A Multi-Objective Heuristic for the Optimization of Virtual Network Function Chain Placement

Stanislav Lange*, Alexej Grigorjew*, Thomas Zinner*, Phuoc Tran-Gia*, Michael Jarschel§

*University of Würzburg, Institute of Computer Science, Chair of Communication Networks, Würzburg, Germany
{stanislav.lange, alexej.grigorjew, zinner, trangia}@informatik.uni-wuerzburg.de
§Nokia Bell Labs, Munich, Germany
michael.jarschel@nokia-bell-labs.com

***Abstract*—The Network Functions Virtualization (NFV) paradigm offers network operators benefits in terms of cost efficiency, vendor independence, as well as flexibility and scalability. However, in order to profit most from these features, new challenges in the area of management and orchestration of the virtual network functions (VNFs) need to be addressed.**

In particular, this work deals with the VNF chain placement problem (VNFCP). For a given network situation, the task consists of determining the number, location, and assignment of VNF instances and the routing of demands. At the same time, several metrics like CPU utilization and the delay of individual flows need to be taken into account. For applicability in networks with dynamically changing conditions, algorithms need to explore the solution space of this NP-hard problem in a timely manner.

The contribution of this work is threefold: firstly, we design MO-VNFCP, a multi-objective heuristic for the VNFCP. Secondly, we investigate the convergence behavior of the algorithm in a case study. Finally, we provide a comparison between the proposed algorithm and an alternative approach from literature.

***Index Terms*—NFV, VNF Chain Placement, Multi-Objective Optimization.**

I. Introduction

In today's networks, a wide range of networking tasks, like load balancing and firewalling, are performed by specialized hardware middleboxes. Although they provide high performance, there are also downsides with respect to capital expenditures, flexibility, scalability, and dependence on the vendor of the appliance. These limitations are addressed by the Network Functions Virtualization (NFV) paradigm [1]. By leveraging virtualization techniques, middleboxes are replaced with software instances that run in a virtualized environment on commodity-of-the-shelf hardware. In addition to a lower initial capital cost, flexibility is improved due to the possibility of relocating virtual network functions (VNFs) between points of presence. Furthermore, dynamic scaling can be performed by adapting the number of active VNF instances.

In order to reap these benefits, mechanisms for the management and orchestration of VNFs need to be developed. In this context, several questions arise:

1) How many VNF instances are required?
2) Where should they be deployed?
3) Which demands are handled by which instance?
4) Which path should a demand take through the network?

Combined with requirements like link capacity and ultra-low latency that have to be met, these questions define the virtual network function chain placement (VNFCP) problem that is addressed in this work. Usually, operators need to optimize multiple, possibly competing goal functions. Existing approaches deal with this by aggregating multiple objectives into a single, weighted sum, or only focus on a specific subset. However, this aggregation might conceal trade-offs between the objectives. Furthermore, the operators' preferences might not be known beforehand and usually also depend on the available alternatives. Hence, we approach this problem in a multi-objective fashion. If necessary, the performance of each placement can still be aggregated into a single score while taking into account the observed characteristics of the objective functions, enabling automated decision making [2].

As the VNFCP problem is NP-hard, exact solutions like integer linear programs (ILPs) have large requirements on computational resources and time. Thus, they are usually unfeasible for realistic problem scales. On the other hand, many heuristics from literature are either limited by a local view on the problem due to time constraints, or have high memory demands. In contrast to these, our algorithm performs a global optimization by considering all demands simultaneously.

Our approach is capable of generating a set of Pareto-optimal solutions that represent different trade-offs between competing objectives. Hence, it is applicable during the planning phase of a network in which human decision makers are involved. Furthermore, MO-VNFCP can be used for regular re-optimization during the operational phase, correcting possible deviations caused by faster online approaches, which often neglect downscaling when demands change. We focus on the offline part of the optimization for the remainder of this work.

The contribution of this work is comprised as follows. In Section II, we provide a formal problem statement for the VNFCP. Section III describes our multi-objective heuristic MO-VNFCP, whose source code is available online[1]. Our evaluation methodology and scenarios are introduced in Section IV, and the evaluation results are presented in Section V. Finally, related work is discussed in Section VI, and Section VII concludes the paper.

[1] https://github.com/lsinfo3/mo-vnfcp

978-0-9883-0453-6/17 $31.00 © 2017 ITC
DOI 10.1109/ITC.2017.24

II. Virtual Network Functions Chain Placement

In this section, an overview of the VNFCP problem is given. The input and output variables are formalized and extended by various constraints regarding link bandwidth, delay requirements, and node resource capacities. Finally, exemplary optimization objectives are introduced. For reference, Table I summarizes all used symbols in these definitions.

Notation. In the following, the term $a_{\{x,y,z\}}$ represents the *enumeration* a_x, a_y, a_z. In contrast, $a_{x,y,z}$ refers to the *composition*, i.e., $((a_x)_y)_z$ or $a_{x_{y_z}}$, respectively. For example, $r_{\text{inst},i}$ is defined as the i-th element of r_{inst}.

A. Input Variables

(1) The network is represented by an undirected graph $G = (V, E)$. Nodes $v \in V$ represent network elements that can be part of a telco cloud point of presence and therefore may have access to computational resources ($v_{\text{cpu}}, v_{\text{ram}}, v_{\text{hdd}} \in \mathbb{R}$). Network links are modeled by edges $e \in E$ with their respective bandwidth $e_{\text{bw}} \in \mathbb{R}$ and delay $e_{\text{d}} \in \mathbb{R}$.

