

Rafael Colares[a,*]

[a] *Orange Labs, 46 Avenue de la République, 92320 Châtillon, France*

Abstract

This is document provides some ideas on how to model the Resilient VNF Placement and SFC Routing problem through a compact MIP formulation.

Keywords: Virtual Network Functions, Service Function Chain, resilience, combinatorial optimization, MIP formulation

1. Instance description

Let $G = (V, A)$ be a directed, loopless, connected graph. Each node $v \in V$ has a capacity $c_v \in \mathbb{R}$, a unitary processing cost $p_v \in \mathbb{R}$, and an availability $0 < a_v \leq 1$, (*i.e.*, a failure probability of $1 - a_v$). Each each link $(u, v) \in A$ has a length d_{uv} and a capacity B_{uv}. Moreover let $\mathcal{F}$ be the set of VNF types, where each VNF $f \in \mathcal{F}$ has a resource consumption $r_f \in \mathbb{R}$. Finally, let K be the set of SFC demands, where each demand $k \in K$ is defined by (i) an origin $o_k \in V$ and a destination $d_k \in V$, (ii) a bandwidth $b_k \in \mathbb{R}$, (iii) a maximum latency $L^k \in \mathbb{R}$, (iv) a required availability A^k, and (v) an ordered set of VNFs $F^k \subseteq \mathcal{F}$.

2. Solution Description

2.1. Path configuration definition

Given an SFC $k \in K$, let $I^k = \{1, \ldots, |F_k|\}$. For each SFC $k \in K$, one has to find an o_k-d_k path within which k can be routed. Such path can

*Corresponding author
Email address: rafael.colares@gmail.com (Rafael Colares)

be partitioned into sections. Let $\bar{I}^k = \{1, \ldots, |F_k| + 1\}$ denote the set of sections the SFC has to go through, where section 1 refers to the subpath between o_k and the node processing the first VNF, section $2 \leq i \leq |F^k|$ refers to the subpath between the nodes where the $(i-1)$-th and the i-th VNFs are processed, and section $|F^k| + 1$ refers to the subpath between the node processing the last VNF and d_k.

Let $P = \{\pi_i : i \in \bar{I}^k\}$ denote a *path configuration* for a given SFC $k \in K$, where π_i is the path of section i. Let $S(P) \subseteq V$ denote the VNF placement of P, *i.e.*, the set of nodes processing the SFC's VNFs along path P. The availability of P, denoted by a_P, is given by

$$a_P = \prod_{v \in S(P)} a_v.$$

Path configuration feasibility. A path configuration P is said to be feasible for an SFC $k \in K$ if and only if the length of P, denoted by $l(P)$, is at most the SFC's maximum latency L^k.

2.2. SFC configuration definition

Let $C = \{P_1, \ldots, P_q\}$ denote an *SFC configuration* for a given SFC $k \in K$, where P_i is the i-th path configuration routing the SFC.

SFC Configuration feasibility. An SFC configuration $C = \{P_1, \ldots, P_q\}$ is said to be feasible for SFC k if and only if (i) every path configuration in C is feasible, (ii) the path configurations are *VNF-disjoint*, that is, sets $S(P_i)$ are disjoint for $1 \leq i \leq q$, (iii) the SFC required availability A^k is satisfied, that is,

$$1 - \prod_{P \in C} \Big(1 - a_P\Big) \geq A^k.$$

Notice that since the path configurations are required to be VNF-disjoint and there is at least one node processing a VNF within any path configuration, it follows that $|V|$ is a trivial upper bound on the number of path configurations within an SFC configuration.

2.3. Network deployment definition

An optimal network deployment consists of finding a feasible SFC configuration for each SFC $k \in K$ such that link and node capacities are respected and the total VNF placement cost is minimum.

3. A compact MIP formulation

3.1. Variables

Let f^k_{aip} be a binary variable stating whether or not arc $a \in A$ is used for routing the i-th section of the p-th path configuration of the SFC configuration assigned to SFC $k \in K$.

Let x^k_{vip} be a binary variable stating whether or not the i-th VNF of SFC $k \in K$ is processed on node $v \in V$ within its p-th path configuration.

Moreover, let y^k_{vp} be a binary variable stating whether or not any VNF of SFC k is processed on node $v \in V$ within its p-th path configuration, and let

$$z^f_v = \begin{cases} 1, \text{ if VNF} f \in \mathcal{F} \text{ is placed on node } v \in V \\ 0, \text{ otherwise.} \end{cases}$$

Finally, let α^k_p be an auxiliary binary variable stating whether or not the p-th path configuration is used for routing SFC $k \in K$.

3.2. Objective Function

The goal is to minimize the total cost of VNF placements. Therefore the objective function can be described by

$$\min \sum_{v \in V} \sum_{f \in \mathcal{F}} (p_v r_f) z^f_v, \tag{1}$$

where $p_v \in \mathbb{R}$ is the unitary processing cost of node $v \in V$ and $r_f \in \mathbb{R}$ is the resource consumption of VNF $f \in \mathcal{F}$.

3.3. Routing Constraints

Flow conservation constraints are used for modeling paths for each section of a path configuration used.

$$\sum_{a\in\delta^+(o_k)} f^k_{aip} - \sum_{a\in\delta^-(o_k)} f^k_{aip} = \alpha^k_p - x^k_{o_k ip} \qquad \forall k \in K, 1 \le p \le |V|, i = 1 \quad (2)$$

$$\sum_{a\in\delta^+(v)} f^k_{aip} - \sum_{a\in\delta^-(v)} f^k_{aip} = -x^k_{vip} \qquad \forall k \in K, 1 \le p \le |V|, v \in V \setminus o_k, i = 1 \quad (3)$$

$$\sum_{a\in\delta^+(v)} f^k_{aip} - \sum_{a\in\delta^-(v)} f^k_{aip} = x^k_{v(i-1)p} - x^k_{vip} \qquad \forall k \in K, 1 \le p \le |V|, v \in V, 2 \le i \le |F^k| \quad (4)$$

$$\sum_{a\in\delta^+(v)} f^k_{aip} - \sum_{a\in\delta^-(v)} f^k_{aip} = x^k_{v(i-1)p} \qquad \forall k \in K, 1 \le p \le |V|, v \in V \setminus d_k, i = |F^k|+1 \quad (5)$$

$$\sum_{a\in\delta^+(d_k)} f^k_{aip} - \sum_{a\in\delta^-(d_k)} f^k_{aip} = x^k_{d_k(i-1)p} - \alpha^k_p \qquad \forall k \in K, 1 \le p \le |V|, i = |F^k|+1 \quad (6)$$

