A multidimensional colored packing approach for network slicing with dedicated protection

Hicham Lesfari, Frédéric Giroire, Giuseppe Di Lena, Chidung Lac

A multidimensional colored packing approach for network slicing with dedicated protection

Hicham Lesfari*, Frédéric Giroire*, Giuseppe Di Lena*†, Chidung Lac†
*Université Côte d'Azur, CNRS, Inria, France
†Orange Labs, France

***Abstract*—Network Function Virtualization (NFV) enables the virtualization of core-business network functions on top of a NFV infrastructure. NFV has gained an increasing attention in the telecommunication field these last few years. Virtual network functions (VNFs) can be represented by a set of virtual network function components (VNFCs). These VNFCs are typically designed with a redundancy scheme and need to be deployed against failures of, e.g., compute servers. However, such deployment must respect a particular resiliency mechanism for protection purposes. Therefore, choosing an efficient mapping of VNFCs to the compute servers is a challenging problem in the optimization of the software-defined, virtualization-based next generation of networks. In this paper, we model the problem of reliable VNFCs placement under anti-affinity constraints using several optimization techniques. A novel approach based on an extension of bin packing is proposed. We perform a comprehensive evaluation in terms of performance under real-world ISP networks along with synthetic traces. We show that our methods can calculate rapidly efficient solutions for large instances.**

I. Introduction

A technology which has gained an important momentum in the networking arena is the Network Function Virtualization (NFV) paradigm. The main idea is to dissociate the dependency on dedicated hardware, such as proprietary software appliances, by allowing network functions such as load balancing, firewall, content filtering and deep packet inspection, to be virtualized and executed on generic servers. As such, Virtual Network Functions (VNFs) can be instantiated and scaled on demand without the need to install new equipment, increasing flexibility to accompany user demands [1]. Another fundamental and arising technology is Software-Defined Networking (SDN) that simplifies network monitoring and management. It enables to decouple the control plane from the data plane, and provides global vision and control of the network [2]. The synergy of SDN and NFV gives rise to highly dynamic, programmable and flexible networks where both network infrastructures and resources are shared between the various network services.

Network slices are usually provided in the form of an end-to-end logical network provisioned with a set of isolated virtual resources on a shared physical infrastructure. Specifically, a network slice is composed of a set of VNFs, and each VNF is made up of several Virtual Network Function Components (VNFCs). Some of these components are anticipated to host services which usually operate at high velocity. Therefore, a short-lived network outage can disrupt significantly these slices. Reliable placement of these VNFCs is a vital part of the virtual resources allocation and remains one of the key challenges of network slicing. A viable solution for the placement of VNFCs with respect of network slicing consists in having a dedicated backup instance for each component such that some instances handle the service traffic, while others are used as a protection mechanism against failures of compute servers. Indeed, for resiliency purposes, some VNFCs have to be run in parallel such that different replicas of a same VNFC are concurrently deployed as a dedicated protection mechanism. Such resiliency constraints belong to the class of anti-affinity rules and are often related to the hosts that are anticipated to provision them. In particular, the network infrastructure across Internet Service Providers consists of compute servers made up of different capacity constraints zones as part of the horizontal scaling. An example of such zones are the NUMA zones [3] which are used as a protection mechanism for the host and are practically being deployed as such on physical motherboards.

Our work consists in finding an efficient placement strategy to map the network slices with the infrastructure, while ensuring a dedicated protection. Such protection usually follows some design models according to the pre-provisioning, along with deployment models according to the type of traffic regulating the slice demands. Such slices are, for instance, handled in the OpenStack cloud computing infrastructure in an online fashion without any sorting mechanism. However, giving priority to resolving capacity constraints and ignoring the conflicting aspects of the problem addressed may lead to an overestimation of the necessary number of compute servers.

In this paper, we propose different algorithms for the reliable allocation of resources for network slicing, and compare them with the placement algorithm used by the main computing engine behind the OpenStack infrastructure. Moreover, we propose a coloring based algorithm which extends over multiple dimensions, while satisfying the dedicated protection mechanism for the components. We also take into consideration the distribution of free resources across the compute servers after the provisioning while devising our algorithms. In particular, we consider the effect of the sorting criteria of the network slice over the total performance.

The rest of this paper is organized as follows. We first

This work has been supported by the French government through the UCA JEDI (ANR-15-IDEX-01) and EUR DS4H (ANR-17-EURE-004) Investments in the Future projects.