(2) The available network function types t are given in a set T. Each function $t \in T$ has a specific amount of required resources ($t_{\text{cpu}}, t_{\text{ram}}, t_{\text{hdd}} \in \mathbb{R}$) and possesses similar attributes as links ($t_{\text{bw}}, t_{\text{d}} \in \mathbb{R}$).

(3) The set R of traffic requests (or traffic demands) consists of tuples $r = (r_{\text{src}}, r_{\text{dst}}, r_{\text{bw}}, r_{\text{d}}, r_{\text{c}})$. The *source* and *destination* nodes of the respective demands are given by $r_{\text{src}}, r_{\text{dst}} \in V$, while $r_{\text{bw}} \in \mathbb{R}$ represents the minimum required bandwidth, and $r_{\text{d}} \in \mathbb{R}$ is the maximum tolerated latency of the request r. The requested network function *chain* is represented by a sequence of function types $t_i \in T$: $r_{\text{c}} = (t_1, \ldots, t_{|r_{\text{c}}|})$. These functions need to be applied to all respective flows of this traffic request exactly in the given order.

For some use cases, a subset of the above requirements may be sufficient. For example, delay insensitive flows can be incorporated by lifting the respective restriction, e.g., $r_{\text{d}} = \infty$.

B. Output Variables

The resulting placements are represented by the following variables in the solution space:

(1) Each placed instance $z \in \mathcal{I}$ has a specific type $z_{\text{type}} \in T$ and a location $z_{\text{node}} \in V$. Additionally, the set of all traffic requests r that utilize the instance are given by $z_{\text{reqs}} = \{r \in R \mid z \in r_{\text{inst}}\}$.

(2) For each request $r \in R$, both the route r_{route} and the utilized VNF instances r_{inst} are given. The route through the network is represented by a node sequence $v_i \in V$: $r_{\text{route}} = (v_1, \ldots, v_{r_\ell})$. The used instances r_{inst} are modeled by a sequence of the same length r_ℓ. It contains the used VNF instances for each visited node $v_i \in r_{\text{route}}$: $r_{\text{inst}} = (z_1, \ldots, z_{r_\ell})$, where $z_i \in \mathcal{I} \cup \{\emptyset\}$ for all $1 \leq i \leq r_\ell$. If $z_i = \emptyset$, no network function is applied to the traffic on node v_i.

(3) For each node $v \in V$, all respective VNF instances placed on v are given by $v_{\text{inst}} = \{z \in \mathcal{I} \mid z_{\text{node}} = v\}$.

(4) For each link $e \in E$, the multi-set of all traffic requests $r \in R$ whose route includes $e = \{v_1, v_2\}$ is given by $e_{\text{reqs}} = \{r \in R \mid (v_1, v_2) \subseteq r_{\text{route}} \vee (v_2, v_1) \subseteq r_{\text{route}}\}$. Note that e_{reqs} may include the same request multiple times, e.g., if the link is used in both directions.

Here, v_{inst}, e_{reqs}, and z_{reqs} can be calculated from the values of $\mathcal{I}$, z_{node}, r_{route}, and r_{inst}. They are merely specified for convenience. Only the latter are subject of the optimization.

TABLE I
Overview of used symbols.

	Symbol	Description
	v, V	Set of network graph nodes $v \in V$.
	e, E	Set of network graph edges $e \in E \subseteq \binom{V}{2}$.
	G	Network graph $G = (V, E)$.
	$v_{\{\text{cpu, ram, hdd}\}}$	Available resources $v_{\{\text{cpu, ram, hdd}\}} \in \mathbb{R}$ on node $v \in V$.
$(*)$	v_{inst}	Set of placed VNF instances $z \in \mathcal{I}$ on node $v \in V$.
	$e_{\text{bw}}, e_{\text{d}}$	Available bandwidth $e_{\text{bw}} \in \mathbb{R}$ and imposed delay $e_{\text{d}} \in \mathbb{R}$ on link $e \in E$.
$(*)$	e_{reqs}	Multiset of all requests $r \in R$ that traverse the link $e \in E$.
	t, T	Set of available network function types $t \in T$.
	$t_{\{\text{cpu, ram, hdd}\}}$	Consumed resources $t_{\{\text{cpu, ram, hdd}\}} \in \mathbb{R}$ by type $t \in T$.
	$t_{\text{bw}}, t_{\text{d}}$	Available bandwidth capacity t_{bw} and imposed delay t_{d} of type $t \in T$.
	r, R	Set of traffic requests $r = (r_{\text{src}}, r_{\text{dst}}, r_{\text{bw}}, r_{\text{d}}, r_{\text{c}}) \in R$.
	$r_{\text{src}}, r_{\text{dst}}$	The traffic request's *source* and *destination* nodes $r_{\text{src}}, r_{\text{dst}} \in V$.
	$r_{\text{bw}}, r_{\text{d}}$	Requested bandwidth $r_{\text{bw}} \in \mathbb{R}$ and maximum accepted delay $r_{\text{d}} \in \mathbb{R}$ of traffic request $r \in R$.
	r_{c}	Requested service chain: sequence $(t_1, t_2, \ldots)$ with $t_i \in T$.
$(*)$	r_{route}	Assigned path through the network: $r_{\text{route}} = (v_1, v_2, \ldots, v_{r_\ell})$, $v_i \in V$.
$(*)$	r_ℓ	Length of the assigned path.
$(*)$	r_{inst}	Applied VNFs $(z_1, z_2, \ldots, z_{r_\ell})$ with $z_i \in \mathcal{I} \cup \{\emptyset\}$.
$(*)$	r_{seq}	Locations $r_{\text{seq}} = (v_1, \ldots, v_{|r_{\text{c}}|})$, $v_i \in V$ of all applied functions $t \in r_{\text{c}}$.
$(*)$	$z, \mathcal{I}$	Set of placed VNF instances $z \in \mathcal{I}$.
$(*)$	$z_{\text{type}}, z_{\text{node}}$	The type $z_{\text{type}} \in T$ and node $z_{\text{node}} \in V$ of an instance $z \in \mathcal{I}$.
$(*)$	z_{reqs}	Set of all requests $r \in R$ that use the instance $z \in \mathcal{I}$.

