Noname manuscript No.
(will be inserted by the editor)

Customizable Deployment of NFV Services

Vinicius Fulber-Garcia · Alexandre Huff · Leonardo da C. Marcuzzo · Marcelo C. Luizelli · Alberto E. Schaeffer-Filho · Lisandro Z. Granville · Carlos R. P. dos Santos · Elias P. Duarte Junior

Received: ????/??/?? / Accepted: ????/??/??

Abstract Network Functions Virtualization (NFV) promotes a paradigm shift in the core network, by enabling the execution of network functions on a virtualized software plane instead of on dedicated hardware. Despite its benefits, NFV introduces new challenges, of which we highlight those related to the deployment of virtualized network services. Current NFV deployment solutions (*i.e.*, those for composition, embedding, and scheduling) are usually limited to optimize hard-coded criteria, and cannot be customized to address specific demands defined by both network operators and NFV-as-a-Service customers. In this paper, we present a customizable NFV deployment framework that allows multiple criteria and multiple objectives to be applied to service composition, embedding, and scheduling. We evaluate the proposed framework integrated to deployment solutions specified in the literature. A case study is presented for the customized deployment of a traffic control and security service, and demonstrates the flexibility and effectiveness of the proposed framework.

Vinicius Fulber-Garcia, Elias P. Duarte Junior
Federal University of Paraná
E-mail: {vfgarcia, elias}@inf.ufpr.br

Alexandre Huff
Federal Technological University of Paraná
E-mail: alexandrehuff@utfpr.edu.br

Leonardo da C. Marcuzzo, Carlos R. P. dos Santos
Federal University of Santa Maria
E-mail: {lmarcuzzo, csantos}@inf.ufsm.br

Marcelo C. Luizelli
Federal University of Pampa
E-mail: marceloluizelli@unipampa.edu.br

Alberto E. Schaeffer-Filho, Lisandro Z. Granville
Federal University of Rio Grande do Sul
E-mail: {alberto, granville}@inf.ufrgs.br

Keywords Network Functions Virtualization · Network Service · Service Function Chain · Framework · Deployment · Composition · Embedding · Scheduling

1 Introduction

Network Functions Virtualization (NFV) [1] is a networking paradigm that allows the execution of network functions in a software plane enabled by current virtualization technologies (*e.g.*, containers and virtual machines) instead of on physical and typically proprietary hardware. It is a consensus that the NFV paradigm reduces the capital and operational costs of the network infrastructure and improves the flexibility of the life cycle of network functions [2] [3]. Standards have been published for NFV by the European Telecommunications Standards Institute (ETSI) which has defined a reference architecture for network service orchestration and management [4]. The Internet Engineering Task Force (IETF) has published, among others, documents defining service function chains [5] and the network service header [6].

Nevertheless, effective techniques for creating, setting up, and managing virtualized network functions and services must be developed so that NFV can fulfill its potential. In particular, there are multiple challenges related to virtualized network service deployment. The deployment of a network service includes every operation that is executed between service acquisition (or development) and service execution [7]. The NFV deployment process can be seen as a Resource Allocation (RA) problem [8] that consists of three tasks: composition, embedding, and scheduling. These tasks present a precedence order: first, the `composition` task is executed to create a network service topology with the network functions that will be, afterwards, `embedded` in the underlying network and, finally, they are `scheduled` for execution on virtual machines or containers running on commercial off-the-shelf hardware. In this work we claim it should be possible to tailor each of these tasks to specific needs and policies of both network operators and NFV-as-a-Service customers.

As the optimization problem that corresponds to NFV deployment is NP-hard [8], several solutions have been proposed for individual tasks of the deployment process, to circumvent the complexity restrictions. These solutions are often based on heuristics that reach approximate solutions with polynomial complexity and rely on different metrics to drive the deployment of network functions (*e.g.*, network function features, network infrastructure characteristics, and customer policies). However, to the best of our knowledge, all solutions in the literature employ hard-coded metrics. This static strategy is often not the best, given the variety of network service types, the variety of virtualized environments (*e.g.*, cloud, fog, edge), and specific customer demands. Heterogeneous deployment scenarios require customizable deployment solutions, given the multiple concurrent requirements that are relevant when the deployment is done. Current strategies force network operators to adapt

to what is available, which can certainly lead to under-optimized deployment results.

NFV-as-a-Service (NFVaaS) providers – such as T-NOVA [9] and FENDE [10] – provide solutions for virtualized network service deployment. These solutions are often based on performance parameters, including server overhead, energy consumption, and transmission delays. It is possible to say that NFVaaS providers are mostly concerned with network service embedding and scheduling tasks, given that their priority is the operation and management of their virtualization environments (*i.e.*, not the network services). We argue that the development and adoption of customizable deployment solutions can promote a priority shift – from provider to customer. Of course, provider requirements must still be taken into consideration, but together with customer objectives.

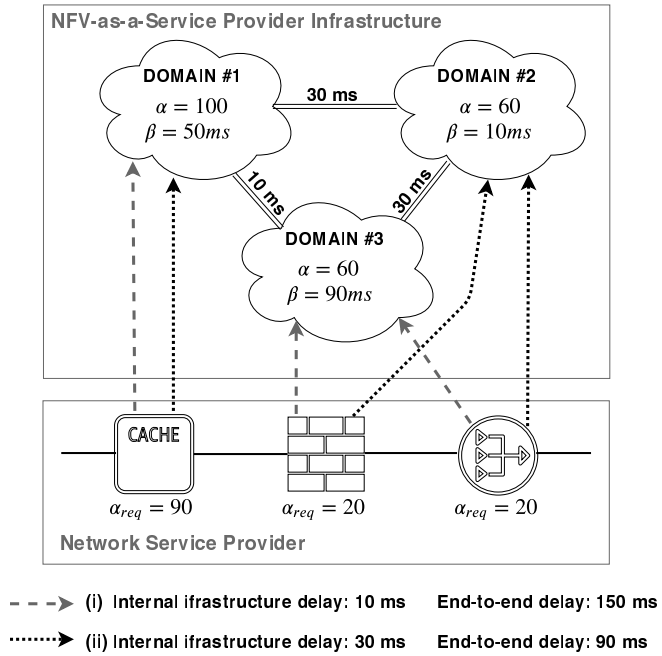

Fig. 1 An Example of Service Function Chain Embedding

Figure 1 depicts a scenario that exemplify the main differences between static and customizable NFV deployment strategies. The scenario consists of deploying a simple virtualized video service composed of three network functions (cache, firewall, and Network Address Translator) as presented in [11]. This service must be embedded in a multi-domain environment with three distinct domains supported by an NFVaaS provider. Also, the video service provider aims to minimize the transmission delay between the network function that implements the cache and the customers. The NFVaaS provider offers

two embedding solutions: (i) a static solution that minimizes the transmission delay among the available domains; and (ii) a customizable solution configured with multiple evaluation metrics chosen from a catalog or directly informed by the video service provider through a standard document, such as a Service Function Chaining Request (SFCR) [12]. Each domain is shown with the average transmission delays to the other domains (double lines), the amount of resources available (α), and the average transmission delays between the domains and customers of the video service (β). Finally, each network function is shown along with the required computational resources for its instantiation (α_{req}).

If the video service provider opts for the static solution to embed the service, the transmission delay between the available domains is minimized by allocating the cache function to `DOMAIN #1` and the other functions to `DOMAIN #3` (scenario i - dashed arrows). This alternative results in an internal transmission delay of 10 milliseconds. In contrast, consider that the video service provider chooses the customizable solution configured to minimize two metrics: the first of which the transmission delay among domains (from the NFVaaS provider catalog) and the second the average transmission delay between customers and domains (video service provider proprietary information). In this case, a different service mapping is returned. The network function that implements the cache is mapped to `DOMAIN #1` and other functions are allocated to `DOMAIN #2` (scenario ii - dotted arrows). This second mapping results in an internal transmission delay of 30 milliseconds, but an end-to-end transmission delay equal to 90 milliseconds (we consider that the delay within a domain is negligible). This simple example shows that the end-to-end delay of the customizable approach (90ms) is much lower than that of the static solution (150ms).

Of course, the deployment metrics must be carefully selected to maximize the results. In the previous example, the static solution did return an optimized result (according to the metrics it had available), but that was not enough to fulfill the video service provider requirements. By allowing evaluation metrics to be chosen, the video service provider dynamically indicates which features should be evaluated and optimized, thus achieving a service mapping that better fits the objectives (*i.e.*, minimize the transmission delays between the network function that implements the cache and the service users). Different parts may have access to different metrics. However, as mentioned before, there are currently no customizable NFV deployment solutions, and worse, several challenges have to be addressed before such a solution is specified. The main challenge can be expressed in the following question: *how to evaluate different sets of metrics for the particular optimization strategy (i.e., deployment solution) that is being used?* Note that an arbitrary number of metrics can be used, and furthermore it is necessary to determine how to combine different metrics, for instance, by defining different weights to be applied in each case to the metrics. It is also important to define a way to evaluate the results quantitatively.