3.4. Capacity Constraints

Node capacities must be satisfied.

$$\sum_{k\in K} \sum_{p=1}^{|V|} \sum_{i\in I^k} b^k r_{f^k_i} x^k_{vip} \le C_v \qquad \forall v \in V. \quad (7)$$

Link capacities must be satisfied.

$$\sum_{k\in K} \sum_{p=1}^{|V|} \sum_{i\in I^k} b^k f^k_{aip} \le B_a \qquad \forall a \in A. \quad (8)$$

3.5. Latency Constraints

The length of each path configuration must be at most the SFC's maximum latency.

$$\sum_{a \in A} \sum_{i \in \bar{I}^k} d_a f^k_{aip} \leq L^k \alpha^k_p \qquad \forall k \in K, p \in \{1, \ldots, |V|\}. \tag{9}$$

3.6. Linking constraints

A path configuration is used for routing SFC $k \in K$ if and only if each of its VNFs is processed on exactly one node. This can be expressed as follows.

$$\sum_{v \in V} x^k_{vip} = \alpha^k_p \qquad \forall k \in K, i \in I^k, 1 \leq p \leq |V|. \tag{10}$$

If a VNF is processed on node $v \in V$, within the p-th path configuration of SFC $k \in K$, then variable y must be activated.

$$x^k_{vip} \leq y^k_{vp} \leq \alpha^k_p \qquad \forall k \in K, 1 \leq p \leq |V|, i \in I^k, v \in V. \tag{11}$$

A VNF can only be assigned to an SFC if it is already placed.

$$x^k_{vip} \leq z_v^{f^i_k} \qquad \forall k \in K, 1 \leq p \leq |V|, i \in I^k, v \in V.$$

Since VNF disjunction applies, these inequalities can be reinforced as follows.

$$\sum_{p=1}^{|V|} x^k_{vip} \leq z_v^{f^i_k} \qquad \forall k \in K, i \in I^k, v \in V. \tag{12}$$

3.7. VNF-disjoint constraints

The path configurations of each SFC must be VNF-disjoint.

$$\sum_{p=1}^{|V|} y^k_{vp} \leq 1 \qquad \forall k \in K, v \in V. \tag{13}$$

3.8. Availability constraints

The availability of each SFC $k \in K$ must meet its required service level A^k.

$$\prod_{p=1}^{|V|} \left(1 - \prod_{v \in V} \left(\alpha_p^k - (1 - a_v) y_{vp}^k\right)\right) \leq 1 - A^k \qquad \forall k \in K. \tag{14}$$

These inequalities are clearly non-linear. An easy way to linearize it is to replace them by an exponential number of linear inequalities as follows.

Let $S^k = \{(v, p) \in V \times P : y_{vp}^k = 1\}$ denote the VNF placement of an SFC $k \in K$. Then, the availability of such placement is given by

$$A(S^k) = 1 - \prod_{p=1}^{|V|} \left(1 - \prod_{v \in V : (v,p) \in S^k} a_v\right). \tag{15}$$

It follows that the following inequalities models the availability constraints

$$\sum_{(v,p) \in S} (1 - y_{vp}^k) + \sum_{(v,p) \notin S} y_{vp}^k \geq 1 \qquad \forall k \in K, S \subseteq V \times P : A(S) < A^k. \tag{16}$$

Such inequalities however appear in exponential number and must hence be treated on demand.

4. New Valid Inequalities

For each SFC $k \in K$, we look for a set of VNF-disjoint paths ensuring the required availability A^k.

Remark 1. *If at least one section of each selected path fails, then the SFC crashes.*

From remark 1, it follows that if an availability A^k is required, then the probability that at least one section of each selected path fails must be at most $1 - A^k$.

Theorem 1. *The following inequalities are valid*

$$\sum_{v\in V}\sum_{(i,p)\in S}\big(\log(1-a_v)\big)x_{vip^k} \le \log(1-A^k) \qquad \forall k\in K, S\subseteq I^k\times P : \exists (i,p)\in S\ \forall p\in P \tag{17}$$

Proof. Let $\bar{x}$ be a feasible solution and $W^k = \{(v,i,p)\in V\times I^k\times P : \bar{x}^k_{vip} = 1\}$. Then,

$$\sum_{(v,i,p\in W^k}\big(\log(1-a_v)\big) \le \log(1-A^k)$$

and hence,

$$\log\left(\prod_{(v,i,p)\in W^k}(1-a_v)\right) \le \log(1-A^k),$$

which only holds if

$$\prod_{(v,i,p)\in W^k}(1-a_v) \le 1-A^k,$$

that is, the necessary condition derived from Remark 1. □

Theorem 2. *The separation problem associated with inequalities* (17) *can be solved in polynomial time.*

Proof. Start with $S=\emptyset$. For each $p\in P$, choose the section $i\in I^k$ with the smallest $\sum_{v\in V}\big(-\log(1-a_v)\big)x_{vip^k}$ and add (i,p) to S. □

5. Most and least available SFC configurations

In this section, we investigate the most and least available SFC configurations that can be obtained using a fixed given number of paths. From such study, we derive lower and upper bounds on the number of paths activated within an optimal solution. These bounds allow the fixing of a considerable amount of variables in our formulation. In the following let $\hat{v}_i$ (resp. $\check{v}_i$) denote the i-th most (resp. least) available node in V.

5.1. Most available SFC configuration

Claim 1. *The most available SFC configuration one can obtain for a given SFC using exactly q paths is*

$$\hat{C}_q = \{P_1, \ldots, P_q\}, \text{where } P_i = \{\hat{v}_i\}. \tag{18}$$

The availability it induces is given by $1 - \prod_{i=1}^{q} (1 - a_{\hat{v}_i})$.