Symbol	Description
$\mathcal{S}$	Set of compute servers
Z	Set of zones for each compute
Γ	Total number of VNFC instances
$\mathcal{C}$	Set of all VNFCs
c	A component in $\mathcal{C}$
T_c	Number of available replicas of c
$\mathcal{C}_c^k$	The k-th replica of c
f_c^k	Network vector for $\mathcal{C}_c^k$
$R_c\|\|A_c$	Design resiliency scheme for c

TABLE I: Glossary of notations

discuss the relevant research literature in Section II. We then formulate in Section III the problem statement and describe different algorithms. In Section IV, we evaluate our approach over synthetic and real-world network slices and discuss the obtained results. Conclusions are drawn in Section V, together with open questions for future work.

II. Related Work

A large number of works has been devoted to the deployment and management of network services. We refer the reader to the surveys in [4] and [5].

Failure protection. The problem of providing network protection against failures has been studied for different use cases such as bandwidth-optimal recovery placement [6] and reliable service function chaining [7]. In [8], the problem of distributing VNF replicas between both primary and backup paths while maximizing the availability has been considered, based on a heuristic algorithm. The setting where slicing placement requires multiple network functions, which require a different amount of each resource to process a unit of flow has been studied in [9]. Different from previous studies on failure recovery, we present a resilience mechanism for VNFCs placement to protect against compute server failures where the network slicing is deployed over an infrastructure of NUMA zones.

Vector Bin packing. The classical bin packing (BP) problem has been extensively studied. While for its one dimensional setting, [10] showed an asymptotic polynomial time approximation scheme (APTAS) which was later improved by [11], it turned out that there is no APTAS for square packing [12]. The approximability of the multidimensional setting of BP has been studied in [13] and [14], where improved approximation algorithms were designed. Some variants of BP [15], [16], [17] have been studied ever since, where some pair of items are in conflict, i.e. not allowed to be packed together in the same bin. However, a setting under conflicts over an arbitrary number of dimensions has not been considered in previous works.

To the best of our knowledge, a general multidimensional setting under conflict constraints for reliable network slicing has not been explored. We combine both the multidimensional aspect with the capacity and resiliency constraints, and provide fast and efficient combinatorial algorithms.

III. Problem statement

A. Problem definition

In order to deploy a network service, Internet Service Providers (ISPs), Content Delivery Providers (CDPs) and other Telco operators usually buy or rent a technical solution from a vendor. Such solution is composed of a set of VNFs, each VNF being itself a set of VNFCs. Each VNFC requires for its operations a certain amount of virtual CPU and memory. To run the solution, the customer uses an infrastructure composed of compute servers, networking resources, and storage assets. Compute servers, used for illustration in this study, are made up of several zones, each of these zones being characterized by a maximum amount of CPU and memory.

The deployment of VNFs/VNFCs must respect a set of rules. First, since the capacity of a compute server is limited, only a subset of VNFCs can be hosted in any compute. Moreover, each single VNFC cannot be split across different zones. Last, for resiliency purposes, each VNFC is usually subject to an anti-affinity rule which is defined by a design redundancy model. Let us suppose that a certain VNFC has some design scheme denoted by $R||A$. Such a scheme implies that the vendor has decided that R replicas (replica and instance are used interchangeably in this paper) of the VNFC are needed to handle the traffic, but A additional ones have to be deployed for protection purposes. Alternatively, every R instances must be protected by A backups such that if a single compute server fails, at least R instances of the VNFC always run. Such redundancy protection is therefore designed against single failures of the compute servers.

The main aim is to find the minimal number of compute servers needed to host the VNFs of the network service, by piling up the VNFC instances in these computes, while respecting the anti-affinity rule for the redundant instances. We refer to this problem by VNFC-Protec.

We use the notations in Table I in the rest of this paper, to make it self-contained.

Network vector. We represent a network service by a set $\mathcal{C}$ of VNFCs, such that an instance $\mathcal{C}_c^k$ from $\mathcal{C}$ denotes the k-th replica of the VNFC c. Moreover, for each component c, let $\alpha(c) = \{\mathcal{C}_c^k, 1 \leq k \leq T_c\}$ denote the set of its available replicas. We associate with c a network vector f_c^k which acts as a summary indicator for the network characteristics required by the VNFC. Since all replicas of a same VNFC are similar, it holds that, for each component c fixed:

$$\forall k, k' \in \alpha(c) : f_c^k = f_c^{k'}.$$

Therefore, we can drop the subscript k for ease of notation and assimilate $f_c = f_c^k$ for each copy $k \in \alpha(c)$ of c.