In this paper, we present a framework to allow the development of customizable deployment solutions. We introduce a model to process a variable set of metrics, as well as to define their weights and generate a single value as an evaluation of the result. We apply the proposed model to extend deployment solutions found in the literature. Thus, these solutions become customizable, and can be evaluated by comparing with the original results. The proposed framework has two objectives. First, the evaluation of arbitrary metrics is specified as a mono-objective maximization problem (*Objective 1*). In this way, we simplify and standardize the evaluation of the diverse deployment solutions. The second objective (*Objective 2*) is to decouple metric evaluation from other operations of the deployment solutions.

The main contributions of this work are summarized as follows:

- **Customizable NFV Deployment**. We introduce an NFV deployment framework that allows multiple different metrics to be applied on-demand depending on specific objectives defined by customers and providers;
- **A Metric Evaluation Method**. We propose a strategy to evaluate sets of metrics to use for a specific composition, embedding, or scheduling. A single quantitative index is employed to allow the comparison of different solutions;
- **Extensions of Current Deployment Solutions**. We evaluate the proposed solution with a series of experiments using NFV deployment solutions of the literature.

The rest of this paper is organized as follows. Section 2 presents preliminary definitions as well as relevant related work. Next, we describe and specify the metric evaluation method in Section 3. In Section 4, we present the implementation of the proposed method. In Section 5, we describe experiments executed to evaluate the proposed method and demonstrate how it can be used to extend deployment solutions of the literature. Conclusions follow in Section 6.

2 Definitions & Related Work

This section presents an overview of paradigms and technologies, as well as the concepts from statistics used in the work. Next, relevant related work is presented.

2.1 An Overview of Network Functions Virtualization

Network Functions Virtualization (NFV) employs virtualization technologies, such as full virtualization, paravirtualization, and container virtualization, to execute network functions (NF), instead of executing them on dedicated hardware. Despite the fact that there are benefits in using dedicated hardware

(*e.g.*, fast packet processing), the NFV paradigm enhances network flexibility and reduces both CApital and OPerational EXpenditures (CAPEX and OPEX) [1]. NFV supports the creation of complex network services through the composition of multiple Virtualized Network Functions (VNF) in service topologies [13]. The creation and instantiation of network services involve a series of tasks that collectively define the service deployment process. These tasks include different types of actions, from service acquisition (or development) to its execution and maintenance in the virtualized environment.

Overall, network service deployment consists of three main tasks of a typical Resource Allocation problem [8]: composition, embedding, and scheduling. These tasks prepare the execution of a network service taking into account its performance requirements, given a set of metrics and objectives (*e.g.*, avoid service overhead, maximize throughput, minimize latency, and reduce memory consumption). In particular, the `composition` task is responsible for defining the service topology; `embedding` allocates the service topology into the physical substrate; and `scheduling` determines on which processors network functions are to be executed, as well as the corresponding deadlines, among other features.

The same network service can be provided and executed on different topologies, using different embedding and scheduling strategies. In this work, we refer these multiple possible solutions for a given deployment as `candidates`. Different deployment solutions evaluate the candidates using different `metrics`, such as throughput, delay, priorities, financial cost, resource usage, and CPU time consumption. The evaluation of a metric generates `partial results` that represent a value that summarizes how good a single metric is for a particular candidate. The idea is to enable the identification of the best candidates with an objective evaluation. The `objective` indicates whether an optimization metric is to be maximized or minimized. Notice that different metrics can be evaluated according to multiple different objectives. Deployment solutions that employ multiple criteria must correlate partial results obtained from the application of different metrics taking into account costs and benefits so that the best candidate can be identified.

In order to decide the best alternative to deploy a network service according to a set of evaluation metrics and objectives, it is possible not only to adopt a naive strategy based on exhaustive search, but also heuristic alternatives. Running exhaustive search involves generating all possible candidates, that are subsequently evaluated to find the global optimal candidate. This process has exponential cost. Heuristic alternatives, on the other hand, are more efficient, often polynomial, but return candidates that are not guaranteed to be the best, but should be close enough. Examples of exhaustive search deployment solutions can be found in [14] and [12], while examples of heuristic search solutions are described in [15] and [16].

Overall, three stages summarize the execution of deployment solutions: (i) definition, (ii) evaluation, and (iii) classification of candidates. Typically, all the tasks of the network service deployment require the execution of these stages. The first stage (`definition of candidates`) explores and exploits

the search space to define candidates that solve a deployment problem. At this point, candidates can be complete (*e.g.*, non-iterative candidate definition, such as in solutions based on exhaustive methods and genetic heuristics) or partial (*e.g.*, iterative candidate definition, such as in solutions based on greedy search and A* heuristics) according to the definition strategy adopted. The second stage (`evaluation of candidates`) consists of evaluating metrics and objectives for the previously defined set of candidates, thus indicating suitability parameters that can be used for their classification. At last, in the third stage (`classification of candidates`), the suitability parameters are jointly inspected to determine the best candidates which are either used to update the algorithm for the next iteration or just sent as output to the end-user. Note that the framework proposed in this work tackles the second and the third stages, thus enabling the adoption of any candidate definition strategy in the first stage.

2.2 An Overview of Feature Scaling, Weighing, & Pearson Correlation

Recent work on virtualized network services employ statistical methods for anomaly detection [17], monitoring [18], and service scaling [19]. Statistical methods can be used to correlate metrics and can improve the accuracy of the decision-making process during different phases of the NFV life cycle. Next, we give a brief overview of feature scaling, weight normalization, and Pearson correlation, all of them used in our proposed framework.

Feature scaling methods are used to transform the ranges of independent variables. Among these methods, the `Proportion Of Maximum Scaling (POMS)` [20] allows the transformation of a set of samples S in a range $[S_{min}, S_{max}]$ to the range $[0; 1]$. This method is useful to translate samples obtained for different metrics, each defined on a particular range, to a common neutral range. POMS is executed to map samples to a range from zero to the maximum absolute distance between any pair of samples ($[0; S_{max} - S_{min}$). For each sample, the proportional difference to the maximum absolute distance is computed to define the mapped value. Equation 1 shows how the transformation occurs. Observe that POMS overlooks the data dispersion in the original sample set, so data stabilization techniques must be previously applied when outliers affect the solutions.

$$\forall s \in S : \frac{s - S_{min}}{S_{max} - S_{min}} \tag{1}$$

`Weighing` is a method used to adjust samples that differ in terms of the relative importance that they have on a computation. A weighing factor is used to increase or decrease the influence of a particular sample. If done carelessly, factors can lead to a violation of the range of values required for the results. To avoid that, a `normalization` process must be employed. This process has two objectives: (i) map the weights w in set W from $[W_{min}; W_{max}]$ to $[0; 1]$; and (ii) ensure that the sum of all weights w in W is 1 ($\sum_{i=1}^{|W|} w_i = 1$). These

objectives are reached by dividing each individual weight w by the sum of all weights in W, as shown in Equation 2. The weight normalization guarantees that if the distinct samples are in a common range, the results will be also in this range regardless the original weight scale.

$$\forall w \in W : \frac{w}{\sum_{i=1}^{|W|} w_i} \tag{2}$$

The `Pearson correlation` coefficients (also called r or ρ) represent the degree of positive or negative linear correlation between two quantitative variables [21]. The Pearson coefficient of variables x and y is in the range $[-1; 1]$, where -1 and 1 indicate, respectively, a negative and a positive perfect match correlations between x and y. Furthermore, for a zeroed coefficient there is no linear correlation between x and y. Usual interpretations of Pearson coefficients are: 0.0 to +-0.3 (negligible correlation); +-0.3 to +-0.5 (weak correlation); +-0.5 to +-0.7 (moderate correlation); +-0.7 to +-0.9 (strong correlation); +-0.9 to +-1.0 (very strong correlation). Equation 3 shows how the Pearson coefficient is computed for two variables x and y, which defined by samples s in S. In summary, the Pearson coefficient is calculated by dividing the covariance of x and y (that provides information about the linear relation between the variables) by the square root of the product of the variance of the variables (that maps the coefficient to the range $[-1, 1]$, thus standardizing the interpretation of both the correlation strength and direction).

$$\frac{\sum_s^S (s_x - \overline{S_x}) * (s_y - \overline{S_y})}{\sqrt{\sum_s^S (s_x - \overline{S_x})^2 * \sum_s^S (s_y - \overline{S_y})^2}} \tag{3}$$

2.3 Related Work

Currently, many network service deployment solutions are available to compose [22,12,15,23,24], embed [14,25–28,16,29–31], and schedule [32–34] virtualized network services. Furthermore, other deployment solutions tackle multiple deployment tasks at the same time, for example, the composition/embedding [35,36] and the embedding/scheduling [37] of network services. Several of the existing solutions, such as [22,12,15,24,27,28,16,30,32,33,31], use optimization based on a single metric (*e.g.*, traffic ratio, latency, throughput, monetary cost, and resource usage). Other solutions, for example [23,14,29,34], employ more complex objective functions with two or more evaluation metrics. In all these cases, the metrics are defined in advance and are static, *i.e.*, cannot be modified when a new service is deployed. Recently, we presented a customizable deployment solution [38] that allows adaptive service composition based on exhaustive search.

Some solutions allow some limited configuration of the static metrics, so that they can be adapted on-demand. An example is the metric weighing of [25] and [26] that allows metrics to be weighed before the solution is computed.