Rows labeled with $(*)$ indicate output variables.

C. Constraints

With the above model for input and output variables, the following constraints are considered for the VNFCP problem in this work. Let $P = \{\text{cpu}, \text{ram}, \text{hdd}\}$ be the set of considered node resource types.

$$\forall r \in R, \forall (v, w) \subseteq r_{\text{route}} : \{v, w\} \in E \vee v = w \quad (1)$$

$$\forall r \in R : r_{\text{route},1} = r_{\text{src}} \wedge r_{\text{route},r_\ell} = r_{\text{dst}} \quad (2)$$

$$\forall r \in R, \forall i \in \{1, ..., r_\ell\} : r_{\text{inst},i} = \emptyset \vee r_{\text{inst},i,\text{node}} = r_{\text{route},i} \quad (3)$$

$$\forall r \in R : r_{\text{c}} = r_{\text{inst}} \backslash \{\emptyset\} \quad (4)$$

$$\forall t \in T : |\{z \in \mathcal{I} \mid z_{\text{type}} = t\}| \leq t_{\text{licenses}} \quad (5)$$

$$\forall v \in V, \forall p \in P : \sum_{z \in v_{\text{inst}}} z_p \leq v_p \quad (6)$$

$$\forall e \in E : \sum_{r \in e_{\text{reqs}}} r_{\text{bw}} \leq e_{\text{bw}} \quad (7)$$

$$\forall z \in \mathcal{I} : \sum_{r \in z_{\text{reqs}}} r_{\text{bw}} \leq z_{\text{type,bw}} \quad (8)$$

$$\forall r \in R : \sum_{(v,w) \subseteq r_{\text{route}}} \{v, w\}_{\text{d}} + \sum_{z \in r_{\text{inst}}} z_{\text{type,d}} \leq r_{\text{d}} \quad (9)$$

Equation 1 ensures that all used links actually exist in the network. With Equation 2, the source and destination of each request are always the first and last node on its assigned route. Equations 3–4 establish consistency between r_{route}, r_{inst}, and r_{c}. Equation 5 respects the maximum allowed numbers of instances. Physical resource constraints are covered by equations 6–8. Finally, Equation 9 tackles service level agreements and ensures that the requested maximum delays are not exceeded.

While Equations 1–4 are mandatory for reasonable placements, constraints 5–9 represent resource limitations. If they are not satisfied, the respective solution is called *unfeasible*.

D. Objectives

Typical objectives of the VNFCP problem involve the minimization of costs and delays. The former can be deduced further, e.g., into the number of active VNF instances and the amount of used physical resources in the network. The latter can, for example, either be expressed by an overall sum of all requests' delays in a given placement, or by considering their relation to the shortest possible route for a fair comparison.

In this work, the objectives $F = (f_{\text{delay}}, f_{\text{hops}}, f_{\text{inst}}, f_{\text{cpu}})$ are minimized simultaneously, which are defined as follows.

$$f_{\text{delay}} := \sum_{r \in R} \left(\sum_{(v,w) \subseteq r_{\text{route}}} \{v, w\}_{\text{d}} + \sum_{t \in r_{\text{c}}} t_{\text{d}} \right) \tag{10}$$

$$f_{\text{hops}} := \sum_{r \in R} |\{(v, w) \subseteq r_{\text{route}} \mid v \neq w\}| \tag{11}$$

$$f_{\text{inst}} := |\mathcal{I}| \tag{12}$$

$$f_{\text{cpu}} := \sum_{z \in \mathcal{I}} z_{\text{type,cpu}} \tag{13}$$

Thereby, the overall delay of all demands is represented by f_{delay}, and their total number of hops by f_{hops}. Objective f_{inst} attempts to minimize the total number of placed instances, regardless of their resource demands, while f_{cpu} is specifically focused on used CPU resources. This choice of objectives matches the aims of the reference algorithm from literature and enables their performance comparison in Section V.

E. Pareto-Optimal Solution Set

In the multi-objective context, the output usually consists of multiple solutions. The solution set contains all *Pareto-optimal* placements that the respective algorithm encounters during the optimization, i.e., all such placements that cannot be improved in any way without deteriorating another objective at the same time. These Pareto-optimal placements are called *Pareto frontier* in the objective space ($\mathbb{R}^4$).

III. Heuristic Placement

Our algorithm is based on a simplified model to obtain a Pareto frontier approximation. In this section, we introduce this model, as well as the heuristic that explores the solution space by means of a neighborhood search.

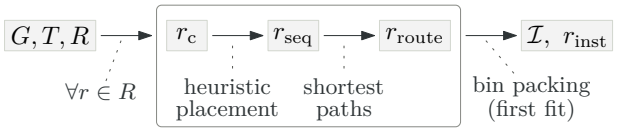


Fig. 1. Overview of the placement generation strategy.

A. Simplified Model

The simplified model deals with the different layers of VNF placements individually, i.e., instance location, assignment of demands to instances, and routing of demands. Figure 1 summarizes the solution generation strategy with this intermediate representation. In that context, a placement s is represented by a set of node sequences $s = \{r_{\text{seq}} \mid r \in R\}$. For each demand $r \in R$, the locations of the requested VNFs $r_{\text{c}} = (t_1, \ldots, t_{|r_{\text{c}}|}) \in T^{|r_{\text{c}}|}$ are stored in $r_{\text{seq}} = (v_1, \ldots, v_{|r_{\text{c}}|}) \in V^{|r_{\text{c}}|}$. Thereby, the actual route through the network r_{route} is acquired by applying a shortest path algorithm for each pair (r_{src}, v_1), (v_i, v_{i+1}) $\forall i \in \{1, \ldots, |r_{\text{c}}| - 1\}$, and $(v_{|r_{\text{c}}|}, r_{\text{dst}})$.