For the VNFC-Protec problem, we consider a three-dimensional network vector $f_c = (f_c^{cpu}, f_c^{ram}, f_c^{res}) \in \mathbb{R}^3$ such that, for each VNFC c, $f_c^{cpu} \in (0, 1]$ is its normalized amount of needed CPU, $f_c^{ram} \in (0, 1]$ is its normalized amount of needed memory and $f_c^{res} \in \mathbb{N}$ is its deployment resiliency factor as described hereafter.

Deployment resiliency factor. Let us consider a VNFC c with a design scheme $N_c||R_c = 3||1$ decided by the vendor. Thus, every three instances of c needed to manage the traffic, must be protected by one additional instance. Moreover, let us assume that there are a total of $T_c = 7$ replicas of the VNFC. It implies that the compute servers in the infrastructure must

host seven instances of c such that: 3 instances are protected by one backup and 2 instances are protected by one backup. Therefore, the deployment scheme induced by $(N_c||R_c, T_c)$ would be: $5||\mathbf{2}$. The deployment resiliency factor associated with the aforementioned VNFC c is defined as $f_c^{res} = 2$ from the deployment scheme.

More generally, the *deployment resiliency factor* for a component c under the design scheme $N_c||R_c$ is defined as:

$$f_c^{res} = \left\lceil \frac{T_c}{N_c + R_c} \right\rceil$$

The factor corresponds therefore to the maximum number of replicas for the VNFC c allocable to a single compute server. It captures the anti-affinity rule induced by the traffic for a given VNFC. Note that the *deployment scheme* for a VNFC c can be thus expressed as $T_c - f_c^{res}||f_c^{res}$.

B. Exact ILP

We describe an exact Integer Linear Program which computes the optimal solution by minimizing the number of hosts needed. The main variables are the boolean variables x_s and z_{kcsz} defined as follows:

- $x_s = 1$ if there is a VNFC instance assigned to the compute server $s \in \mathcal{S}$ and 0 otherwise.
- $z_{kcsz} = 1$ if the copy k of the VNFC $c \in \mathcal{C}$ is assigned to the host $s \in \mathcal{S}$ in the zone $z \in Z$ and 0 otherwise.

The objective is to minimize the number of compute servers, i.e.,

$$\min \sum_{s \in \mathcal{S}} x_s \tag{1}$$

The maximum capacity for each zone is then:

$$\sum_{c \in \mathcal{C}, k \in \alpha(c)} z_{kcsz} \cdot f_c^{cpu} \leq z_C \quad \forall s \in \mathcal{S}, z \in Z \tag{2}$$

$$\sum_{c \in \mathcal{C}, k \in \alpha(c)} z_{kcsz} \cdot f_c^{ram} \leq z_R \quad \forall s \in \mathcal{S}, z \in Z \tag{3}$$

All instances of each VNFC must be assigned:

$$\sum_{s \in \mathcal{S}, z \in Z, k \in \alpha(c)} z_{kcsz} = T_c \quad \forall c \in \mathcal{C} \tag{4}$$

The anti-affinity rule for each VNFC is expressed as:

$$\sum_{z \in Z, k \in \alpha(c)} z_{kcsz} \leq f_c^{res} \quad \forall c \in \mathcal{C}, s \in \mathcal{S} \tag{5}$$

A compute is used if at least one instance is assigned to:

$$z_{kcsz} \leq x_s \quad \forall s \in \mathcal{S}, z \in Z, c \in \mathcal{C}, k \in \alpha(c) \tag{6}$$

The binary constraints are as follows:

$$z_{kcsz} \in \{1, 0\} \quad \forall s \in \mathcal{S}, z \in Z, c \in \mathcal{C}, k \in \alpha(c) \tag{7}$$

$$x_s \in \{1, 0\} \quad \forall s \in \mathcal{S} \tag{8}$$

For small sized instances of the VNFC-PROTEC problem, the ILP provides good solutions (see Section IV). However, finding an optimal solution requires a prohibitive computation time as the problem is NP-complete. For large size instances, it is even impossible to find feasible solutions using the ILP. We thus propose, in the following, efficient heuristics and a coloring algorithm to solve the VNFC-PROTEC problem.