Other solutions claim to allow "customization" in different contexts. For example, the proposal in [35] aims to customize the network services deployment by taking into account the heterogeneous operational behavior of distinct implementations of the same network function. Furthermore, the solution proposed in [25], besides providing a metric weighing mechanism, enables the operators to tune several pre-defined parameters of the algorithm. It is important to notice that although the UNIFY [39] and SONATA [40] projects allow pluggable deployment solutions, as they give a choice of predefined solutions, they do not allow the evaluation metrics to be customized. Additionally, some TELCOS enable end-users to customize features of the network service deployment. However, these solutions are proprietary and thus are not available in the literature. Arguably, different types of infrastructures, customer demands, and network services require more flexible deployment processes. The present work attempts to fill this gap by customizing the deployment evaluation metrics in the context of NFVaaS.

3 The Proposed Framework for Customizable NFV Deployment

In this section, we describe the proposed framework for network service deployment. The purpose of this framework is to allow the definition of a customizable set of metrics to optimize the composition, embedding, or scheduling of network services. The framework also defines an index, called Suitability Index (SI), that allows different candidates to be compared. Recall that a candidate is a possible solution for a deployment task. The Suitability Index shows how good each candidate is in terms of a single value in the [0;1] range. Considering the three stages of the network service deployment process described in the previous section, the proposed framework tackles the evaluation and classification of candidates, thereby adapting to any candidate definition strategy.

Let A be the set of metrics employed. Each metric α in A consists of a tuple $(objective, weight)$, where $objective$ can be $maximization$ or $minimization$, and $weight$ is a real number. Besides the set of metrics A, this function receives as input a set B of candidates. Each candidate $\beta \in B$ has an associated set G that contains the partial results obtained by computing metrics α for candidate β. A partial result $\gamma \in G$ is described as tuple $(\alpha, value)$, where $value$ is a real number corresponding to the partial result for metric α. The notation is summarized in Table 1.

In order to smooth normalized weights and reduce the bias among positively correlated metrics, function $prep(A, S, r)$ is defined. This is an optional function that preprocesses the positively correlated metrics (with the minimum value of the Pearson coefficient equal to r) in A taking into account a sample of partial results or benchmarks (S). To do that, let C a set of ς metrics clusters, so that positively linearly correlated metrics are in a common cluster ($|\varsigma| > 1$), and other metrics (non-positive correlated and non-correlated metrics) are in a dedicated cluster ($|\varsigma| = 1$). Finally, $eval(A, B)$ is the main

Table 1 Notations of Proposed Framework Model

Notation	Description
A	Set of metrics
α	A metric in A
B	Set of candidates for deployment
β	A candidate in B
G	Set of partial results for all α in A
γ	A partial result in G
C	Set of clusters of metrics α in A
ς	A cluster in C
S	Set of samples of partial results or benchmarks for all α in A
r	Reference Pearson coefficient
SI_β	Service index for candidate β
$min(X)$	Function that returns the minimum value in set (X)
$max(X)$	Function that returns the maximum value in set (X)
$sum(X)$	Function that returns the sum of values in set (X)
$comb(\binom{X}{2})$	Function that returns all the 2-element combinations of set X
$X;Y$	Function that returns the set operation of $X \cup \{Y\}$
$prep(A,S,r)$	Function that executes the proposed method to smooth metric bias
$eval(A,B)$	Function that executes the proposed evaluation/classification method

function of the proposed framework that triggers the execution of the customizable evaluation/classification method.

Three constraints must be satisfied to ensure the correctness of the execution of the proposed framework. First, there must be at least one metric to be evaluated (Equation 4). Thus, it is possible to compare the deployment candidates. Second, the weight of each metric must be greater than zero (Equation 5). The weights are normalized before they are processed, thus zero- and negative-weighed metrics are not allowed. Finally, exactly one partial result for each metric and each candidate should be available (Equation 6). In the end, the framework returns a comparable SI for all deployment candidates.

$$A \neq \emptyset \tag{4}$$

$$\forall \alpha \in A : \alpha.weight > 0 \tag{5}$$

$$\forall \alpha \in A : (\forall \beta \in B : \exists \gamma \in \beta.G \mid \gamma.\alpha = \alpha) \tag{6}$$

Function $prep(A,S,r)$ processes samples of partial results or metric evaluation benchmarks (S) to find positive correlations that have a minimum Pearson coefficient of r, creating clusters of correlated metrics (a cluster represents a single piece of information). The weight of a cluster is the sum of weights of all the metrics in the same cluster ($C.weight$), which creates an information bias. To solve the problem that the bias can represent (*i.e.*, overvalued information), the cluster weight is reduced to the maximum weight of any of its metrics, and this new weight is proportionally divided to each of the cluster's

metrics, taking into account their normalized original weights. Equation 7 employs function $max(\varsigma)$ that returns the maximum $weight$ of some metric α in ς.

$$\forall \varsigma \in C : (\forall \alpha \in \varsigma : \alpha.weight \leftarrow \frac{max(\varsigma) * \alpha.weight}{\sum_{\alpha}^{\varsigma} \alpha.weight} \tag{7}$$

Next, the surplus weight that result of this process (*i.e.*, the cluster's weight before the reduction minus the weight of the cluster after the reduction – $C_{surplus}$) is added to the weights of all metrics in a proportional way (regardless of the metric being in the cluster or not), as presented in Equation 8. It is important to note that the option for a weight smoothing process instead of straightforward metric exclusion is due to the fact that any $r < 1$ does not imply in a perfect match correlation. In this way, correlated metrics with a Pearson coefficient less than 1 (but greater than r) still provide some information, even if tiny bit of it, that may be relevant to the deployment process.

$$\forall \alpha \in A : \alpha.weight \leftarrow \alpha.weight + \frac{\alpha.weight * C.surplus}{1 - C.surplus} \tag{8}$$

Function $eval(A, B)$ consists of three steps: POMS, smoothing, and indexing. Consider functions $max(\alpha)$ and $min(\alpha)$ that return respectively the maximum and minimum $value$ of some partial result γ of a metric α in the candidate set B. Every partial result obtained for that metric is contained in the range $[min(\alpha); max(\alpha)]$. The first step (POMS) is responsible for mapping the metric results from that range $[min(\alpha); max(\alpha)]$ to the maximum absolute distance range $[0; max(\alpha) - min(\alpha)]$ and then to a common range $[0; 1]$. Equation 9 is used in the first step of function $eval(A, B)$. Observe that, because of the resulting POMS standard range, this step enables the adoption of different metrics defined on different ranges.

$$\forall \beta \in B : (\forall \gamma \in \beta.G : \gamma.value \leftarrow \frac{\gamma.value - min(\gamma.\alpha)}{max(\gamma.\alpha) - min(\gamma.\alpha)}) \tag{9}$$

Next, in the smoothing step, the metrics with minimization objectives are submitted to a complement function. This function transforms the POMS so that a minimization problem becomes a maximization problem. Thus the complement function maps the highest result from 1 to 0 and the lowest result from 0 to 1. Thus, the complement function enables a uniform interpretation of the results, as the entire evaluation consists only of maximization problems. Equation 10 shows the complement functions.

$$\forall \beta \in B : (\forall \gamma \in \beta.G \mid \gamma.\alpha.objective = minimization : \gamma.value \leftarrow 1 - \gamma.value) \tag{10}$$

The last step (indexing) performs the weighing of the results and computes the SI for each candidate. In the weighing phase, all metric weights are normalized in the range $[0; 1]$ as presented in Equation 11. This operation updates the set of metrics, so that the normalized weights indicate how representative the metrics are for the SI creation. Finally, the service indexes are computed for each candidate β (SI_β) as the sum of the corresponding weighed results (Equation 12).

$$\forall \alpha \in A : \alpha.weight \leftarrow \frac{\alpha.weight}{(\sum_{\alpha}^{A} \alpha.weight)} \tag{11}$$

$$SI_\beta \leftarrow \sum_{\gamma}^{\beta.G} \gamma.value * \gamma.\alpha.weight \mid \beta \in B_{std} \tag{12}$$

A candidate with the maximum SI (*i.e.*, 1) has achieved the best possible results for all metrics in comparison to the other candidates. In contrast, a candidate with the minimum SI (*i.e.*, 0) has got the worst results for every requested metric. Note that a specific evaluation may not result in maximum and minimum SIs, but still all indexes are in the range $[0; 1]$. The index conciliates multiple heterogeneous metrics, so that a set of deployment candidates can be compared and ranked. Finally, a candidate with the highest SI may not represent the global best result for a deployment instance (*i.e.*, in case the candidate definition is based on heuristics that do not guarantee global best is always achieved), but it does represent the local best result taking into account the available candidates and metrics.

4 Implementation Settings

The implementation of the proposed framework, called Classification and Holistic Evaluation Framework (CHEF), consists of three main functions: (i) `Weight Normalization`, (ii) `Prep`, and (iii) `Eval`. `WeightNormalization` is presented in Algorithm 1. It receives as input the set of metrics A and normalizes metric weights. This function consists of two phases: first, the sum of all weight values is computed and stored in variable *sum* (lines 2 to 5); next, the metric weights are iteratively normalized by the quotient of the weight and sum (lines 6 to 8); finally, the raw weights are replaced by their normalized counterparts in A. The `Eval` function, in turn, consists of the three steps of the proposed evaluation/classification method and is described next.