The number of required VNF instances can be derived from r_{seq} by solving the bin packing problem. For each node $v \in V$ and each network function type $t \in T$, the objects $\{r_{\text{bw}} \mid r \in R, \exists i : r_{\text{seq},i} = v \wedge r_{\text{c},i} = t\}$ need to be placed in bins of size t_{bw}. Because the bin packing problem itself is NP-complete [3], a first-fit heuristic is applied in this work that produces results within the 1.7-fold of the optimal solution [4].

B. Heuristic Algorithm: MO-VNFCP

In order to obtain optimized placements, we apply a modified variant of Pareto simulated annealing [5], a multi-objective extension of the simulated annealing algorithm that has achieved promising results in the context of other placement problems [6]. Algorithm 1 provides an overview of its main steps. In addition to the three problem specific input variables (G, T, R), the algorithm requires five additional parameters. The initial temperature $\tau_0 \in \mathbb{R}^+$, the minimum temperature $\tau_{\text{min}} \in \mathbb{R}^+$, and the cooling factor $\rho \in (0, 1)$ determine the number of temperature levels for the outmost while-loop. For $0 < \tau_{\text{min}} < \tau_0$, there are $\lceil \log_\rho(\tau_{\text{min}}/\tau_0) \rceil$ temperature levels in total. The number of inner for-loop iterations for the same temperature level is defined by $m \in \mathbb{N}$. Finally, the parameter $|S| \in \mathbb{N}$ determines how many individual placements are explored simultaneously by the algorithm.

At first, an initial solution set S is generated with the strategies outlined below. This set is then improved by iteratively modifying small bits of existing solutions and evaluating these new *neighbor* solutions. Based on their performance, new solutions are either accepted into S, effectively replacing their parent, or dropped and the algorithm continues at the existing placement, as defined by a dedicated acceptance probability.

1) Initial Solutions: The initial solution set should not only be generated with regard to covering a broad area in the solution space, but ideally also contain both feasible and partly optimized placements. In this work, the following four placement strategies were used.

Algorithm 1: mo–vnfcp(G, T, R)

Additional parameters: $\tau_0, \tau_{\min}, \rho, m, |S|$

```
S ← generateInitialSolutions(G, T, R)
M ← initializeParetoFrontier(S)
τ ← τ_0
while τ > τ_min do
    for i ∈ {1, ..., m} do
        Y ← {generateNeighbor(s, τ) | s ∈ S}
        M ← updateParetoFrontier(M, Y)
        S ← accept y ∈ Y with probability p(S, Y, τ)
    τ ← τ · ρ
return M
```

Random: The simplest approach is to assign all VNF locations randomly. For each demand $r \in R$ and each requested VNF $t_i \in r_c$, the respective location $v_i \in V$ is picked at random from all available nodes.

Short Pre-Optimization: The above random placements can be pre-optimized by applying a shorter MO-VNFCP execution before attempting the actual optimization. The initial solutions then consist of elements from the short execution's Pareto frontier.

Minimize Delays: This heuristic assigns all demands to their shortest path and applies all network functions on this path.

Minimize Number of Instances: A minimum number of VNF instances is placed in the network to cover all requests. Their locations are based on the betweenness centrality, i.e., the instances are placed on those nodes with the highest number of shortest paths from unserved requests traversing them. The routes are, in turn, composed of the shortest paths through all requested VNF instances. This is a greedy assignment, i.e., the first served requests usually have more instances to choose from.

2) Neighbor Generation: In general, new neighbor placements are generated by randomly modifying one (or multiple) node sequences r_{seq}. In an effort to support the convergence w.r.t. the considered objectives, several improvements were incorporated in the selection process. Figure 2 provides a high level overview of the neighbor generation control flow.

As the distribution of instances is derived indirectly from r_{seq}, the selection of new sequences is extended by dedicated probability parameters. With the probability $p_{\text{removeVNF}}$, the algorithm reassigns all r_{seq} for each $r \in z_{\text{reqs}}$ of a selected instance $z \in \mathcal{I}$, as opposed to only reassigning a single demand. This effectively removes the instance, as all of its assigned requests are migrated. Similarly, during the selection of nodes $v_i \in r_{\text{seq}}$, only nodes with compatible VNF instances are considered initially. With a probability $p_{\text{createVNF}}$ (or if existing instances are congested), all nodes with sufficient resources are taken into account instead. Both probabilities can be configured by the operator and may depend on the current temperature level in order to adapt towards the end of the optimization. This enables better control of the instance distribution and provides some flexibility to deal with varying problem inputs.

To handle the large decision space, weighted randomness is applied during neighbor generation. Both the selection of the reassigned demands $r \in R$ and the choice of their new locations r_{seq} are performed w.r.t. their impact on the objectives. Figure 3 shows an example of one such r_{seq} selection, inspired by [7]. Figure 3a displays the current VNF distribution in the network without the respective unassigned demand r (Figure 3b). A multistage graph with node and link weights is constructed with one stage for each requested function $t \in r_c$, as shown in Figure 3c. Each node in a stage represents an available location for the corresponding function. A node's weight is determined by the cumulative delay on the shortest path from a to the node. Each node v_i in stage i is connected to each node v_{i+1} in stage $i+1$, and the links (v_i, v_{i+1}) are weighted by the delay of the shortest path from v_i to v_{i+1} in the real network. Then, a weight-based selection is performed from r_{dst} to r_{src}, as indicated by Figure 3d. For each choice v, the probability corresponds to the inverse of its respective weights, including the node's weight (delay $a \to v$) and all links' weights from v to b. In this example, the node with the highest probability is always chosen.