C. Lower bounds

In order to boost the subsequent algorithms, we provide a lower bound denoted by L_B on the minimum number of needed compute servers. Let z_C (respectively z_R) denote the maximum amount of virtual CPU cores (respectively memory units) available in a single zone. Since a feasible deployment must satisfy a set of constraints, it holds that:

$$L_B = \max \left(\frac{\sum_c T_c f_c^{cpu}}{|Z| \cdot z_C}, \frac{\sum_c T_c f_c^{ram}}{|Z| \cdot z_R}, \max_c \left\lceil \frac{T_c}{f_c^{res}} \right\rceil \right) \tag{9}$$

While the first two quantities are derived from the resource allocation of the VNFCs under the capacity constraints of the computes nodes, the third quantity stems from the anti-affinity deployment constraint.

D. Heuristics

Slice sorting. Since the network vector is multidimensional, we study the effect of sorting the set of VNFCs that occur within a slice over the placement allocation. A widely used approach in one-dimensional placement algorithms consists in sorting the instances in decreasing order by their size. We thus consider a descending lexicographical ordering rule denoted by $\mathcal{R}$ based on the network vector dimensions: f^{cpu}, f^{ram} and f^{res}. Each of the six possible rules alternates between giving priority to capacity constraints satisfaction and anti-affinity resolving. Let $F_{\mathcal{R}}$ denote the first dimension over which the instances are sorted within the slice according to $\mathcal{R}$.

Compute scheduling. In addition to minimizing the number of compute servers, it is also of relevant importance to allocate the VNFCs evenly across the infrastructure with respect to the resources. We propose different algorithms which determine differently how to dispatch the VNFCs across the computes.

- The ORDERED-VECTOR greedy algorithm extends the First Fit Decreasing approach which was shown to be an efficient fast heuristic for one-dimensional resource placement. It operates by first sorting the slice according to a rule $\mathcal{R}$. Then, at each step, a VNFC is always placed over the first compute server which satisfies both the capacity and resiliency constraints. Furthermore, we increment this algorithm with a swap procedure which first finds the set P_{min} of compute nodes with the minimum number of assigned set $C_{reallocate}$ of VNFCs. Then, it attempts to reallocate randomly each instance c in $C_{reallocate}$ across the computes in $\mathcal{S} \backslash P_{min}$ by swapping c with a random feasible instance among the already placed instances in $\mathcal{C} \setminus C_{reallocate}$. We obtain a new algorithm denoted by ORDERED-VECTOR-SWAP.
- The LEAST-LOAD algorithm considers a sorted slice according to $\mathcal{R}$. Then, it sorts the compute servers in the decreasing order based on the free amount of available load along $F_{\mathcal{R}}$ where the load of a compute is defined as the total sum of the resource $F_{\mathcal{R}}$ across the instances that it currently hosts. For every VNFC, the instance is placed at the least loaded compute which doesn't break the resiliency constraint after sorting the computes. While

the algorithm starts with the predetermined lowerbound in Eq. (9), the SLIDING-LEAST-LOAD algorithm (see Algorithm 1) increases the L_B every time L_B computes are not enough to host the network slice.

- We consider the default placement algorithm used by the Nova scheduler, which is the main computing engine behind the OpenStack cloud computing infrastructure, denoted by OS-NOVA. The algorithm is similar to LEAST-LOAD with the following differences. First, the network slice is not sorted. Second, the load of the computes is defined as the average sum of the CPU and RAM resources. We increment OS-NOVA with two variants. While ORDERED-OS-NOVA sorts according to the rule $\mathcal{R}$, HEURISTIC-RANDOM does the same but also chooses randomly a compute from a subset of the least loaded computes by uniformly shuffling a fraction γ of the first sorted computes.

E. Colored Bin Packing

We model the VNFC-PROTECT problem by a generalized version of bin packing that we denote by COLOURED-BIN-PACKING. We consider an undirected graph $G = (V, E)$ where $V = \{1, ..., n\}$ represents the set of items of respective sizes $s_1, ..., s_n$ such that $s_i \in [0, 1]^d$ (for $d > 0$) and E the set of links between the items. The goal is to find a minimum-size partition of the items into independent sets (or *bins*) $B_1, ..., B_m$ of G such that $\|\sum_{i \in B_k} s_i\|_\infty \leq 1$, where $\|.\|_\infty$ denotes the standard l_∞ norm.