Algorithm 1 `Metric Weights Normalization`

1: **function** WeightNormalization($\boldsymbol{A}$)
2: $sum \leftarrow 0$ ▷ $O(1)$
3: **for each** α **in** A **do** ▷ $O(n)$
4: $sum \leftarrow sum + \alpha.weight$ ▷ $O(n)$
5: **end for**
6: **for each** α **in** A **do** ▷ $O(n)$
7: $\alpha.weight \leftarrow \frac{\alpha.weight}{sum}$ ▷ $O(n)$
8: **end for**
9: **end function**

The preprocessing function that smooths linear biases is described in Algorithm 2. Function $Prep(A, S, r)$ receives as input the set of metrics A, a sample set S of partial results or metric benchmarks, and a minimum Pearson coefficient r to conclude that a combination of metrics is correlated. A secondary function $Clustering(\alpha, checked, matches)$ creates clusters of correlated metrics implemented as a recursive routine based on depth-first search. `Clustering` receives a metric α, a set of already checked metrics (*checked*), and a set of tuples (α_x, α_y) producing as output pairs of correlated metrics (*matches*). First, the `Clustering` function includes the current metric α in the *checked* set (line 2) as well as in the result set α (line 3). Next, for each combination of correlated metrics that includes α as the first element (line 4), the current result set is recursively updated by joining with the set returned from $Clustering(\alpha_y, checked, matches)$ in line 5 (consider $X; Y$ as the set operation of $X \cup Y$). Note that due to the conditional "**not** α_y **in** *checked*" in line 3, this function is called exactly once for each correlated metric. At last, the cluster with positively correlated metrics is returned in line 7.

Function $Prep(A, S, r)$, which is the main function executed to smooth metric biases, consists of three steps: correlation (lines 11–14), clustering (lines 15–17), and smoothing (lines 18–30). First, a preprocessing cluster object is created in variable PC (line 11). This object is initialized with three empty sets and a zeroed integer: $.matches \leftarrow \emptyset$, $.clusters \leftarrow \emptyset$, $.check \leftarrow \emptyset$, and $.surplus \leftarrow 0$. In the correlation phase, combinations of 2-metrics ($comb\binom{A}{2}$) are processed taking into account the samples S to find their respective Pearson coefficient ($Pearson((\alpha_x, \alpha_y), S)$) in line 12. Observe that line 13, which includes a 2-metric correlation in $PC.matches$, is executed if some condition from line 12 is satisfied: a positive Pearson coefficient greater than r and metrics with the same objectives, or a negative Pearson correlation less than $-r$ and metrics with different objectives. Lines 15 and 16 summarize the clusterization step. Note that, as $PC.checked$ is updated after the `Clustering` function is executed, the recursion is called at most once for a particular metric. Line 18 normalizes the metrics' weight enabling the execution of the smoothing step. For each cluster with multiple metrics (line 19), the maximum and the sum of metric weights are defined (respectively, lines 20 and 21 – consider that $sum(\varsigma)$ returns the sum of weights of metrics α in ς). Also, the difference between these weights (line 22) is summed up to the surplus variable.

Algorithm 2 `Biases Smoothing Preprocessing`

```
1: function Clustering(α, checked, matches)
2:     check; α                                                          ▷ O(1)
3:     cluster ← {α}                                                     ▷ O(1)
4:     for each (α, α_y) in matches that (not α_y in checked) do         ▷ O(n^2)
5:         cluster; Clustering(α_y, checked, matches)                    ▷ O(n)
6:     end for
7:     return cluster                                                    ▷ O(1)
8: end function
9:
10: function Prep(A, S, ρ)
11:     PC ← PreprocessCluster()                                         ▷ O(1)
12:     for each (α_x, α_y) in comb(A, 2) that ((Pearson((α_x, α_y), S) ≥ ρ and α_x.objective =
    α_y.objective) or (Pearson((α_x, α_y), S) ≤ −ρ and α_x.objective ≠ α_y.objective)) do  ▷
    O(m^2 * n^2)
13:         PC.matches; (α_x, α_y)                                       ▷ O(n^2)
14:     end for
15:     for each α in A that (not α in C.checked) do                     ▷ O(n)
16:         PC.clusters; Clustering(α, PC.checked, PC.matches)           ▷ O(n^3)
17:     end for
18:     WeightNormalization(A)                                           ▷ O(n)
19:     for each ς in PC.clusters that (|ς| > 1) do                      ▷ O(n)
20:         ς_max ← max(ς)                                               ▷ O(n^2)
21:         ς_sum ← sum(ς)                                               ▷ O(n^2)
22:         PC.surplus ← PC.surplus + ς_sum − ς_max                      ▷ O(n)
23:         for each α in ς do                                           ▷ O(n^2)
24:             α.weight ← (α.weight * ς_max) / ς_sum                    ▷ O(n^2)
25:         end for
26:     end for
27:     for each α in A do                                               ▷ O(n)
28:         α.weight ← α.weight + (α.weight * PC.surplus) / (1 − PC.surplus)   ▷ O(n)
29:     end for
30: end function
```

The maximum weight is proportionally distributed to all the metrics (lines 23 and 24), replacing their original weights. Finally, lines 27 and 28 show the proportional redistribution of the surplus weight to all the metrics in A.

The evaluation function is described in Algorithm 3. It receives as input the set of metrics A and the set of candidates B. Note that every candidate β in B has information on the set of partial results G so that a metric α in A has a corresponding value γ in G. The `Eval` function consists of the three steps described in the previous section: POMS, smoothing, and indexing. To improve performance, the execution of these steps can overlap in the algorithm. The evaluation function can be described as follows. First, the metric weight normalization is done (line 2) and results are kept to be used later. Then, in lines 3 to 6, the result set SI is initialized and later it is returned as the output with the suitability indexes for each candidate. The partial result evaluation starts in line 7; the suitability indexes are iteratively computed by processing a single metric at a time for all candidates (line 10). The maximum and minimum partial results for each metric are then obtained as required by POMS (lines 8 and 9). Then, the partial results are processed by POMS (line 11), followed by

Algorithm 3 `Proposed Evaluation/Classification Method`

```
1: function EVAL(A, B)
2:     WeightNormalization(A)                                          ▷ O(n)
3:     SI ← ∅                                                          ▷ O(1)
4:     for each β in B do                                              ▷ O(m)
5:         SI; (β, 0)                                                  ▷ O(m)
6:     end for
7:     for each α in A do                                              ▷ O(n)
8:         α_max ← max(α)                                              ▷ O(n * m)
9:         α_min ← min(α)                                              ▷ O(n * m)
10:        for each β in B do                                          ▷ O(n * m)
11:            if α_max ≠ α_min then                                   ▷ O(n * m)
12:                β_γ.α.value ← (β_γ.α.value − α_min) / (α_max − α_min)   ▷ O(n * m)
13:            else                                                    ▷ O(n * m)
14:                β_γ.α.value ← 1                                     ▷ O(n * m)
15:            end if
16:            if α.objective = minimization then                      ▷ O(n * m)
17:                β_γ.α.value ← 1 − β_γ.α.value                       ▷ O(n * m)
18:            end if
19:            SI_β.value ← SI_β.value + β_γ.α.value * α.weight        ▷ O(n * m)
20:        end for
21:    end for
22:    return SI                                                       ▷ O(1)
23: end function
```

the computation of the complement in case there are metrics with minimization objective functions (lines 16 to 18). This is followed by weighing and SI update (line 19). Finally, the candidate suitability indexes are returned in line 22.

Consider that n is the total number of metrics and m is the total number of samples (S) or candidates (B), the complexity of presented functions can be computed as follows. Imported functions $max(X)$, $min(X)$, and $sum(X)$ have linear complexity ($O(n)$ or $O(m)$), function $comb\binom{X}{2}$ is n quadratic ($O(n^2)$), and function $Pearson((\alpha_x, \alpha_y), S)$ is quadratic on m. The Weight normalization function (Algorithm 1) is linear with complexity $f(n) = 4n + 1$ ($O(n)$). The clustering function (Algorithm 3) is quadratic with complexity $f(n) = n^2 + n + 3$ ($O(n^2)$). The prepossessing function to smooth of weight biases (Algorithm 3) is quartic with complexity $f(n, m) = (m^2 * n^2) + n^3 + 5n^2 + 6n + 1$ ($O(m^2 * n^2)$). The evaluation/classification function (Algorithm 3) is quadratic with complexity $f(n, m) = 10(n * m) + 2n + 2m + 2$ ($O(n * m)$).

5 Experimental Evaluation

In order to evaluate the proposed framework, CHEF was implemented using the Python3 programming language[1]. Experiments were run to evaluate the execution time and also to observe how easy it is to integrate CHEF to

[1] Available at https://github.com/ViniGarcia/NFV-FLERAS

existing deployment solutions[2]. A case study of the evaluation of multiple custom metrics is also presented. Results were obtained by running the proposed framework and deployment solutions on an Intel Core i5-3330@3.00Ghz server with 8GB RAM DDR3 running Debian 8. Experiments were executed 30 times with a confidence level of 95%.