Proof. The highest availability a path can have is the availability of the most available node. This is obtained when the required VNFs are all placed on such node. Since VNF-disjunction applies, the proposition holds and the highest availability one can obtain with exactly q paths is $1 - \prod_{i=1}^{q} (1 - a_{v_i})$. □

Given an SFC $k \in K$, let $p^k_{min} \leq \bar{p}$ denote the minimum number of paths required to ensure the SFC's required availability A^k. Providing a lower bound on p^k_{min} would allow the fixing of variables α. For instance, if $p^k_{min} \geq q$, then one can impose without loss of optimality that

$$\alpha^k_p = 1 \qquad \text{for } p = 1, \ldots, q.$$

Proposition 1. *Given an SFC k, let $\hat{q}$ be the minimum integer q for which $\hat{C}_q$ is feasible, that is, the minimum integer q for which*

$$1 - \prod_{i=1}^{q} (1 - a_{v_i}) \geq A^k \tag{19}$$

holds. Then, $p^k_{min} \geq \hat{q}$.

Proof. From the definition of $\hat{C}_{\hat{q}}$, there is no feasible configuration using strictly less than $\hat{q}$ paths. □

5.2. Least available SFC configuration

While the most available SFC configuration using exactly q paths can be easily found in polynomial time, finding the least available one is a much harder task.

Theorem 3. *The problem of finding the least available SFC configuration using exactly q paths, denoted as LA(q), is NP-Hard.*

Proof. We use a polynomial reduction from the NP-Hard 3-Partition Problem (3PP), which can be defined as follows. Given a set of $3q$ integers $S = \{s_1, \ldots, s_{3q}\}$, where $\sum_{i=1}^{3q} s_i = qB$, the 3PP consists of deciding whether or not set S can be partitioned into q triplets such that the sum of the elements in each triplet equals B. Given a set $S = \{s_1, \ldots, s_{3q}\}$ defining a 3PP instance, let us construct an instance of $LA(q)$ as follows. Let $V = \{v_1, \ldots, v_{3q}\}$, with $a(v_i) = e^{-s_i}$ for $i = 1, \ldots, 3q$. Moreover, let $\mathcal{F} = \{f_1, f_2, f_3\}$ denote the set of available VNFs. Finally, let k denote the single SFC to be deployed with $F^k = \mathcal{F}$. Our reduction does not require restrictions on the topology, capacities or latency, but for the sake of clarity, let $G = (V, A)$ be a complete network with unlimited arc and node capacities and irrelevant arc lengths, and assume SFC k has an irrelevant bandwidth and unlimited maximum latency.

Let $\check{C}_q = \{\check{P}_1, \ldots, \check{P}_q\}$ denote the least available SFC configuration using exactly q paths. Next we show that there exists a partition of set S solving the 3PP, if and only if sets $\check{P}_i \in \check{C}_q$ provide a solution to the 3PP.

Recall that the availability induced by an SFC configuration $C = \{P_1, \ldots, P_q\}$ is defined by

$$A(C) = 1 - \prod_{i=1}^{q} (\bar{a}(P_i)),$$

where $\bar{a}(P_i) = 1 - \prod_{v \in P_i} a_v$. From the definition of our constructed instance, we know that $\prod_{v \in V} a(v) = e^{-qB}$. Let us hence consider the following optimization problem:

$$\max \prod_{i=1}^{q} (\bar{a}(P_i)), \text{ subject to } \prod_{v \in V} a(v) = e^{-qB}. \tag{20}$$

Notice that $|\check{P}_i| = 3$, for $i = 1, \ldots, q$. Indeed, if some set $\check{P}_i \in \check{C}_q$ has less than 3 nodes, then the SFC configuration obtained by adding any node to P_i would be less available than $\check{C}_q$ since $0 < a(v) < 1$ for any $v \in V$. Moreover, from VNF-Disjunction constraints, sets $\check{P}_i$ must be disjoint and

hence

$$\prod_{i=1}^{q}(a(\check{P}_i)) = \prod_{v\in V} a(v) = e^{-qB}. \tag{21}$$

The optimization problem defined by (20) can thus be rewritten as

$$\max \prod_{i=1}^{q}(\bar{a}(P_i)), \text{ subject to } \prod_{i=1}^{q}(1-\bar{a}(P_i)) = e^{-qB}, \tag{22}$$

and the Lagrangian function associated with such problem is

$$L(\bar{a}(P_1),\ldots,\bar{a}(P_q),\lambda) = \prod_{i=1}^{q}(\bar{a}(P_i)) - \lambda\left(\prod_{i=1}^{q}(1-\bar{a}(P_i)) - e^{-qB}\right).$$

Using Karush-Kuhn-Tucker (KKT) conditions, it follows that an optimal solution to such problem should satisfy the following conditions:

$$\frac{\partial L}{\partial \bar{a}(P_i)} = \prod_{\substack{j=1:\\ j\neq i}}^{q}(\bar{a}(P_j)) - \lambda\left(\prod_{\substack{j=1:\\ j\neq i}}^{q}(1-\bar{a}(P_j))\right) = 0 \text{ for } i = 1,\ldots,q. \tag{23}$$

Therefore,

$$\frac{\prod_{\substack{i=1:\\ i\neq j}}^{q}(\bar{a}(P_i))}{\prod_{\substack{i=1:\\ i\neq j}}^{q}(1-\bar{a}(P_i))} = \frac{\prod_{\substack{i=1:\\ i\neq k}}^{q}(\bar{a}(P_i))}{\prod_{\substack{i=1:\\ i\neq k}}^{q}(1-\bar{a}(P_i))} = \lambda \text{ for } j,k = 1,\ldots,q.$$

By expanding the products, we obtain the following necessary optimality condition

$$\bar{a}(P_j)) = \bar{a}(P_k) \text{ for } j,k = 1,\ldots,q. \tag{24}$$

From (21) and (24), we get $a(P_i) = e^{-B}$ for $i = 1,\ldots,q$, and thus,

$$\ln a(P_i) = \ln \prod_{v\in P_i} a_v = \sum_{v\in P_i} \ln a_v = \ln\left(e^{-B}\right) = -B \text{ for } i = 1,\ldots,q.$$

Finally, recall from our instance construction that $a(v_i) = e^{-s_i}$ for $i =$

$1, \ldots, 3q$ and therefore,

$$\sum_{v \in P_i} \ln a_v = \sum_{v_j \in P_i} s_j = B \text{ for } i = 1, \ldots, q. \tag{25}$$