When $E = \emptyset$, COLOURED-BIN-PACKING reduces to the standard bin packing problem ($d = 1$). Therefore, the problem is NP-complete, APX-hard and no algorithm can achieve a better approximation ratio than $\frac{3}{2}$ (unless P=NP). It comes from the approximation hardness of standard bin packing through a reduction from the partition problem [18]. Moreover, if $\sum_{i \in V} s_i \leq 1$, we obtain the special case of the graph coloring problem, i.e., determining the chromatic number $\chi(G)$, which cannot be approximated within factor $N^{1-\varepsilon}$ for an input of N items, for all $\varepsilon > 0$ (unless P=NP) [19].

Equivalence. For each VNFC c with a given deployment scheme $T_c - f_c^{res} || f_c^{res}$, we dispatch its replicas in $\alpha(c)$ into f_c^{res} partitions where each partition is represented by a clique graph denoted by K_i^c, for $i = 1, ..., f_c^{res}$. The resiliency graph G is therefore a disjoint union of graphs $K_1, ..., K_\Gamma$ where $K_1 = \bigcup_{i=1}^{f_c^{res}} K_i^c$. By making the set of edges E depend on the deployment resiliency factor, we thus directly represent the anti-affinity rules of the VNFCs. In order to solve the COLOURED-BIN-PACKING, we leverage upon efficient algorithms for the bin packing tailored for a certain class of graphs. Since the resiliency graph is clearly a perfect graph, it belongs to the class of graphs for which one can find in polynomial time a coloring that uses a minimum number of colors. Therefore, we propose COLOURED-APPROX (see Algorithm 2) which operates over multi-dimensional network vectors by using different weighting systems. The main idea is to carefully remove small subgraphs of items which induce problematic instances due to the anti-affinity rule.

Algorithm 1 SLIDING-LEAST-LOAD

Input: VNFC set $\mathcal{C}$, Ordering rule $\mathcal{R}$
Output: A set of compute nodes $\mathcal{S}$ and a mapping of VNFC nodes to compute nodes $m_\mathcal{R} : \mathcal{C} \to \mathcal{S}$

Set L_B according to Eq. (9) ▷ Lower bound
$\mathcal{C}_\mathcal{R} \leftarrow$ sort $\mathcal{C}$ decreasingly using $\mathcal{R}$
unfeasible = True
while unfeasible **do**
  Add L_B compute nodes to $\mathcal{S}$
  for each VNFC c in $\mathcal{C}_\mathcal{R}$ **do**
    Mapped[c] $\leftarrow$ False
    for each compute node s in $\mathcal{S}$ **do**
      for each zone $z(s)$ **do**
        $\beta_z^{cpu} \leftarrow$ Free available CPU in z
        $\beta_z^{ram} \leftarrow$ Free available RAM in z
      $A(s) \leftarrow (\max_{z(s)} \beta_z^{cpu}, \max_{z(s)} \beta_z^{ram})$
    $\mathcal{S} \leftarrow$ sort $\mathcal{S}$ decreasingly per (A, $F_\mathcal{R}$)
    for each compute node s in $\mathcal{S}$ **do**
      if s can host c **then** ▷ Assignment constraints
        Assign c to s
        Mapped[c] $\leftarrow$ True
        Break
    if not Mapped[c] **then**
      $L_b \leftarrow L_b + 1$ ▷ Slide
      Break
  if Mapped[c] **then**
    unfeasible = False
return $\mathcal{S}$

Algorithm 2 COLOURED-APPROX

Input: Resiliency graph $G = (V, E)$, Ordering rule $\mathcal{R}$
Output: A set of compute nodes $\mathcal{S}$ and a mapping of VNFC nodes to compute nodes $m_\mathcal{R} : V \to \mathcal{S}$

Partition V into $P_H = (H_1, H_2)$ (See Sec. III-E)
Set the pairs E_H with w_H according to Eq. (10) and (11)
Create a weighted bipartite graph $\mathcal{B} = (P_H, E_H, w_H)$
Find a maximum weight matching $\mathcal{M}$ in $\mathcal{B}$
for each (c_1, c_2) in $\mathcal{M}$ **do**
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{c_1, c_2\}$ ▷ Joint placement
Collect in M the nodes from $\mathcal{M}$
$\mathcal{N} = V \setminus M$ ▷ Components not yet placed
Compute a feasible coloring with $\chi_\mathcal{R}(G[\mathcal{N}])$ colors
$K_\chi \leftarrow$ classes from the obtained coloring
for each class κ in K **do**
  Run ORDERED-VECTOR over the items in κ
  Add the obtained computes to $\mathcal{S}$
return $\mathcal{S}$