5.1 Execution Time

The objective of this experiment is to evaluate CHEF in terms of the impact it has on the execution time of the deployment solution. We refer the reader to [15,28,16] for reference results regarding the typical execution times of deployment solutions. As a result, we show that the proposed framework presents predictable behavior when increasing the number of candidates and evaluation metrics. In the first experiment, we evaluated the total execution time for different combinations of metrics and candidates. We used random sets A of n metrics and B of m candidates as inputs to the `Eval` function and measured its execution time.

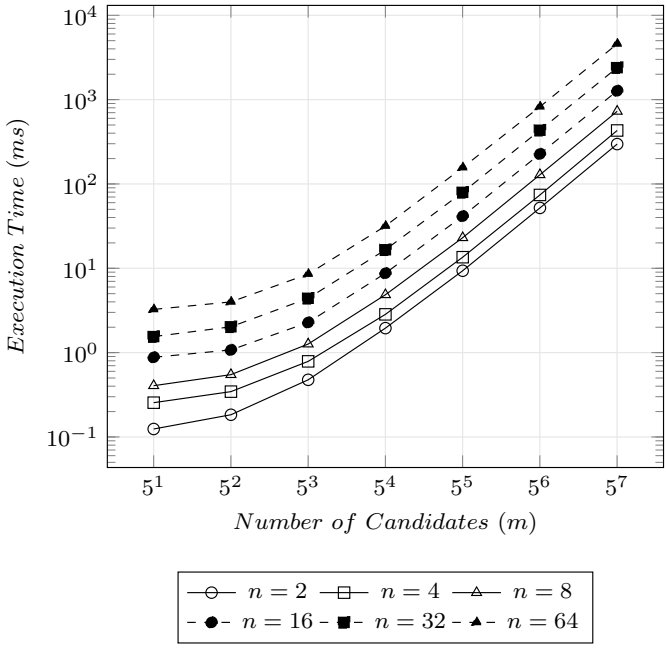

Fig. 2 CHEF Execution Time

[2] Available in branch "ChefExperiments" at https://github.com/ViniGarcia/NFV-FLERAS

Figure 2 shows the results obtained for the execution time of the proposed framework. In the experiment, we gradually increased the number of candidates evaluated by CHEF, varying from 5 to 78,125 candidates. These candidates were evaluated considering partial results coming from 6 different sets of metrics. Each metric set "i" is composed of 2^i metrics, where "i" varies from one to six. As the number of candidates (m, X-axis) increases, we show how the execution time performs according to the number of evaluation metrics (n). Also, as the number of evaluation metrics (n) increases, we show how the execution time performs for a set of candidates (m).

As can be seen in Figure 2, the execution time increases roughly linearly as the number of metrics and the number of candidates increase. As the number of metrics doubles, the execution time increases by 80%. Increasing the number of candidates five times causes the execution time to increase in average 280%. Note that results for small numbers of candidates ($m = 5$ and $m = 25$) do not follow the expected linear increase of the execution times. This occurs because, operations that are independent of the number of candidates, such as metric validations, dominate the overall processing time. After removing outliers, the execution time increases 76% as the number of metrics doubles, and 393% as the number of candidates is multiplied by 5. These results confirm the previously presented algorithm complexity of $O(n * m)$.

5.2 Integrating CHEF to NFV Deployment Solutions

In this experiment, we show that the proposed framework can be integrated to existing deployment solutions with few code changes, so that their functionality is extended to encompass the evaluation of multiple metrics with heterogeneous objectives. Furthermore, their native metrics can also be used with the same results. Two different deployment solutions were used: (i) the `Mehraghdam, Keller, and Karl (MKK) composer` [22] and (ii) the `Mijumbi, Serrat, Gorricho, Bouten, Turck, and Davy (MSGBTD) mapping` [32]. The first deployment solution uses both exhaustive and greedy heuristic searches to execute the definition stage in the composition of service topologies that minimize the traffic (*actually, the ratio between incoming and outgoing traffic*) going through the network functions. The second deployment solution, in turn, maps a service topology to a datacenter network by defining the server on which each network function must be placed. The definition stage of this service mapping solution is computed using a greedy heuristic that minimizes the service processing time and the monetary cost. These deployment solutions were selected so that we could evaluate the performance of the proposed framework applied to both the composition and the embedding tasks of the NFV deployment process. Furthermore, the evaluation also included both exhaustive and heuristic searches in the definition stage.

We implemented both the MKK composer and MSGBTD mapping in Python3. Next, we integrated the proposed framework in two ways that we call "`indexing`" and "`multi-criteria`". The indexing approach corresponds

to the two solutions using our evaluation/classification method. In this way, the solutions compute the proposed suitability index for each candidate with their original hard-coded evaluation metrics and objectives (*i.e.*, minimization of traffic ratio in MKK, and minimization of processing time and monetary cost in MSGBTD). In the multi-criteria approach, besides integrating the proposed evaluation/classification method, the solutions were also modified to allow the selection of on-demand metrics and evaluation objectives.

Experiments were performed to quantify the similarity of the codes before and after the integration of the proposed framework, in order to evaluate how much effort it takes to integrate the framework to the existing deployment solutions. In order to validate the extended versions, their results were compared to those of the original strategies, changing only the evaluation/classification method but using the same set of metrics. We executed experiments for 28 composition requests and 8 mapping requests. To check code similarity, the cosine technique was employed, which has been used for general data comparison, including both source code [41] and binary code [42]. The cosine technique converts general sets of data to vectors, which allows the estimation of the similarity level by the computation of the cosine of the angle between pairs of vectors. In this way, a result equal to 1 indicates that the compared sets of data are equal, while when the result is 0 the compared sets are totally different.

Table 2 CHEF Integration (Similarity)

Deployment Solution	**Code Similarity**		**Results Similarity**	
	Indexing	*Multi-criteria*	*Indexing*	*Multi-criteria*
Exhaustive MKK	0.996	0.982	1.000	1.000
Greedy MKK	0.993	0.986	1.000	1.000
Greedy MSGBTD	0.996	0.952	1.000	1.000

First, the original code of the deployment solutions is compared with the corresponding extended versions. As shown in Table 2, the amount of code that differs between the original version of the deployment solutions and the indexing ones corresponds to less than 0.5%. There is a small difference which is related to adding calls to CHEF in the evaluation and classification stages. The multi-criteria extension is also similar to the original version, but requires a few extra functions so that the differing code portions vary from 1.4% to 4.8%. Furthermore, both the indexing and multi-criteria versions reached the same deployment results as the original version for all submitted requests (100% of similarity). In other words, the new versions reach the same results as the original versions for the same metric set, which validates our strategy and the implementation.

The previous experiments evaluate the similarity of the different versions of the deployment solutions and their respective results. Now, we evaluate

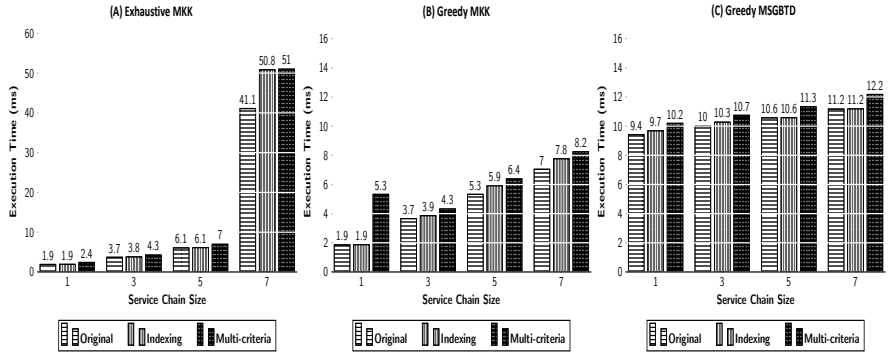

Fig. 3 CHEF Integration (Execution Time)

the time overhead of using the proposed framework integrated to the existing solutions. The overhead is caused because partial results are indexed and the solutions are adapted to the multi-criteria evaluation. To quantify this overhead, we conducted an experiment with four different service topologies, presented in Figure 3. In the composer experiment, the size of the service topology is the number of network functions requested, all of them in a linear segment that can be ordered in different ways (*i.e.*, functions can be allocated to any position of the chain). In the mapping test, the service topology size is the number of functions that must be mapped to a datacenter network with 20 servers. To keep the experiment fair, the multi-criteria version was configured to evaluate only metrics of the original solutions.

The solutions based on heuristic search presented a small increase of the execution time, as shown in Figure 3 (B) and (C). This occurs because of the small number of candidates to be evaluated at each algorithm iteration (*i.e.*, the decision of a position or server for a network function) – 7 candidates for composition, and 20 for mapping. On the other hand, the solutions based on exhaustive search show higher execution times in Figure 3 (A). This is due to the exponential growth of the number of candidates once all possible topologies are evaluated. This high number of candidates created by the exhaustive search solution also presents an overhead for the proposed framework, as it requires more time and thus resources to compute all the suitability indexes. According to these results, it is possible to conclude that CHEF can be applied to the solutions based on heuristic search, causing a small and predictable overhead.