Therefore, the 3PP has a positive answer if and only if (25) holds for the least available SFC configuration $\check{C}_q = \{\check{P}_1, \ldots, \check{P}_q\}$. □

Given an SFC $k \in K$, let $p^k_{opt} \geq p^k_{min}$ denote the number of paths activated for SFC k within an optimal solution. Since many optimal solutions might exist, let p^k_{opt} be the minimal one. Finding an upper bound on p^k_{opt} would allow the elimination of variables and hence the reduction of the proposed formulation's dimension. Indeed, if $p^k_{opt} \leq q$, then one can impose without loss of optimality that

$$\begin{aligned} \alpha^k_p &= 0 && \text{for } p > q, \\ y^k_{vp} &= 0 && \text{for } v \in V, p > q, \\ x^k_{vip} &= 0 && \text{for } v \in V, i \in I^k, p > q, \\ f^k_{aip} &= 0 && \text{for } a \in A, i \in I^k, p > q. \end{aligned}$$

If the availability induced by least available SFC configuration $\check{C}_q = \{\check{P}_1, \ldots, \check{P}_q\}$ satisfies the requirements of a given SFC k, then $p^k_{opt} \leq q$. Therefore, finding the smallest q for which $\check{C}_q$ is feasible would then provide an upper bound on p^k_{opt}. Since finding $\check{C}_q$ is NP-Hard as stated by Theorem 3 we next propose another upper bound on p^k_{opt}.

Claim 2. *Let $k \in K$ be any SFC. Within any feasible SFC configuration $C^k = \{P^k_1, \ldots, P^k_q\}$, the availability of any path configuration P^k_i, $1 \leq i \leq q$, is at least*

$$\prod_{i=1}^{|I^k|} a_{\check{v}_i}.$$

Proof. Let P denote the least available path configuration possible for SFC k. Then, $|P| = |I^k|$, since otherwise a path configuration $P' = P \cup v'$, for any $v' \notin P$, would be less available than P. Therefore, it is clear that

path configuration P is composed by the $|I^k|$ least available nodes and its availability is given by

$$\prod_{i=1}^{|I^k|} a_{\check{v}_i}.$$

□

Theorem 4. *Let q^k be the minimum integer number for which $(1-a_P)^{q^k} \leq 1 - A^k$, where*

$$a_P = \prod_{i=1}^{|I^k|} a_{\check{v}_i}. \tag{26}$$

Then, $p^k_{opt} \leq q^k$ for any $k \in K$.

Proof. Let $C^k = \{P_1, \ldots, P_r\}$ denote a feasible SFC configuration for a given SFC $k \in K$, such that $r > q^k$. Then, the path configuration obtained from C^k by removing any of its path configurations remains feasible (*c.f.* Claim 2) and the cost of the new solution is at most the cost of the previous one. Therefore, $p^k_{opt} \leq q^k$. □

6. Notes for future work

- Placements dominance relationships in order to reinforce (16);
  - Idea: Placing an additional VNF on an activated path can only make the availability drop. Hence given a placement $S \subseteq V \times P$ such that $A(S) < A^k$, every placement $S' = S \cup T$ does not satisfy the availability constraint, where $T \subseteq D(S) = \{(v,p) \in V \times P : \exists (u,p) \in S\}$. Therefore, the following constraints can be seen as a lifted version of (16) and remain valid.

    $$\sum_{(v,p)\in S} (1-y^k_{vp}) + \sum_{(v,p)\notin S \cup D(S)} y^k_{vp} \geq 1 \qquad \forall k \in K, S \subseteq V \times P : A(S) < A^k.$$

    Such constraints can be interpreted by thinking that a non-feasible placement can only become feasible if either a non-active path

becomes active, or a node processing a VNF is removed from an active path.

- Symmetry handling: For a given demand, the permutation of its paths results in an equivalent solution;
  - Idea: A lexicographic order on the activation of paths can be imposed and hence Orbitopal Fixing (see Kaibel et al. (2011)) or symmetry-breaking constraints can be applied.
- Strong Node Capacity Inequalities:

$$\sum_{k \in K} \sum_{p=1}^{|V|} \sum_{i \in I^k} b^k r_{f_i^k} x_{vip}^k \leq C_v \qquad \forall v \in V. \tag{27}$$

References

Kaibel, V., Peinhardt, M., & Pfetsch, M. E. (2011). Orbitopal fixing. *Discrete Optimization*, *8*, 595–610.

Piecewise-linear approximations

Rafael Colares[a,*]

[a] *Orange Labs, 46 Avenue de la République, 92320 Châtillon, France*

Abstract

This is document provides some ideas on how to approximate the feasible region of the Resilient VNF Placement and SFC Routing formulation through piecewise linear approximations.

Keywords: Virtual Network Functions, Service Function Chain, resilience, combinatorial optimization, MIP reformulations

1. Approximations

Consider the set $P1$ induced by the following system of inequalities:

$$
\begin{aligned}
(P1) \qquad & \prod_{p \in P} (1 - a_p) \leq 1 - A, && && (1)\\
& a_p \leq \alpha_p && \forall p \in P, && (2)\\
& a_p \leq \prod_{v \in V} (a_v)^{y_{vp}} && \forall p \in P, && (3)\\
& a_p \geq 0 && \forall p \in P, && (4)\\
& y_{vp} \in \{0, 1\} && \forall v \in V, p \in P, && (5)
\end{aligned}
$$

where y_{vp} is a binary variable stating whether or not a VNF is hosted on node $v \in V$ along path $p \in P$, α_p is a binary variable stating whether or not path $p \in P$ is active, and a_p is a real variable denoting the availability of path $p \in P$. Moreover, A is the required SFC availability, and a_v is the availability of node $v \in V$.

*Corresponding author

Email address: rafael.colares@gmail.com (Rafael Colares)

$P1$ models the satisfiability of the availability requirements of an SFC. Next we show how to approximate such set using only linear inequalities.