Weighting. The technique of weights has been used for one-dimensional bin packing problems and produces efficient algorithms [20], [21]. Each item i is assigned a weight w_i based on its size s_i and its packing in some fixed solution, such that the number of bins of the algorithm is close to the

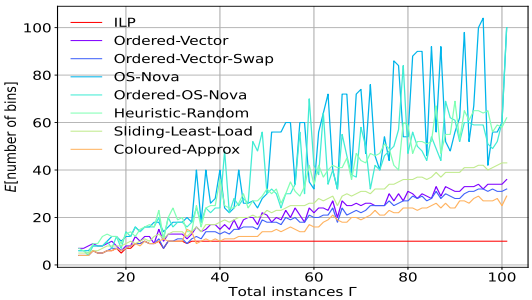

Fig. 1: Performance in terms of bins.

total sum of weights. We denote by $\mathcal{D}$ the set of network characteristics such as CPU, memory, disk usage.

The COLOURED-APPROX algorithm operates in several steps. First, an item i is said to be *large* (resp. *small*) if the quantity $max\{f_i^d, d \in \mathcal{D}\}$ in $[0,1]$ is larger (resp. smaller or equal) than $1/2$. The set of items V of the resiliency graph G is then partitioned into (H_1, H_2) such that H_1 (resp. H_2) collects the large (resp. small) items. Next, an edge (i,j) between two items i,j with i in H_1 and j in H_2 is added if the following conditions are satisfied:

$$\forall \in \mathcal{D}, f_i^d + f_j^d \leq 1 \text{ and } (i,j) \notin E \tag{10}$$

Eq. (10) captures both the capacity and resiliency constraints characterizing the situation where two items can be jointly placed in a bin. The edge (i,j) is associated with a weight w_{ij} defined as:

$$w_{i,j} = \mathcal{F}(f_i, f_j) + b_j \tag{11}$$

where f_i, f_j are the network vectors, $\mathcal{F}$ is an aggregator function (e.g., maximum, mean, weighted sum) such that it outputs a value in $[0,1]$, and b_j is a bias term associated with the small item. Several values of b_j were proposed in such a way to classify the items into intervals, then a weight is associated by either giving the same weight to all items in the same interval, or scaling them by a multiplicative factor [22], [23]. We adopt the special weight function in [16] which has the benefit of not rounding up the size (along each dimension) of an item to the next unit fraction. For each d in $\mathcal{D}$, let p_d be the integer such that $f_j^d \in (\frac{1}{p_d(p_d+p_d)}, \frac{1}{p_d}]$. The term b_j is set to $\frac{1}{\rho(\rho+1)}$ where $\rho = \max(p_d, d \in \mathcal{D})$. Note that as a special case, when $|\mathcal{D}| = 1$, the network vector is one-dimensional and COLOURED-APPROX is a $\frac{5}{2}$-approximation algorithm [16].

IV. PERFORMANCE EVALUATION

In this section, we first present our evaluation setup. Then, we provide a broader assessment of the various solutions proposed under different scenarios.

A. Evaluation setup

Slide demands. We consider the design and deployment of several slices in the network. Each slice has to implement a set of VNFs where each VNF is composed of a given number VNFCs. Each VNFC requires a specific amount of CPU, memory, and has anti-affinity constraints. Our aim is to evaluate the performance of the different placement algorithms (see Sec. III) over the resource allocation of a given slice across an infrastructure. The latter is made up of a given number of compute servers where each compute is composed of a set of zones with a given CPU and memory capacities.

Metrics. We compare the different algorithms in terms of the expected number of bins obtained, the running time, and the distribution of free resources (referred to by wastage) across the different bins. Since the number of bins varies across the algorithms under the same slice, to analyze the wastage, we consider the variation coefficient C_v which is the ratio of the standard deviation to the mean and captures the level of dispersion around the mean of free resources of the compute servers after the placement.

B. Evaluation scenario

Generation. We run our experiments over a synthetic scenario based on a random generator of network slices. For a fixed number of zones (between 1 and 5), the generator provides a network slice made up from a random number of VNFCs. A VNFC has a given cost determined by its network vector where f^{cpu}, f^{ram} are selected uniformly at random between 0 and 1, the number of replicas T_c (which comprises the backup instances) randomly between 1 and 10, and the resiliency factor f^{res} randomly between 1 and T_c. Therefore, a given slice is made up of σ instances where $\sigma = \sum_{c=1}^{\Gamma} T_c$.