5.3 Case Study: Customized Multi-criteria Composition

To evaluate the proposed framework in a customizable deployment, for different sets of metrics and objectives, we present a case study of the composition task. This case study considers the composition of a network service for security and traffic control. This service consists of five functions that are frequently deployed on datacenters [43]: Firewall (FW), Intrusion Prevention

System (IPS), Deep Packet Inspector (DPI), Traffic Shaping (TS), and Application Delivery Controller (ADC). The service topology is shown in Figure 4 and allows three network functions to be ordered in different ways (*i.e.*, IPS, DPI, and TS); this is exactly the optimization target for the composition. The experiments were executed with the multi-criteria version of the exhaustive MKK composer.

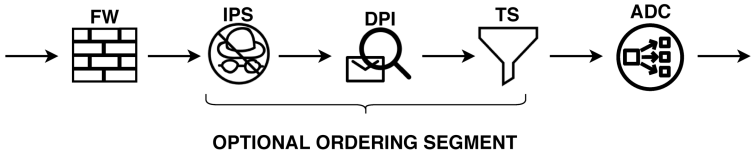

Fig. 4 Traffic Control and Security Service

In the case study, an NFVaaS customer requests the provider to deploy the network service. The customer gives as input the profile of network functions with three evaluation metrics: Average Traffic Ratio (ATR), Average Energy Consumption (AEC), and Average Processing Delay (APD). The profiles are presented in Table 3. The NFVaaS provider employs the multi-criteria version of MKK solution to compose the customer's service topology. In this way, the customer is actually requesting the analysis of four different composition scenarios, using minimization as objective function, and the same weight applied to all metrics: (i) ATR, (ii) ATR + AEC, (iii) ATR + APD, and (iv) ATR + AEC + APD. Table 4 shows the results returned for the customized scenarios.

Table 3 Network Function Profiles

	FW	**IPS**	**DPI**	**TS**	**ADC**
Avg. Traffic Ratio - ATR (Gbps Input/Gbps Output)	0.75	0.9	0.85	0.95	1.0
Avg. Energy Consumption - AEC (Watts/Gbps)	15	30	60	35	10
Avg. Processing Delay - APD (ms/pkt)	0.2	0.8	0.8	0.2	0.05

Table 4 Results of Customized Multi-criteria Compositions

Evaluation Scenarios	**Optimal Service Topology (MKK Exhaustive)**	**Suitability Index**
(i) ATR	FW → DPI → IPS → TS → ADC	1.000
(ii) ATR + AEC	FW → IPS → DPI → TS → ADC	0.868
(iii) ATR + APD	FW → DPI → TS → IPS → ADC	0.814
(iv) ATR + AEC + APD	FW → DPI → IPS → TS → ADC	0.774

Observe that the four customized scenarios resulted in three different service topologies. The first scenario aims to minimize the overall traffic ratio and returns the optimal result for this single metric (suitability index of 1.000). However, when the minimization of energy consumption and processing delay were also included by the framework in the second and third scenarios, two distinct chains are returned: in the first, the IPS function was allocated in the first position of the segment because of its low energy consumption when compared to DPI; in the second alternative, the TS was assigned to the second position of the optional ordering segment due to the fact that its traffic ratio reduction is close to that of the IPS, but it presents a much smaller processing delay. Finally, when all the evaluation metrics are processed together, the same service topology of the first scenario (*i.e.*, which just evaluates the traffic ratio metric) was returned. Note that, in the last three scenarios, none of the returned chains obtained suitability index equal to one. This is an indication that some metrics were better evaluated in other chains, but the final result represents the best cost-benefit taking into consideration the complete metric set.

Table 5 Results of Different Weighing of Customized Multi-criteria Compositions

Evaluation Scenarios (Metric [Weight])	Optimal Service Topology (MKK Exhaustive)	Suitability Index
(ii) ATR [1.0] + AEC [1.0]	FW → IPS → DPI → TS → ADC	0.868
(ii.1) ATR [0.5] + AEC [1.0]	FW → IPS → DPI → TS → ADC	0.912
(ii.2) ATR [1.0] + AEC [0.5]	FW → DPI → IPS → TS → ADC	0.908
(iii) ATR [1.0] + APD [1.0]	FW → DPI → TS → IPS → ADC	0.814
(iii.1) ATR [0.5] + APD [1.0]	FW → DPI → TS → IPS → ADC	0.826
(iii.2) ATR [1.0] + APD [0.5]	FW → DPI → IPS → TS → ADC	0.866
(iv) ATR [1.0] + AEC [1.0] + APD [1.0]	FW → DPI → IPS → TS → ADC	0.774
(iv.1) ATR [0.5] + AEC [0.5] + APD [1.0]	FW → DPI → IPS → TS → ADC	0.730
(iv.2) ATR [0.5] + AEC [1.0] + APD [0.5]	FW → DPI → IPS → TS → ADC	0.762
(iv.3) ATR [1.0] + AEC [0.5] + APD [0.5]	FW → DPI → IPS → TS → ADC	0.830

It is important to note that, in addition to the modification of the used set of metrics, it is possible to get different results by modifying the weights defined for each evaluation metric. Table 5 shows the result of increasing/decreasing different metric weights, and in different combinations, in the evaluation of new cases from the previously described scenarios ii, iii, and iv. This experiment consists of defining a dominant metric among the available ones. The domi-

nance imposes a reduction of the dominated metric (dM) weights to half of the dominant metric (DM) weight – *i.e.*, $dM.weight = \frac{DM.weight}{2}$. Observe that, in this way, the dominant metric will always have a higher normalized weight in relation to the dominated metrics. However, the impact of the dominant metric depends on the total number of metrics considered in the evaluation (*i.e.*, the impact reduces as the number of metrics increases). The original customized scenarios (ii, iii, and iv) were adapted to different test cases where a single particular metric is dominant, and all others are dominated.

For scenarios ii and iii presented in Table 5, the dominance of metrics AEC (ii.1) and APD (iii.1) – in comparison to ATR – resulted in the same service topologies observed when no dominance was applied to their metric weights (respectively, ii and iii). But, in the cases of ii.1 and iii.1, the suitability indexes did achieve a higher value than the corresponding cases ii and iii. When the dominance is applied to the ATR metric (ii.2, iii.2, as well as iv.3), in turn, resulted in a common service topology for all the tested scenarios. This example reflects the importance of this particular positioning of network functions to optimize the ATR metric (note that the positions are different of what was observed in both ii.1 and iii.1).

Another relevant result is the fact that the evaluation and classification of cases 1, 2, and 3 from scenario iv results in the same service topology, regardless of the chosen dominant metric. In this scenario, the dominance imposes a lower impact on the final result than it imposes on scenarios ii and iii. This occurs due to the evaluation of the three metrics (one dominant and two dominated) instead of two metrics (one dominant and one dominated). In this way, CHEF favors the positioning of network functions that can optimize multiple metrics for increasing the suitability index – for example, placing DPI in the second position of the topology improves both ATR and APD. In the same way, placing TS in the fourth position of the topology improves both ATR and AEC. Thus, this joint metric optimization establishes for scenario iv the worst mean (0.774) and the highest dispersion (percentage variation of 13.7%) of the suitability indexes among all tested scenarios. In comparison, scenario ii presents a suitability index mean of 0.896 and a dispersion of 5.1%, and scenario iii presents a suitability index mean of 0.835 and a dispersion of 6.4%.

With these results, the NFVaaS customer can choose the service topology composition that is best suited to his/her particular demands. For example, if green computing is a primary concern, the service topology from the second scenario is certainly the best (ii or ii.1). Otherwise, if the customer aims to optimize the quality of service regarding the delay metric, the third scenario should be chosen (iii or iii.1). Finally, if all the evaluation metrics should be taken into account, the last scenario is presented as the best option (iv).

With the popularization of the NFV paradigm, NFVaaS providers will have to deal with complex deployment requests of diverse types complying with customer policies. In this way, the evolution of current NFV deployment solutions and the development of new ones must consider adopting mechanisms

to deal with requirements defined on-demand. As shown in this section, CHEF is a feasible approach to accomplish this task.

6 Conclusion

Network Functions Virtualization has been proposed as a novel paradigm that allows the execution of network functions and services on a software plane by using virtualization technologies (*e.g.*, full virtualization, paravirtualization, container virtualization). However, despite the multiple potential benefits of this paradigm, such as higher flexibility for set up and management and lower capital and operational costs, many gaps have to be addressed before the potential of NFV is fulfilled. Some of these challenges are related to the NFV deployment process. The deployment process is executed to compose, embed, and schedule a virtualized network service according to optimization criteria, based on evaluation metrics. Current deployment solutions only allow a static hard-coded set of evaluation metrics to be used. Ideally, multiple metrics and particular demands from network operators and NFVaaS customers should be taken into consideration to deploy a complex service.