During the above procedure, new VNFs might be instantiated. In this case, requests that used other instances before are rerouted towards the new instance if it benefits their delays and number of hops. As far as the considered objectives are concerned, this can be done without deteriorating other aspects of the placement.

3) Acceptance Probability: In order to prevent getting stuck in local optima, the algorithm accepts worse placements with a certain probability. However, depending on the problem instance, the quality of generated neighbor placements may vary significantly. Hence, the acceptance probability dynamically adapts to the situation and supports convergence towards global optima. In addition, the current temperature level is taken into account in order to reduce the acceptance rate of worse solutions towards the end of the optimization.

When comparing two individual solutions a, b in the multi-objective context, the following cases are relevant here:

- b is *dominated* by a , i.e., a is not worse in any objective, and better in at least one of them,
- a is *dominated* by b,
- a and b are *incomparable* (or similar), i.e., neither is better than the other w.r.t. all objectives.

Further details regarding the solutions' relationship can be obtained from literature [8]–[11].

Based on that, let $n_{\{\text{better,worse,incomp}\}}$ be the number of better, worse, and incomparable neighbors, compared to their parent, during the last temperature level with m neighbors in total. Then, the acceptance probabilities $p_{\text{accept},\{\text{better,worse,incomp}\}}$ are defined as follows.

$$p_{\text{accept,better}} := 1 \tag{14}$$

$$p_{\text{accept,worse}} := \frac{\tau}{\tau_0} \cdot c_{\text{worse}} \cdot \frac{n_{\text{better}}}{m} \tag{15}$$

$$p_{\text{accept,incomp}} := \frac{\tau}{\tau_0} \cdot c_{\text{incomp}} \cdot \frac{n_{\text{better}}}{n_{\text{incomp}}} \tag{16}$$

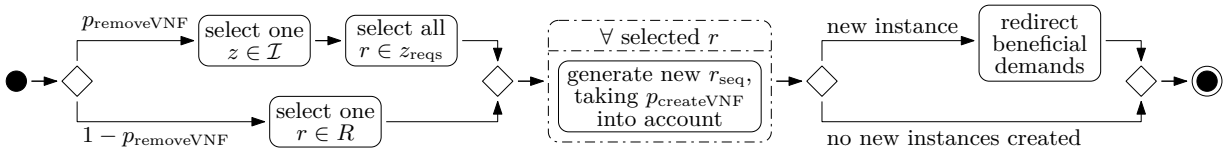
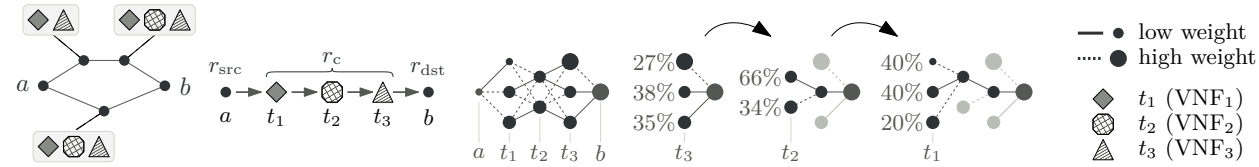


Fig. 2. Overview of the neighbor generation process.



(a) Network graph and current VNF locations. (b) Traffic demand r. (c) Delay-weighted multistage graph. (d) Individual choice probabilities for each $t \in r_{\text{c}}$.

Fig. 3. Exemplary weighted r_{seq} selection during neighbor generation.

Here, $c_{\{\text{worse,incomp}\}}$ are additional parameters that allow problem dependent scaling, if necessary, e.g., $c_{\text{worse}} = 1.1$ and $c_{\text{incomp}} = 1.2$.

IV. Performance Assessment Methodology

This section describes the methodology used to compare the results of multi-objective optimizers. In particular, three *quality indicators*, namely the *hypervolume*, *epsilon*, and *weighted sum* indicators, are used to assess the performance of multiple points in a multi-dimensional solution set X by means of a single measure $I_\bullet(X)$. Each of them represents different aspects of the solutions' quality. Finally, problem instances and VNF characteristics used in the evaluation are described.

A. Hypervolume Indicator

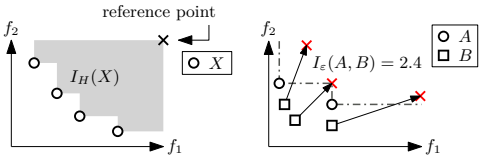
The idea of the hypervolume indicator $I_H(X)$ [11], [12] is to compute the area of solutions in the objective space that are dominated by a given Pareto frontier approximation X. Because this volume is infinite for a minimization problem, a reference point x is used that bounds the objective space, as illustrated in Figure 4a. Hereby, higher values of $I_H(X)$ indicate a better performance w.r.t. this indicator.

To cope with differences in the objectives' ranges, the points are normalized. The reference point is then defined as $x := (1, \ldots, 1)^T$. Therefor, let n be the number of objectives, and $F_{\text{max}} = (f_{1,\text{max}}, \ldots, f_{n,\text{max}})$ the biggest observed objective values across all solution sets that are compared with each other. Then, each point is divided by $1.5 \cdot F_{\text{max}}$, which also ensures some distance to the reference point.