We also validate our results over a real scenario based on traces from a real ISP network.

Settings. We set $\gamma = 0.25$ and the time limit for the ILP solver to 10 hours. Moreover, we chose among two families of aggregators: $\mathcal{F}^d_{mean}(f_i, f_j) = \frac{f_i^d + f_j^d}{2}$ for each dimension d and $\mathcal{F}_{mean}(f_i, f_j) = \frac{\sum_d f_i^d + f_j^d}{4}$, in addition to $\mathcal{F}^d_{max}(f_i, f_j) = \max(f_i^d + f_j^d)$ for each dimension d and $\mathcal{F}_{max}(f_i, f_j) = \max(\{f_i^d, f_j^d, d \in \mathcal{D}\})$.

Results. As shown in Fig. 1 and Fig. 2, only a limited subset of our instances have been submitted to the ILP solver, which turned out to need a generous time-limit of 9 hours to be able to solve optimally a mere subset of 34 instances (the starting point of the red straight lines indicate the moments where we stopped taking into account the ILP algorithm). Therefore, the ILP does not scale and becomes quickly irrelevant as the size of the slice increases. Fig. 2 shows that, unlike the ILP, all the various algorithms remain very fast as the difficulty of the network slices increases.

We note that the quality of OS-NOVA regularly deteriorates in terms of bins. Moreover, ORDERED-OS-NOVA being relatively better than OS-NOVA shows the importance of sorting the slice before the placement. On top of the VNFCs sorting, shuffling the top least loaded compute nodes provides an apparent improvement as demonstrated by HEURISTIC-RANDOM. This is explained by the fact that in the default version of OS-NOVA, the compute servers are sorted evenly by f^{cpu} and f^{ram}. However, the pre-sorting of the slice implies that sorting the computes by $F(\mathcal{R})$ of the ordering rule is more beneficial as SLIDING-LEAST-LOAD unveils. The

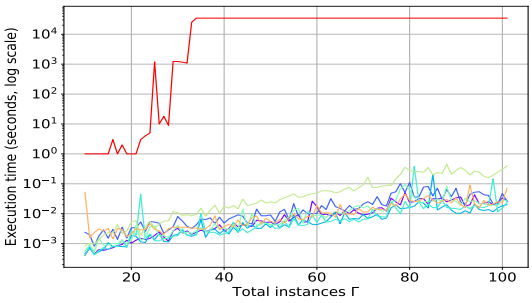

Fig. 2: Performance in terms of running time.

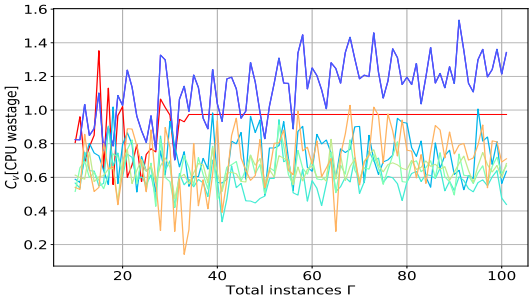

Fig. 3: Performance in terms of CPU wastage.

latter records quite honorable results and, as expected, spreads the components evenly across the compute nodes (Fig. 3 considers the CPU as in the given slice $F(\mathcal{R})$ =CPU). While the algorithm remains fast, it is outperformed by the others (aside the ILP). This shows that even an overestimation L_B of the necessary number of compute notes does not affect the minimization problem of the least load paradigm.

As expected, the swap procedure improves over ORDERED-VECTOR while remaining fast. On the other hand, while COLOURED-APPROX and ORDERED-VECTOR-SWAP obtain the best performances in bins, the coloring approach spreads the wastage evenly in contrast with the ORDERED-VECTOR-SWAP which obtains the worst performance with a relatively high C_v. We recall that in the one-dimensional case, the latter provides a guaranteed theoretical approximation performance. The benefit of the weighting techniques lies in not giving the priority to resolving capacity constraints, but also taking into account in a joint manner the anti-affinity rule. We observed that the scenario studied under the real ISP traces provided a similar assessment with, in particular, the scalability and efficiency of the COLOURED-APPROX as the number of zones increases. Last, we note that when the matching is empty, COLOURED-APPROX becomes equivalent to ORDERED-VECTOR with the major difference that making the feasible coloring over the resiliency graph depend on $\mathcal{R}$ ensures that the distribution of wastage remains reasonable.