In this paper, we introduced a framework (called CHEF) for NFV deployment that uses a customizable evaluation/classification according to multiple heterogeneous metrics. This new multi-criteria framework can be integrated to existing solutions that execute any task related to NFV deployment (*i.e.*, composing, embedding, and scheduling). Considering that deployment solutions often consist of three stages (definition, evaluation, and classification of candidates), the proposed framework accomplishes both the evaluation and classification stages, given any definition stage. A deployment solution that adopts CHEF enables its users to customize the evaluation metrics as a particular service to be deployed. Furthermore, the evaluation objectives and relative weights are also customizable. We demonstrated and evaluated the integration of the proposed framework to two currently available deployment solutions (one for service composition, and another for service embedding). The results show that the deployment solutions present less than 5% of code dissimilarity between their original versions and the respective extended versions. Also, we show that an acceptable overhead is imposed to the execution time of the extended versions of the deployment solutions, even when exhaustive strategies are employed. Finally, we presented a case study that shows the differences in the deployment results as distinct metric sets are used, which illustrates that custom and multi-criteria evaluation can be indeed very effective.

Future work includes the integration of CHEF to other deployment solutions, thus further demonstrating its flexibility. Another future work is the investigation of a holistic deployment platform that enables, besides the customized evaluation of metrics, the customization of the entire deployment process. In such a platform, the sequence of stages of each deployment task would be defined on-demand. Thus, multiple tasks could be defined cooperatively so that the best cost-benefit is achieved for the complete deployment result. In

conclusion, we aim to natively provide customizable deployment solutions in the context of FENDE [10], and NIEP [44].

References

1. E. Hernandez-Valencia, S. Izzo, and B. Polonsky. How will NFV/SDN transform service provider opex? *Network*, 29(3):60–67, 2015.
2. W. Kellerer, A. Basta, P. Babarczi, A. Blenk, M. He, M. Klugel, and A. M. Alba. How to Measure Network Flexibility? A Proposal for Evaluating Softwarized Networks. *IEEE Communications Magazine*, 56(10):186–192, 2018.
3. A. Chiha, M. Van der Wee, D. Colle, and S. Verbrugge. Network Slicing Cost Allocation Model. *Journal of Network and Systems Management*, 28(3):1–33, 2020.
4. R. Cziva, C. Anagnostopoulos, and D. P. Pezaros. Dynamic, Latency-Optimal vNF Placement at the Network Edge. In *International Conference on Computer Communications*, pages 693–701, Honolulu, USA, 2018. IEEE.
5. J. Halpern and C Pignataro. Service Function Chaining (SFC) Architecture. Technical report, Internet Engineering Task Force (IETF SFC WG), 2015.
6. P. Quinn, U. Elzur, and C. Pignataro. Network Service Header (NSH). Technical report, Internet Engineering Task Force (IETF SFC WG), 2018.
7. L. Bondan, C. R. P. dos Santos, and L. Z. Granville. Comparing virtualization solutions for NFV deployment: A network management perspective. In *International Symposium on Computers and Communication*, pages 669–674, Messina, Italy, 2016. IEEE.
8. J. G. Herrera and J. F. Botero. Resource allocation in NFV: A comprehensive survey. *Transactions on Network and Service Management*, 13(3):518–532, 2016.
9. G. Xilouris, M. Kourtis, M. J. McGrath, V. Riccobene, G. Petralia, E. Markakis, E. Palis, A. Georgios, G. Gardikis, J. F. Riera, A. Ramos, and J. Bonnet. T-NOVA: Network functions as-a-service over virtualised infrastructures. In *Conference on Network Function Virtualization and Software Defined Network*, pages 13–14, San Francisco, USA, 2015. IEEE.
10. L. Bondan, M. F. Franco, L. Marcuzzo, G. Venancio, R. L. Santos, R. J. Pfitscher, E. J. Scheid, B. Stiller, F. De Turck, E. P. Duarte, A. E. Schaeffer-Filho, C. R. P. d. Santos, and L. Z. Granville. FENDE: Marketplace-Based Distribution, Execution, and Life Cycle Management of VNFs. *Communications Magazine*, 57(1):13–19, 2019.
11. W. Ding, W. Qi, J. Wang, and B. Chen. OpenSCaaS: an open service chain as a service platform toward the integration of SDN and NFV. *Network*, 29(3):30–35, 2015.
12. A. F. Ocampo, J. Gil-Herrera, P. H. Isolani, M. C. Neves, J. F. Botero, S. Latré, L. Zambenedetti, M. P. Barcellos, and L. P. Gaspary. Optimal Service Function Chain Composition in Network Functions Virtualization. In *Security of Networks and Services in an All-Connected World*, pages 62–76, Zurich, Switzerland, 2017. Springer.
13. V. Fulber-Garcia, E. P. Duarte, A. Huff, and C. R. P. d. Santos. Network Service Topology: Formalization, Taxonomy and the CUSTOM Specification Model. *Computer Networks*, 178:107337, 2020.
14. D. Dietrich, A. Abujoda, and P. Papadimitriou. Network service embedding across multiple providers with nestor. In *Networking Conference*, pages 1–9, Toulouse, France, 2015. IFIP.
15. J. Gil-Herrera and J. F. Botero. A scalable metaheuristic for service function chain composition. In *Latin-American Conference on Communications*, pages 1–6, Guatemala City, Guatemala, 2017. IEEE.
16. Z. Allybokus, N. Perrot, J. Leguay, L. Maggi, and E. Gourdin. Virtual function placement for service chaining with partial orders and anti-affinity rules. *Networks*, 71(2):97–106, 2018.
17. L. Bondan, T. Wauters, B. Volckaert, F. De Turck, and L. Z. Granville. Anomaly detection framework for SFC integrity in NFV environments. In *International Conference on Network Softwarization*, pages 1–5, Barcelona, Spain, 2017. IEEE.
18. G. Gardikis, I. Koutras, G. Mavroudis, S. Costicoglou, G. Xilouris, C. Sakkas, and A. Kourtis. An integrating framework for efficient NFV monitoring. In *Conference on Network Softwarization*, pages 1–5, Seoul, Korea, 2016. IEEE.

19. G. Gardikis, I. Koutras, G. Mavroudis, S. Costicoglou, G. Xilouris, C. Sakkas, and A. Kourtis. Machine learning-driven Scaling and Placement of Virtual Network Functions at the Network Edges. In *International Conference on Network Softwarization*, pages 414–422, Paris, France, 2019. IEEE.
20. T. D. Little. *Longitudinal Structural Equation Modeling (Methodology in the Social Sciences Series)*. Guilford, 2013.
21. J. Benesty, J. Chen, Y. Huang, and I. Cohen. Pearson Correlation Coefficient. In *Noise Reduction in Speech Processing*, pages 1–4. Springer, 2009.
22. S. Mehraghdam, M. Keller, and H. Karl. Specifying and placing chains of virtual network functions. In *International Conference on Cloud Networking*, pages 7–13, Anchorage, USA, 2014. IEEE.
23. S. Dräxler and H. Karl. Specification, composition, and placement of network services with flexible structures. *International Journal of Network Management*, 27(2):1963:1–20, 2017.
24. I. R. D. Kamgang, G. E. M. Zhioua, and N. Tabbane. A QoS-Based Service Chain Composition Approach for a Dynamic End-to-End Resource Allocation in NFV. In *International Conference on Advanced Information Networking and Applications*, pages 139–152, Caserta, Italy, 2020. Springer.
25. B. Németh, J. Czentye, G. Vaszkun, L. Csikor, and B. Sonkoly. Customizable Real-time Service Graph Mapping Algorithm in Carrier Grade Networks. In *Conference on Network Function Virtualization and Software Defined Network*, pages 28–30, San Francisco, USA, 2015. IEEE.
26. J. F. Riera, J. Batallé, J. Bonnet, M. Días, M. McGrath, G. Petralia, F. Liberati, A. Giuseppi, A. Pietrabissa, A. Ceselli, A. Petrini, M. Trubian, P. Papadimitrou, D. Dietrich, A. Ramos, J. Melián, G. Xilouris, A. Kourtis, T. Kourtis, and E. K. Markakis. TeNOR: Steps towards an orchestration platform for multi-PoP NFV deployment. In *Conference on Network Softwarization*, pages 243–250, Seoul, Korea, 2016. IEEE.
27. H. Baek, I. Jang, H. Ko, and S. Pack. Order dependency-aware service function placement in service function chaining. In *Information and Communication Technology Convergence, International Conference on*, pages 193–195, Jeju Island, Korea, 2017. IEEE.
28. M. C. Luizelli, D. Raz, and Y. Sa'ar. Optimizing NFV chain deployment through minimizing the cost of virtual switching. In *International Conference on Computer Communications*, pages 2150–2158, Honolulu, USA, 2018. IEEE.
29. D. Chemodanov, P. Calyam, and F. Esposito. A Near Optimal Reliable Composition Approach for Geo-Distributed Latency-Sensitive Service Chains. In *International Conference on Computer Communications*, pages 1792–1800, Paris, France, 2019. IEEE.
30. W. Bao, D. Yuan, B. B. Zhou, and A. Zomaya. Prune and plant: Efficient placement and parallelism of virtual network functions. *IEEE Transactions on Computers*, 69(6):800–811, 2020.
31. B. Spinnewyn, S. Latré, and J. F. Botero. Delay-constrained NFV Orchestration for Heterogeneous Cloud Networks. *Computer Networks*, 180:107420, 2020.
32. R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and S. Davy. Design and evaluation of algorithms for mapping and scheduling of virtual network functions. In *Conference on Network Softwarization*, pages 1–9, Seoul, Korea, 2015. IEEE.
33. S. G. Kulkarni, W. Zhang, J. Hwang, S. Rajagopalan, K. K. Ramakrishnan, T. Wood, M. Arumaithurai, and X. Fu. Nfvnice: Dynamic backpressure and scheduling for nfv service chains. In *Conference of the ACM Special Interest Group on Data Communication*, pages 71–84, Los Angeles, USA, 2017. ACM.
34. K. Zhang, B. He, J. Hu, Z. Wang, B. Hua, J. Meng, and L. Yang. G-net: Effective GPU Sharing in NFV Systems. In *Conference on Networked Systems Design and Implementation*, pages 187–200, Essaouira, Morocco, 2018. USENIX.
35. H. Moens and F. De Turck. Customizable Function Chains: Managing Service Chain Variability in Hybrid NFV Networks. *Transactions on Network and Service Management*, 13(4):711–724, 2016.
36. M. Wang, B. Cheng, S. Zhao, B. Li, W. Feng, and J. Chen. Availability-aware Service Chain Composition and Mapping in NFV-enabled Networks. In *International Conference on Web Services*, pages 107–115. IEEE, 2019.