Since $x \leq y$ holds iff $\log(x) \leq \log(y)$, $P2$ is equivalent to $P1$.

$$
\begin{aligned}
(P2) \qquad & \prod_{p \in P} (1 - a_p) \leq 1 - A, && && (6)\\
& a_p \leq \alpha_p && \forall p \in P, && (7)\\
& \log(a_p) \leq \sum_{v \in V} \big(\log(a_v) y_{vp} \big) && \forall p \in P, && (8)\\
& a_p \geq 0 && \forall p \in P, && (9)\\
& y_{vp} \in \{0, 1\} && \forall v \in V, p \in P, && (10)
\end{aligned}
$$

Let $g(x)$ be a linear function such that $g(x) \geq \log(x)$ for any $x \in \mathbb{R}$. Then, $P3 \subseteq P2$.

$$
\begin{aligned}
(P3) \qquad & \prod_{p \in P} (1 - a_p) \leq 1 - A, && && (11)\\
& a_p \leq \alpha_p && \forall p \in P, && (12)\\
& g(a_p) \leq \sum_{v \in V} \big(\log(a_v) y_{vp} \big) && \forall p \in P, && (13)\\
& a_p \geq 0 && \forall p \in P, && (14)\\
& y_{vp} \in \{0, 1\} && \forall v \in V, p \in P, && (15)
\end{aligned}
$$

Notice that the system of inequalities defining $P3$ is linear with the exception of inequalities (22). The same procedure used for obtaining $P3$ can now be applied to linearize inequalities (22) and obtain $P4$ such that $P4 \subseteq P3$.

$$(P4) \qquad \sum_{p \in P} g(1 - a_p) \leq \log(1 - A), \tag{16}$$

$$a_p \leq \alpha_p \qquad \forall p \in P, \tag{17}$$

$$g(a_p) \leq \sum_{v \in V} \big(\log(a_v) y_{vp} \big) \qquad \forall p \in P, \tag{18}$$

$$a_p \geq 0 \qquad \forall p \in P, \tag{19}$$

$$y_{vp} \in \{0, 1\} \qquad \forall v \in V, p \in P, \tag{20}$$

The question is how to define the linear function $g(x)$. In Billionnet (2011), $g_u(x)$ is constructed by selecting a vector $u \in \mathbb{R}^K$ such that $0 < u_1 < u_2 < \cdots < u_K = 1$, and $g_u(x)$ is the piecewise-linear function of K segments tangent to the curve $\log(x)$ at the points $u_1, u_2, \ldots, u_K$. Figure 1 illustrates the function $g_u(x)$ in comparison with $\log(x)$, with $u_1 = 0.1$, $u_2 = 0.3$ and $u_3 = 1$.

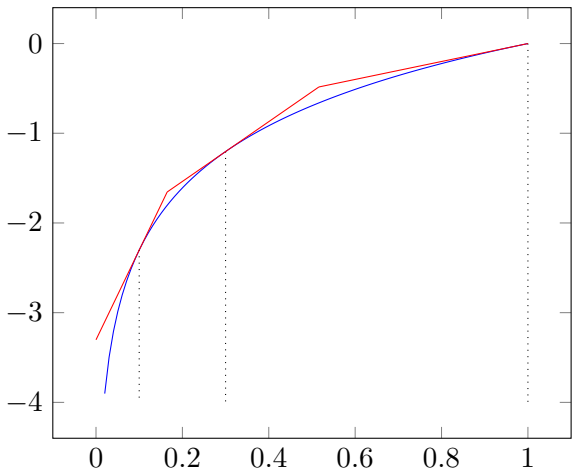


Figure 1: Illustration of $g(x)$

The points where the slope of the function changes are known as breakpoints. The piecewise-linear function g_u constructed from vector $u \in \mathbb{R}^K$

has $K-1$ breakpoints. These breakpoints are defined as follows.

$$b_i = \frac{\log(u_{i+1}) - \log(u_i)}{1/u_i - 1/u_{i+1}} \quad \text{for } i = 1, \ldots, K-1. \tag{21}$$

2. The choice of vector u

In Billionnet (2011), vector u is defined by choosing a "sufficiently" small number for u_1 and then applying the rule

$$u_i = u_1^{(K-i)/(K-1)} \quad \text{for } i = 1, \ldots, K.$$

By doing so, the decrease in the slopes of any two consecutive segments of the piecewise-linear function g_u is constant. More precisely, the ratio between two consecutive slopes is exactly $u_1^{1/(K-1)}$. Obviously, the bigger the K, the better is the approximation.

The nature of our optimization problem requires remarkably good approximations since otherwise many unnecessary backup paths would be required to secure an SFC. To do this, a sufficiently large K would be needed, which can significantly slow down the solving process. In our case however, we do not need to approximate $\log(a_p)$ along all values between 0 and 1. Indeed, if a path is active, we already know a lower bound and an upper bound for its availability (see Document *PostDoc Orange V*). Let lb (resp. ub) denote such lower (resp. upper) bound. It follows that either $a_p = 0$ (in the case the path is not active) or $lb \leq a_p \leq ub$. Moreover, if the path is not active, then $\alpha_p = 0$ and hence constraint (23) forces a_p to be 0. Therefore we only need to well approximate the function log in the interval $[lb, ub]$. For this reason, we let

$$u_i = ub \left(\frac{lb}{ub}\right)^{(K-i)/(K-1)} \quad \text{for } i = 1, \ldots, K.$$

By doing so, the property of regular slope decreasing is preserved within our interval of interest, that is, from lb to ub. More precisely, the ratio between two consecutive slopes is exactly $(lb/ub)^{1/(K-1)}$.

The same reasoning can be applied for approximating $\log(1-a_p)$ in this case our interval of interest is between $(1-ub)$ and $(1-lb)$. Therefore we can set

$$t_i = (1-lb)\left(\frac{(1-ub)}{(1-lb)}\right)^{(K-i)/(K-1)} \qquad \text{for } i = 1,\ldots,K.$$

in order to define function g_t that approximates $\log(1-a_p)$. Our resulting approximated linear system is described here below.

$$(P5) \qquad \sum_{p\in P} g_t(1-a_p) \leq \log(1-A), \tag{22}$$

$$a_p \leq \alpha_p \qquad \forall p \in P, \tag{23}$$

$$g_u(a_p) \leq \sum_{v\in V} \big(\log(a_v) y_{vp}\big) \qquad \forall p \in P, \tag{24}$$

$$a_p \geq 0 \qquad \forall p \in P, \tag{25}$$

$$y_{vp} \in \{0,1\} \qquad \forall v \in V, p \in P. \tag{26}$$

3. To do list

- For the moment I have implemented the piecewise-linear approximation myself. However, CPLEX offers its own implementation of piecewise linear functions. It might be interesting to use CPLEX's implementation. ***Done*** ✓