Due to the complex computation of the hypervolume in $\mathbb{R}^n$, a Monte Carlo approximation is used in this work. Thereby, $m = 100\,000$ uniformly distributed random points are generated in $[0, 1]^n$. Let m_{dom} be the number of such points that are dominated by the investigated solution set X. Then, the hypervolume of X can be estimated by $I_H(X) = m_{\text{dom}}/m$.

B. Epsilon Indicator

The epsilon indicator $I_\varepsilon(A, B)$ [10], [11] can be used to compare two solution sets A, B directly. The idea is to measure



(a) Hypervolume indicator. (b) Epsilon indicator.

Fig. 4. Illustration of two quality indicators with two objectives.

how far B has to be stretched to be dominated by A.

$$I_\varepsilon(A, B) := \inf\{\varepsilon \in \mathbb{R} \mid \forall b \in B\ \exists a \in A : F(a) \leq \varepsilon F(b)\} \quad (17)$$

Figure 4b illustrates the concept in $\mathbb{R}^2$. The smallest factor ε to push all $b \in B$ into the dashed, dominated area of A is $\varepsilon = 2.4$. In accordance to [10], the indicator value is calculated as follows.

$$I_\varepsilon(A, B) = \max_{b \in B} \left\{ \min_{a \in A} \left\{ \max_{1 \leq i \leq n} \left\{ \frac{f_i(a)}{f_i(b)} \right\} \right\} \right\} \quad (18)$$

A unary version of the indicator can be acquired by defining a reference solution set E.

$$I^1_\varepsilon(X) := I_\varepsilon(X, E) \quad (19)$$

This reference can be any Pareto frontier approximation, the actual Pareto frontier, or an arbitrary set of points. In this work, the grand Pareto frontier of all observed approximations for the same problem instance is used. This indicator is to be minimized, smaller values indicate better performance.

C. Weighted Sum Indicator

The weighted sum indicator $I_{WS}(X)$ aims to express how close the solution set X is to a theoretical optimal value for each considered objective from Section II-D, without knowing the actual Pareto frontier.

Let $r_{\{\text{latency,hops}\}}(x)$ be the perceived latency and hop count of demand $r \in R$ for the placement $x \in X$. Further, let $r_{\{\text{shortestLat,shortestHops}\}}$ be the theoretically smallest possible latency and number of hops for the given demand r, involving at least one node with sufficient resources for VNFs on its path. Let R_t be the subset of all traffic demands $r \in R$ that

TABLE II
SUMMARY OF THE CONSIDERED PROBLEM PARAMETERS.

VNF Type t	$\mathbf{t_{cpu}}$	$\mathbf{t_d}$ **[µs]**	$\mathbf{t_{bw}}$ **[Mbps]**
Firewall	4	45	900
Proxy	4	40	900
IDS	8	1	600
NAT	2	10	900
Network Parameter	**Internet2**	**Geant**.{1,2}	**Germany**.{1,2}
Nodes, Links	12, 15	22, 36	50, 88
CPU Locations	7	$\{22, 6\}$	$\{50, 5\}$
Requests	132	462	662
Req. Bandw. Variability	varying	varying	homogeneous
Link Delay Variability	varying	varying	homogeneous
Mean Req. Bandw.	45 Mbps	6.5 Mbps	3.6 Mbps
Mean Rel. Req. Delay*	3.2	3.5	$\{35, 3.5\}$

* requested maximum delay in relation to the shortest path

request the function $t \in T$: $R_t := \{r \in R \mid t \in r_c\}$. Then, the measures *mean delay index* f_{MDI}, *mean hops index* f_{MHI}, *median inverse instance load* f_{MedIL}, and *CPU index* f_{CpuI} are defined as follows.

$$f_{\text{MDI}}(x) := \frac{1}{|R|} \sum_{r \in R} \frac{r_{\text{latency}}(x)}{r_{\text{shortestLat}}} \tag{20}$$

$$f_{\text{MHI}}(x) := \frac{1}{|R|} \sum_{r \in R} \frac{r_{\text{hops}}(x)}{r_{\text{shortestHops}}} \tag{21}$$

$$f_{\text{MedIL}}(x) := \texttt{median} \left\{ \frac{z_{\text{type,bw}}}{\sum_{r \in z_{\text{reqs}}} r_{\text{bw}}} \;\middle|\; z \in \mathcal{I} \right\} \tag{22}$$

$$f_{\text{CpuI}}(x) := \frac{\sum_{z \in \mathcal{I}} z_{\text{type,cpu}}}{\sum_{t \in T} \left\lceil \frac{\sum_{r \in R_t} r_{\text{bw}}}{t_{\text{bw}}} \right\rceil \cdot t_{\text{cpu}}} \tag{23}$$

In each case, higher values indicate worse performance, while a value of 1 represents an optimal performance with respect to the considered objective, which may not always be attainable for every problem instance though.

The weighted sum indicator $I_{WS}(X)$ is then defined by:

$$I_{WS}(X) := \min_{x \in X} \left\{ \frac{f_{\text{MDI}}(x) + f_{\text{MHI}}(x) + f_{\text{MedIL}}(x) + f_{\text{CpuI}}(x)}{4} \right\} \tag{24}$$

D. Problem Instances & Parameters

The evaluation is performed on five different problem instances based on three topologies. While network topologies and demand profiles are available in literature, VNF characteristics and function chain requirements are hard to obtain, hence this work uses mostly artificial data for the latter.

The first part of Table II outlines the different characteristics of considered VNFs $t \in T$ which are based on [7], [13], as well as the pfSense hardware sizing guide[2]. Most traffic demands are generated to request 0–4 functions, uniformly distributed, in random order.