37. L. Wang, Z. Lu, X. Wen, R. Knopp, and R. Gupta. Joint Optimization of Service Function Chaining and Resource Allocation in Network Function Virtualization. *IEEE Access*, 4:8084–8094, 2016.
38. V. Fulber-Garcia, M. C. Luizelli, C. R. P. d. Santos, and E. P. Duarte Jr. CUSCO: A Customizable Solution for NFV Composition. In *International Conference on Advanced Information Networking and Applications*, pages 204–216, Caserta, Italy, 2020. Springer.
39. A. Császár, W. John, M. Kind, C. Meirosu, G. Pongrácz, D. Staessens, A. Takács, and F. Westphal. Unifying Cloud and Carrier network: Eu fp7 project unify. In *International Conference on Utility and Cloud Computing*, pages 452–457, Dresden, Germany, 2013. IEEE.
40. S. Dräxler, H. Karl, M. Peuster, H. R. Kouchaksaraei, M. Bredel, J. Lessmann, T. Soenen, W. Tavernier, S. Mendel-Brin, and G. Xilouris. SONATA: Service programming and orchestration for virtualized software networks. In *International Conference on Communications*, pages 973–978, Paris, France, 2017. IEEE.
41. G. Canfora, L. Cerulo, and M. Di Penta. Identifying Changed Source Code Lines from Version Repositories. In *International Workshop on Mining Software Repositories*, pages 14–14, Minneapolis, USA, 2007. IEEE.
42. Y. Gong, S. Kumar, V. Verma, and S. Lazebnik. Angular quantization-based binary codes for fast similarity search. In *Advances in neural information processing systems*, pages 1196–1204, Lake Tahoe, USA, 2012. NeurIPS.
43. S. Kumar, M. Tufail, S. Majee, C. Captari, and S. Homma. Service Function Chaining: Use Cases in Data Centers. Technical report, Internet Engineering Task Force (IETF SFC WG), 2017.
44. T. N. Tavares, L. d. C. Marcuzzo, V. Fulber-Garcia, G. V. d. Souza, M. F. Franco, L. Bondan, A. E. Schaeffer-Filho, F. De Turck, L. Z. Granville, E. P. Duarte, and C. R. P. d. Santos. NIEP: NFV Infrastructure Emulation Platform. In *International Conference on Advanced Information Networking and Applications*, pages 173–180, Cracow, Poland, 2018. IEEE.

Vinicius Fulber-Garcia is a Ph.D. student in Computer Science at the Department of Informatics of the Federal University of Paraná (UFPR - Brazil) under the supervision of Prof. Ph.D. Elias Procópio Duarte Júnior. He holds a Computer Science degree from Federal University of Santa Maria (UFSM - Brazil) and a Master degree in Computer Science from UFSM Post-Graduate Program in Computer Science. His research interests include, but not limited to, network function virtualization and information theory.

Alexandre Huff is a Ph.D. candidate in Computer Science at the Federal University of Parana, holds a Master's Degree in Computer Science from the State University of Maringa (2010), and a Degree in Computer Technology from the Universidade Paranaense (2004). He is an Adjunct Professor of the Superior Magisterium at the Federal Technological University of Parana, Toledo, Brazil. His research topics include Distributed and Fault-Tolerant Systems, Network Function Virtualization, and Computer Networks.

Leonardo da Cruz Marcuzzo is an IT Technician at the Data Processing Center of Federal University of Santa Maria. He holds a Computer Science degree and a Master's degree from the Federal University of Santa Maria. He participated in the Federated Ecosystem for Distribution, Execution, and Life Cycle Management of VNFs (FENDE) project from the Brazilian National

Research Network. He has experience in Computer Science with particular interests in Network Function Virtualization and Software-defined Networks.

Marcelo Caggiani Luizelli holds a Ph.D. in Computer Science from the Federal University of Rio Grande do Sul (UFRGS). During his Ph.D., Luizelli was a visiting researcher at Nokia Bell Labs and the Technion University, both in Israel. Currently, he is an associate faculty at the Federal University of Pampa (Unipampa) in Brazil. His research activities have been focusing on emerging paradigms such as SDN, NFV, and programmable data planes.

Alberto Egon Schaeffer-Filho holds a Ph.D. in Computer Science (Imperial College London, 2009) and is Associate Professor at Federal University of Rio Grande do Sul (UFRGS), Brazil. Dr. Schaeffer-Filho is a CNPq-Brazil Research Fellow and his areas of expertise are network/service management, programmable networks, and security and resilience of networks. He has authored over 70 papers in leading peer-reviewed journals and conferences related to these topics, and also serves as TPC member for important conferences in these areas, including: CNSM (2020), NetSoft (2020), IEEE/IFIP NOMS (2020), CNSM (2019), NetSoft (2019) and IFIP/IEEE IM (2019). He served as co-chair for IEEE ICC 2018 CQRM Symposium, general co-chair for SBRC 2019, and TPC co-chair for IFIP/IEEE IM (2021). He is also a member of the IEEE.

Lisandro Zambenedetti Granville is Full Professor at the Institute of Informatics of the Federal University of Rio Grande do Sul. He served as TPC Co-Chair of IFIP/IEEE DSOM 2007 and IFIP/IEEE NOMS 2010, General Co-Chair of IFIP/IEEE CNSM 2014, TPC Co-Chair of IEEE NetSoft 2018, TPC Vice-Chair do IEEE ICC 2018, and future executive chair of IEEE GLOBECOM 2022. He was president of the Brazilian Computer Society (2005-2009), IEEE CNOM Chair (2013-2015), and IRTF's NMRG Co-Chair (2011-2018). His interests include network management, software-defined networking, and network functions virtualization.

Carlos Raniery Paula dos Santos is Adjunct Professor of Computer Science at the Department of Applied Computing of the Federal University of Santa Maria (UFSM), Brazil. He holds Ph.D. (2013) and M.Sc. (2008) degrees in Computer Science, both received from the Federal University of Rio Grande do Sul (UFRGS), where he was also Postdoctoral Research Fellow from October 2013 to September 2014. From May 2010 to April 2011 he was a visiting researcher at the IBM T.J. Watson Research Center - Hawthorne, where he developed projects on IT Service Management and Security Management. He has worked on the organization of several international conferences, is on the TPC of many network and service management events, including IM, NOMS and CNSM, and is reviewer of important journals in the area, such as IJNM, TNSM, COMNET, and COMMAG. He also serves as Secretary of the IEEE Technical Committee on Network Operations and Management (CNOM) since

2017. His current research interests focus on design and management of Future Networks and Technologies, including aspects such as network virtualization, quality of service management, network programmability, and security management.

Elias Procopio Duarte Junior is a Full Professor at Federal University of Parana, Curitiba, Brazil, where he is the leader of the Computer Networks and Distributed Systems Lab (LaRSis). His research interests include Computer Networks and Distributed Systems, their Dependability, Management, and Algorithms. He has published more than 250 peer-reviewer papers and has supervised more than 130 students both on the graduate and undergraduate levels. Prof. Duarte is currently Associate Editor of the Computing (Springer) journal and IEEE Transactions on Dependable and Secure Computing, and has served as chair of 25 conferences and workshops in his fields of interest. He received a Ph.D. degree in Computer Science from Tokyo Institute of Technology, Japan, 1997, M.Sc. degree in Telecommunications from the Polytechnical University of Madrid, Spain, 1991, and both BSc and MSc degrees in Computer Science from Federal University of Minas Gerais, Brazil, 1987 and 1991, respectively. He chaired the Special Interest Group on Fault Tolerant Computing of the Brazilian Computing Society (2005-2007); the Graduate Program in Computer Science of UFPR (2006-2008); and the Brazilian National Laboratory on Computer Networks (2012-2016). He is a member of the Brazilian Computing Society and a Senior Member of the IEEE.