V. CONCLUSION AND FUTURE WORK

In this paper, we presented a coloring based approach to carry out an efficient placement of network slices under a general protection mechanism which preserves failure tolerance of the virtual components. Our experiments over synthetic and real traces show that the proposed algorithms outperform the default placement algorithm in OpenStack while remaining very fast. A future work will investigate the asymptotic approximation performance of our algorithms and adapt them under an online provisioning of network slices.

REFERENCES

[1] NFV ISG. Network functions virtualisation (nfv) - network operator perspectives on industry progress. *Update White Paper*, 2013.

[2] Hyojoon Kim and Nick Feamster. Improving network management with software defined networking. *IEEE Communications Magazine*, 51(2):114–119, 2013.

[3] David Gureya, Joao Neto, Reza Karimi, Joao Barreto, Pramod Bhatotia, Vivien Quema, Rodrigo Rodrigues, Paolo Romano, and Vladimir Vlassov. Bandwidth-aware page placement in numa. In *IEEE IPDPS*, pages 546–556. IEEE, 2020.

[4] Juliver Gil Herrera and Juan Felipe Botero. Resource allocation in nfv: A comprehensive survey. *IEEE Transactions on Network and Service Management*, 13(3):518–532, 2016.

[5] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications surveys & tutorials*, 18(1):236–262, 2015.

[6] Andrea Tomassilli, Giuseppe Di Lena, Frédéric Giroire, Issam Tahiri, Damien Saucez, Stéphane Pérennes, Thierry Turletti, Ruslan Sadykov, François Vanderbeck, and Chidung Lac. Design of robust programmable networks with bandwidth-optimal failure recovery scheme. *Computer Networks*, 192:108043, 2021.

[7] Andrea Tomassilli, Nicolas Huin, Frederic Giroire, and Brigitte Jaumard. Resource requirements for reliable service function chaining. In *IEEE ICC*, pages 1–7. IEEE, 2018.

[8] Jian Kong, Inwoong Kim, Xi Wang, Qiong Zhang, Hakki C Cankaya, Weisheng Xie, Tadashi Ikeuchi, and Jason P Jue. Guaranteed-availability network function virtualization with network protection and vnf replication. In *IEEE GLOBECOM*, pages 1–6. IEEE, 2017.

[9] Gamal Sallam, Zizhan Zheng, and Bo Ji. Placement and allocation of virtual network functions: Multi-dimensional case. In *IEEE ICNP*, pages 1–11. IEEE, 2019.

[10] W Fernandez De La Vega and George S. Lueker. Bin packing can be solved within 1+ ε in linear time. *Combinatorica*, 1(4):349–355, 1981.

[11] Narendra Karmarkar and Richard M Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Symposium on Foundations of Computer Science*, pages 312–320. IEEE, 1982.

[12] Gerhard J Woeginger. There is no asymptotic ptas for two-dimensional vector packing. *Information Processing Letters*, 64(6):293–297, 1997.

[13] Chandra Chekuri and Sanjeev Khanna. On multidimensional packing problems. *SIAM journal on computing*, 33(4):837–851, 2004.

[14] Nikhil Bansal, Alberto Caprara, and Maxim Sviridenko. Improved approximation algorithms for multidimensional bin packing problems. In *IEEE FOCS*, pages 697–708. IEEE, 2006.

[15] Klaus Jansen. An approximation scheme for bin packing with conflicts. *Journal of combinatorial optimization*, 3(4):363–377, 1999.

[16] Leah Epstein and Asaf Levin. On bin packing with conflicts. *SIAM Journal on Optimization*, 19(3):1270–1298, 2008.

[17] Leah Epstein, Asaf Levin, and Rob van Stee. Two-dimensional packing with conflicts. *Acta Informatica*, 45(3):155–175, 2008.

[18] Michael R Garey and David S Johnson. Computers and intractability. 1979.

[19] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 681–690, 2006.

[20] Klaus Jansen and Sabine Öhring. Approximation algorithms for time constrained scheduling. *Information and computation*, 132(2):85–108, 1997.

[21] EG Co man Jr, MR Garey, and DS Johnson. Approximation algorithms for bin packing: A survey. *Approximation algorithms for NP-hard problems*, pages 46–93, 1996.

[22] Steven S Seiden. On the online bin packing problem. *Journal of the ACM*, 49(5):640–671, 2002.

[23] Brenda S Baker and Edward G Coffman, Jr. A tight asymptotic bound for next-fit-decreasing bin-packing. *SIAM Journal on Algebraic Discrete Methods*, 2(2):147–152, 1981.