- The approximation described in this document is done by approximating the function $\log(x)$ with a piecewise-linear function $g(x)$ such that $g(x) \geq \log(x)$. This guarantees that the feasible region of the obtained linear program is a subset of the original set of feasible solutions. An approach using a piecewise-linear function $f(x)$ such that $f(x) \leq \log(x)$ can be applied for obtaining a relaxation of the original problem. Such relaxation can then be reinforced with our availability cuts. ***Done*** ✓

- Computational experiments. Some experiments have been performed and results seem to be good but most of the instances were too hard to solve (check Section 4). In comparison with the availability non-aware case that can be solved for up to 40 demands (Ahlam's work), in the availability aware case, each demand requires 2, 3 or even 4 paths. This has a huge impact on the network capacity and other combinatorial aspects... Generate another set of instances with less demands and more representative latencies and capacities.

- Apply the piecewise-linear approximation principle to the formulation submitted to INOC. ***Done*** ✓

4. Computational results

Next is the summary of the first experimental results obtained. (B) stands for the basic formulation and (R) stands for the piecewise-relaxed formulation reinforced with the lazy constraints. On (R) only 2 breakpoints were used.

Table 1: Computational results comparison

Instance		optimal		time (s)		gap (%)		# lazy	
$\|K\|$	Network	(B)	(R)	(B)	(R)	(B)	(R)	(B)	(R)
10	Abilene	5/5	5/5	735	368	0	0	1003	243
20	Abilene	0/5	0/5	7200	7200	51.1	37.5	9552	2399
30	Abilene	0/5	0/5	-	-	-	-	-	-
40	Abilene	0/5	0/5	-	-	-	-	-	-

5. Applying piecewise-linear approximations to INOC's approach

Recall that within this approach we have a set of binary variables x such that $x_{vik} = 1$ if the i-th VNF of SFC k can be executed on node v. For a

given SFC k, let S_i denote the set of nodes that can execute its i-th VNF. An illustration of a solution is depicted below.

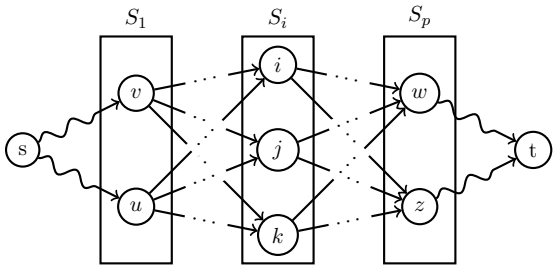


Figure 2: Illustration of a VNF assignment and the possible SFC routes induced by it.

Let a_i be a variable measuring the availability of the i-th section of a given SFC k. Consider the following non-linear formulation for ensuring the SFC's required availability.

$$(Q1) \qquad \prod_{i \in I^k} a_i \geq A^k, \tag{27}$$

$$a_i \leq 1 - \prod_{v \in V} (1 - a_v)^{x_{vik}} \qquad \forall p \in P, \tag{28}$$

$$a_i \geq 0 \qquad \forall i \in I^k, \tag{29}$$

$$x_{vik} \in \{0, 1\} \qquad \forall v \in V, i \in I^k. \tag{30}$$

By applying the logarithmic function on both sides of inequalities (44) and (45) we obtain an equivalent set Q2 as follows.

$$(Q2) \qquad \log\left(\prod_{i \in I^k} a_i\right) \geq \log(A^k), \tag{31}$$

$$\log(1 - a_i) \geq \log\left(\prod_{v \in V} (1 - a_v)^{x_{vik}}\right) \qquad \forall p \in P, \tag{32}$$

$$a_i \geq 0 \qquad \forall i \in I^k, \tag{33}$$

$$x_{vik} \in \{0, 1\} \qquad \forall v \in V, i \in I^k. \tag{34}$$

Since $\log(ab) = \log(a) + \log(b)$ and $\log(a^b) = b \log(a)$, Q2 can be rewritten

as follows.

$$(Q2) \qquad \sum_{i \in I^k} \log(a_i) \geq \log(A^k), \tag{35}$$

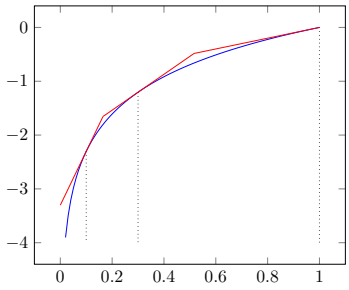
$$\log(1 - a_i) \geq \sum_{v \in V} (\log(1 - a_v) x_{vik}) \qquad \forall p \in P, \tag{36}$$

$$a_i \geq 0 \qquad \forall i \in I^k, \tag{37}$$

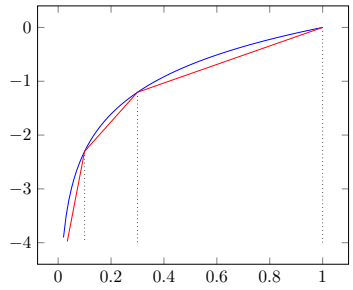
$$x_{vik} \in \{0, 1\} \qquad \forall v \in V, i \in I^k. \tag{38}$$

Let f and g be two piecewise-linear functions such that

$$f(x) \leq \log(x) \leq g(x). \tag{39}$$



(a) Illustration of $g(x)$



(b) Illustration of $f(x)$

Consider next the two following linear formulations.

$$(Q^-) \qquad \sum_{i \in I^k} f(a_i) \geq \log(A^k), \tag{40}$$

$$f(1 - a_i) \geq \sum_{v \in V} (\log(1 - a_v) x_{vik}) \qquad \forall p \in P, \tag{41}$$

$$a_i \geq 0 \qquad \forall i \in I^k, \tag{42}$$

$$x_{vik} \in \{0, 1\} \qquad \forall v \in V, i \in I^k. \tag{43}$$

$$(Q^+) \qquad \sum_{i \in I^k} g\left(a_i\right) \geq \log(A^k), \tag{44}$$

$$g(1-a_i) \geq \sum_{v \in V} \left(\log(1-a_v) x_{vik}\right) \qquad \forall i \in I^k, \tag{45}$$

$$a_i \geq 0 \qquad \forall i \in I^k, \tag{46}$$

$$x_{vik} \in \{0,1\} \qquad \forall v \in V, i \in I^k. \tag{47}$$

From (39), it follows that

$$(Q^-) \subseteq (Q2) \subseteq (Q^+) \tag{48}$$

Question: Should we begin to prepare an extended version of INOC paper with this piecewise-linear approx for a submission on Networks?