Similarly, the second part of Table II summarizes the five problem instances used in the evaluation. The *Internet2* graph, along with its resource demands, was taken from literature [7], while *Geant* and *Germany* were derived from the online library SNDLib[3]. For the latter two topologies, the resource distribution is generated artificially, such that both networks are evaluated with two resource profiles each, featuring high and low numbers of CPU locations. Thereby, *Germany.1* presents an extreme case where all 50 nodes are available for VNF placement and the requested maximum delays are exceptionally high, leaving a wide range of feasible choices for the placement algorithms.

The algorithm's parameters are tweaked individually for each problem instance. Their choice is based on qualitative differences between problem instances, but also backed by a previous parameter study. For example, the optimization of *Germany.1* is performed with the *minimize-number-of-instances* initial solution strategy and $p_{\text{createVNF}} = 0$, while for *Germany.2*, *minimize-delays* and a linear definition of $p_{\text{createVNF}}$ are used.

The general applicability of the algorithm is bounded by the quality / time ratio of its iterative optimization process and the initial heuristics. This effectiveness is strongly influenced by problem difficulty and parameters. To this extent, feasible solutions were acquired for problems with up to 150 available CPU locations, chains with up to 15 functions, and more than 1,000 requests within less than an hour of time on common desktop hardware. The details of these parameter studies are out of the scope of this paper and are therefore omitted.

V. RESULTS

Setup. All experiments are performed on a virtual machine with 24 CPU[4] cores and 4 GB of RAM. The algorithm is implemented to make use of multiple cores by using a separate thread for each simultaneously optimized placement $s \in S$. If not stated otherwise, only feasible solutions are considered in the evaluation process, i.e., only such solutions that satisfy all constraints listed in Section II-C. For the sake of simplicity, only CPU resources are considered.

A. Case Study: Solution Quality Development

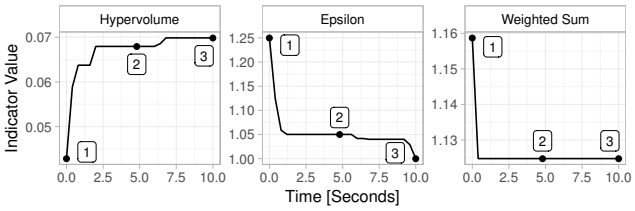
To provide an idea of the optimization process and the expressiveness of the different indicators, Figure 5a displays their development over 10 s for an exemplary run on the *Internet2* topology. Recall that the hypervolume is to be maximized, while the epsilon and weighted sum indicators present better quality with smaller values. Thereby, several characteristics can be identified.

The biggest improvements of the solution quality occur in the first 2 s of the optimization. The probability to generate better neighbor placements is comparably high during the initial phase, and the solution space is explored vastly, for example due to loosened acceptance probabilities. At the same time, some indicators suggest another significant enhancement towards the end of the optimization. These represent a focused improvement of previously prepared placements with lower (or zero) acceptance probabilities for worse solutions and less variation among the VNF instance locations.

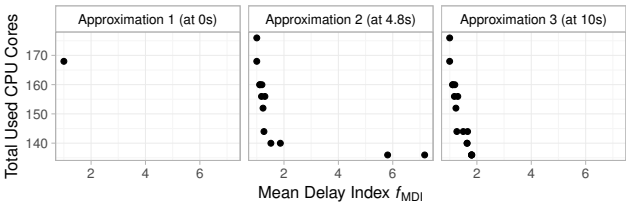
[2]https://www.pfsense.org/hardware/#sizing

[3]http://sndlib.zib.de/

[4]Hosted on OpenStack, host: 2x Intel Xeon E5-2620 v2, HT enabled



(a) Developement of the individual indicator values.



(b) Snapshots of the Pareto frontier approximation at different times.

Fig. 5. Exemplary optimization process for the *Internet2* topology over time.

When comparing the individual improvements of the indicators, each of them represents a different aspect of quality in general. The epsilon indicator is the only one to show an improvement in the last seconds, while the weighted sum indicator displays a rather steady quality throughout the entire process. This is partly caused by the aggregation during their computation: the value of the weighted sum indicator is effectively based on only one solution of the Pareto frontier approximation, namely that with the lowest value. As long as this value is not outperformed, changes in the other solutions are not reflected here. This characteristic is useful to assess the potential for application in automatic decision making, as only one solution can be chosen in the end. Nevertheless, the different behavior of the indicators offers more expressiveness when used together, as pointed out in [11].

Note that the hypervolume and epsilon indicators are mainly suited to compare relative performance in the context of a fixed problem instance, whereas the low value of $I_{WS}(X) \approx 1.125$ indicates a good quality w.r.t. theoretical optima.

In addition to the indicator values, Figure 5b shows three snapshots of the Pareto frontier approximations at the execution times 0 s, 4.8 s, and 10 s. Note that not all four objectives are displayed, only the mean delay index (cf. Equation 20) and used CPU resources are presented to retain a clear overview.

In the first snapshot, all initially generated placements in S follow the *minimize-delay* strategy, hence the set of Pareto optimal solutions only contains the placement with the lowest resource usage for that delay value. At 4.8 s, many new trade-offs between CPU usage and delay are presented. However, the placements with the lowest CPU utilization suffer from high latencies. At the end of the optimization, at 10 s, they become dominated by a feasible low-latency solution, which causes the final improvement for the epsilon indicator. A decision maker can now choose from placements with roughly $135-175$ used CPU cores f_{cpu} and $1.5-2.0$ mean delay index f_{MDI}.