References

Billionnet, A. (2011). Solving the probabilistic reserve selection problem. *Ecological modelling*, *222*, 546–554.

Piecewise-linear approximations

Rafael Colares[a,*]

[a] *Orange Labs, 46 Avenue de la République, 92320 Châtillon, France*

Abstract

This is document provides some ideas on why our piecewise linear approximation is facing numerical issues.

Keywords: Virtual Network Functions, Service Function Chain, resilience, combinatorial optimization, piecewise linear approximations

1. Some notation and remarks

- This document focuses on the case that a single SFC needs to be deployed. (Everything can be trivially extended to the case of multiple SFCs.)
- Required availability : A
- Availability of a node v : $a_v \in [0.9, 0.99]$
- Nodes are sorted by their availability : $a_{v_1} \leq a_{v_2} \leq \cdots \leq a_{v_n}$
- I is the set of sections
- $g(x)$ is a piecewise linear function such that $g(x) \geq \log(x)\ \forall x \in [0, 1]$

*Corresponding author
Email address: rafael.colares@gmail.com (Rafael Colares)

2. Approximation Formulation

$$\sum_{i \in I} g(\alpha^i) \geq \log(A), \tag{1}$$

$$g(1-\alpha^i) = \sum_{v \in V} (\log(1-a_v) x_{vik} \qquad \forall i \in I \tag{2}$$

where x_{vik} is a binary variable stating whether or not the i-th VNF of SFC k is hosted on node $v \in V$. Moreover, α^i is a continuous variable in $[0,1]$ that refers to the availability of section i.

Notice that a key point is to well approximate function g to the function log. For this, having good bounds on the values that α^i can take is crucial.

3. Bounds on availabilities

In our previous approach, we wanted to find bounds on the availability that a ***single path*** can induce. Recall that the following bounds were found:

$$0.6 \equiv a_{v_1}^{|I|} \leq \text{Path availability} \leq a_{v_n} \equiv 0.99 \tag{3}$$

In the INOC approach, we need to find bounds on the availability that a ***section*** can induce. These are given by

$$0.9 \equiv a_{v_1} \leq \text{Section availability} \leq 1-(1-a_{v_n})^n \equiv 0.999\ldots9 \equiv 1 \tag{4}$$

4. Numerical issues

- When building the approximation function $g(1-\alpha^i)$, $\log(0)$ is undefined! $\rightarrow$ replace 0 by ϵ, the smallest absolute number

In Billionnet (2011), $g_u(x)$ is a piecewise linear function constructed by selecting a vector $u \in \mathbb{R}^K$ such that $0 < u_1 < u_2 < \cdots < u_K = 1$, and $g_u(x)$ is the piecewise-linear function of K segments tangent to the curve $\log(x)$ at the points $u_1, u_2, \ldots, u_K$. See figure below.

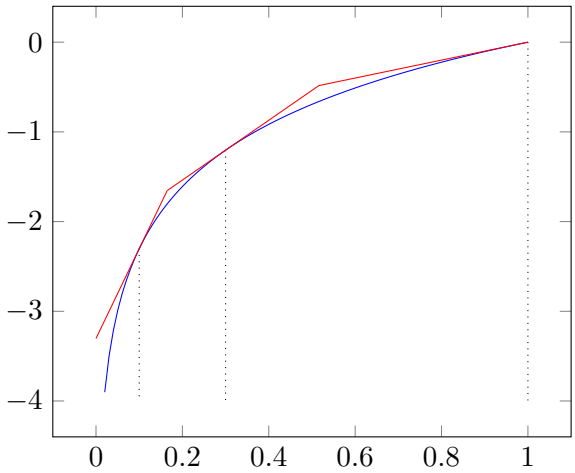


Figure 1: Illustration of $g(x)$

The points where the slope of the function changes are known as breakpoints. The piecewise-linear function g_u constructed from vector $u \in \mathbb{R}^K$ has $K-1$ breakpoints. These breakpoints are defined as follows.

$$b_i = \frac{\log(u_{i+1}) - \log(u_i)}{1/u_i - 1/u_{i+1}} \quad \text{for } i = 1, \ldots, K-1. \tag{5}$$

In Billionnet (2011), vector u is defined by choosing a "sufficiently" small number for u_1 and then applying the rule

$$u_i = u_1^{(K-i)/(K-1)} \quad \text{for } i = 1, \ldots, K.$$

By doing so, the decrease in the slopes of any two consecutive segments of the piecewise-linear function g_u is constant. More precisely, the ratio between two consecutive slopes is exactly $u_1^{1/(K-1)}$. Obviously, the bigger the K, the better is the approximation.

The nature of our optimization problem requires remarkably good approximations. To do this, a sufficiently large K would be needed, which can significantly slow down the solving process. In our case however, we do not need to approximate the function $\log(x)$ along all values between 0 and 1. Indeed, we may take profit of the bounds identified in Section 3.

- Since the bounds are too close (and because of the nature of the slope behaviour of function log), breakpoints are too close for the computer to make any difference between them → CPLEX crashes. → space them linearly ***or*** consider that $\alpha^i \leq 1 - 1e - 10$

5. Final remarks

- including the approximation has a great impact on the upper bound convergence (CPLEX struggles to find good feasible solutions)
- There are still many cuts that are being included (a sign that the approximation is not good enough)
- We are approximating the function log in a zone where the values change very quickly:
  - $\log(0.000001) = -6$
  - $\log(0.000002) = -5.69$

6. Matheuristic

Consider the following algorithm to be called whenever a fractional feasible solution $(\bar{x}, \bar{y})$ is available.

Algorithm 1: Phase 1: Construct a partial solution

foreach $v \in V, f \in F$ **do**
  set y_v^f to 1 with a probability of $\bar{y}_v^f$;
  if $y_v^f = 0$ **then**
    set x_{vik} to 0 $\forall i, k : f(i,k) = f$;
  else
    set x_{vik} to 1 with a probability of $\bar{x}_{vik}$ (if node capacity allows it);
call Phase 2;

Phase 1 builds a partial feasible integer solution by selecting a set of VNFs to be installed on each node. The decision of installing or not a

Algorithm 2: Phase 2: Make the partial solution feasible

```
foreach k ∈ K do
    while Availability of service k < A^k do
        let i be the least available section of k;
        sort nodes in V by c_v^{f(i,k)}(1 − y_v^{f(i,k)}) ;
        let v be the first node that can host the VNF (w.r.t. the
          remaining node capacity);
        set x_vik to 1;
```

VNF on a given node is influenced by the fractional value of the associated variable. If we decide to install the VNF on the node, then we choose a subset of SFCs to be assigned to such node.

When Phase 1 is finished we have an integer solution that is possibly unfeasible with respect to the SFC's availability. Phase 2 is hence responsible for reinforcing the availability of the unfeasible SFCs.

6.1. When do we run the heuristic procedure?

The heuristic procedure described above can be called whenever a fractional feasible solution is available (*i.e.*, after the LP resolution of each node of the branch-and-cut enumeration tree). Calling such procedure on every node, might be cumbersome and counter-productive. For this reason we propose the following policy for deciding whether or not the procedure should be executed.

At any given node, let UB denote the best known upper bound value and obj denote the value of the objective function obtained after the LP resolution of such node. We propose that the heuristic procedure should be called on every node of the branch-and-cut tree with a probability of

$$\frac{UB - obj}{UB}.$$

Such policy encourages the search for an integer feasible solution when the quality of the best known solution is relatively bad.

References

Billionnet, A. (2011). Solving the probabilistic reserve selection problem. *Ecological modelling*, *222*, 546–554.

Matheuristic

Rafael Colares[a,*]

[a] *Orange Innovation, 46 Avenue de la République, 92320 Châtillon, France*

Abstract

This is document provides some ideas on how to generate a feasible VNF placement given a fractional LP solution.

Keywords: Virtual Network Functions, Service Function Chain, resilience, combinatorial optimization, matheuristics

1. Linear Program

Let us recall here the description of our MILP (approximated) formulation, where $g(x)$ is a piecewise-linear function that approximates the logarithmic function from above (*i.e.*, $g(x) \geq \log(x)$ for any $x \in]0, 1]$). The objective function (1) minimizes the VNF placement cost. Inequalities (2) ensure that at least one VNF must be installed on each section of each SFC. The VNF placement constraints (3) impose that a VNF can only be assigned to an SFC if it is already placed. The capacity constraints (4) guarantee that the capacity of each node is not exceeded. Availability constraints (5) and (6) force the SFCs' Service Level Agreement to be (approximately) respected. Finally, constraints (7)-(9) settle the domains of variables.

*Corresponding author
Email address: rafael.colares@gmail.com (Rafael Colares)

$$\min \sum_{v \in V} \sum_{f \in \mathcal{F}} c_v^f y_v^f \tag{1}$$

subject to

$$\sum_{v \in V} x_{vik} \geq 1 \qquad \forall k \in K, i \in I^k, \tag{2}$$

$$x_{vik} \leq y_v^{f(i,k)} \qquad \forall k \in K, i \in I^k, v \in V, \tag{3}$$

$$\sum_{k \in K} \sum_{i \in I^k} b_k r^{f(i,k)} x_{vik} \leq C_v \qquad \forall v \in V, \tag{4}$$

$$\sum_{i \in I^k} g\left(a_i^k\right) \geq \log(A^k), \tag{5}$$

$$g(1 - a_i^k) = \sum_{v \in V} (\log(1 - a_v) x_{vik}) \qquad \forall i \in I^k, \tag{6}$$

$$0 \leq a_i^k \leq 1 \qquad \forall k \in K, i \in I^k, \tag{7}$$

$$x_{vik} \in \{0, 1\} \qquad \forall v \in V, k \in K, i \in I^k, \tag{8}$$

$$y_v^f \in \{0, 1\} \qquad \forall v \in V, f \in \mathcal{F}. \tag{9}$$

2. Matheuristic

Consider the following algorithm to be called whenever a fractional feasible solution $(\bar{x}, \bar{y})$ is available.

Algorithm 1: Phase 1: Construct a partial solution

foreach $v \in V, f \in F$ **do**
  set y_v^f to 1 with a probability of $\bar{y}_v^f$;
  if $y_v^f = 0$ **then**
    set x_{vik} to 0 $\forall i, k : f(i, k) = f$;
  else
    set x_{vik} to 1 with a probability of $\bar{x}_{vik}$ (if node capacity allows it);
call Phase 2;

Algorithm 2: Phase 2: Make the partial solution feasible

```
foreach k ∈ K do
    while Availability of service k < A^k do
        let i be the least available section of k;
        sort nodes in V by c_v^{f(i,k)}(1 − y_v^{f(i,k)}) ;
        let v be the first node that can host the VNF (w.r.t. the
          remaining node capacity);
        set x_vik to 1;
```

Phase 1 builds a partial feasible integer solution by selecting a set of VNFs to be installed on each node. The decision of installing or not a VNF on a given node is influenced by the fractional value of the associated variable. If we decide to install the VNF on the node, then we choose a subset of SFCs to be assigned to such node.

When Phase 1 is finished we have an integer solution that is possibly unfeasible with respect to the SFC's availability. Phase 2 is hence responsible for reinforcing the availability of the unfeasible SFCs.

2.1. When do we run the heuristic procedure?

The heuristic procedure described above can be called whenever a fractional feasible solution is available (*i.e.*, after the LP resolution of each node of the branch-and-cut enumeration tree). Calling such procedure on every node, might be cumbersome and counter-productive. For this reason we propose the following policy for deciding whether or not the procedure should be executed.

At any given node, let UB denote the best known upper bound value and obj denote the value of the objective function obtained after the LP resolution of such node. We propose that the heuristic procedure should be called on every node of the branch-and-cut tree with a probability of $\frac{UB-obj}{UB}$. Such policy encourages the search for an integer feasible solution when the quality of the best known solution is relatively bad.

References