B. Performance Comparison

This section covers a performance comparison of the MO-VNFCP heuristic to an incremental heuristic from literature [7]. The reference algorithm uses a Viterbi-inspired multi-stage graph to assess the weighted costs of routes and new VNF instantiations for individual demands as they arrive. Its cost function includes VNF deployment, energy consumption, traffic forwarding, and service level agreement violation, all of which are based on measures that correspond to our four objectives. Note that it may produce unfeasible solutions w.r.t. our constraints if the respective penalty is considered cheaper than the agreement's satisfaction. Such placements are still included here.

For this evaluation, 50 runs with varying demand distributions are performed for each problem instance. Since the *Internet2* problem instance is relatively small in terms of network nodes, CPU locations, and requests (cf. Table II), our heuristic is configured to optimize for 20 seconds. In case of the remaining problem instances, we set the optimization time to 60 seconds to achieve reliable results with the larger search space as based on empirical studies. Note that these execution times shall only provide an impression of magnitude with common hardware resources (cf. Section V-*Setup*). Thereby, the influence of the execution environment is negligible in comparison to the problem scale and difficulty, as the former usually only varies within the same magnitude.

In contrast, the reference algorithm terminates upon finding one placement and takes between 1 and 5 seconds per execution.

In order to compare our multi-objective Pareto frontier approximation X with a single-objective heuristic solution y, the hypervolume is tweaked to represent a single solution.

$$I_{sH}(X) := \max_{x \in X} I_H(x) \tag{25}$$

Then, the following indicator quotients $Q_\bullet$ express their relative performance.

$$Q_H := \frac{I_{sH}(X)}{I_H(y)};\ Q_\varepsilon := \frac{I_\varepsilon(y)}{I_\varepsilon(X)};\ Q_{WS} := \frac{I_{WS}(y)}{I_{WS}(X)} \tag{26}$$

Thereby, values $Q_\bullet \geq 1$ indicate that our results were *better* in relation to the reference algorithm w.r.t. the used indicator, while values < 1 express worse performance.

The results of the performance comparison are shown in Figure 6. For each problem instance and each indicator, the empirical cumulative distribution function of the 50 quotient values is displayed. Based on this performance assessment, MO-VNFCP outperforms the incremental heuristic clearly in most cases. The only exception is the *Germany.2* problem. In half of these executions, the reference algorithm obtains a slightly lower weighted sum value compared to our approach. However, the differences are within roughly 5%, while the remaining indicators still prefer our heuristic.

In general, the biggest differences are observable for those problem instances with the most available resources. The mean quotient values for *Geant.1* are in the range $2.08-2.82$ for the

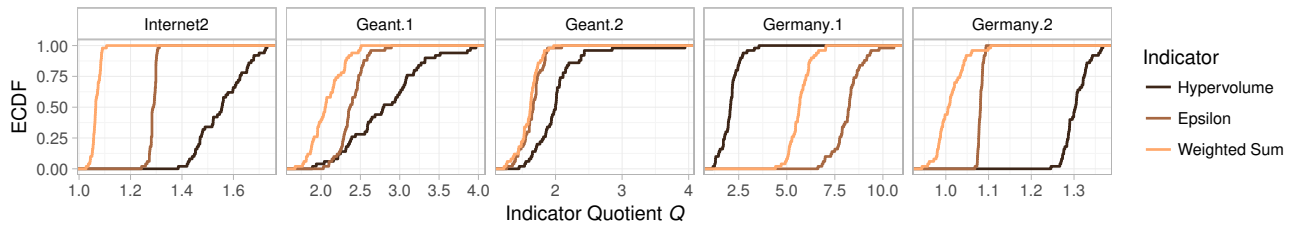


Fig. 6. Relative performance of this work's approach compared to an incremental heuristic from literature [7].

different indicators, which corresponds to a mean improvement of more than 100% in every case. For the *Germany.1* topology, even higher values within $2.07 - 8.21$ are observed, with the highest indicator quotient suggesting an improvement of more than 900% with our approach. These findings highlight the potential for regular re-optimization of placements that are acquired by faster, incremental heuristics. These are usually limited by a local view on the overall problem at the moment they place new instances due to time constraints.

Note that, without knowing the actual Pareto frontier, no general statement regarding the quality of the placements is possible. The above measures only allow a relative comparison which is tied to the applied indicators, their weights and reference points, and the evaluated problem instances. However, by application of multiple indicators and the evaluation with varying topologies, we attempt to provide a thorough view on the algorithm's performance.

VI. Related Work

This section provides an overview of publications related to the VNFCP problem. First, works with a similar problem formulation are presented. Then, several related problems that deal with different subproblems of the VNFCP are discussed. Finally, we cover works that provide solutions to special cases of the VNFCP by simplifying the problem formulation.

A. VNF Chain Placement

In [14], one of the first formal problem statements for the VNFCP is provided. The problem is tackled with an ILP-based approach that takes into account the network demands of virtual machine requests as well as latency constraints. Optimization goals include minimizing the number of servers that host VNFs as well as the total resource usage for VNF instantiation and request handling. Similarly, the authors of [15] and [16] use ILPs to model the VNFCP. Additionally, both provide heuristics that can be used in the context of larger problem instances. These heuristics are based on backtracking or time-limited ILPs with additional constraints, respectively. Furthermore, [17] extends the work in [16] by adding the variable neighborhood search meta-heuristic [18].

In addition to developing an ILP-based solution for static problem instances with a given set of demands, Bari et al. [7] propose a heuristic that is also capable of calculating placements dynamically. In this case, newly arriving demands can be added to the current placement and new VNF instances are created on demand. The objective function in this optimization is defined as a weighted sum of costs, including deployment, energy, and traffic forwarding expenses. Since the authors provide the source code of their implementation as well as the input data, we can perform a comparison between their work and ours (cf. Section V-B).

While the aforementioned approaches deal with a single objective function, the authors of [19] propose three different goal functions and perform a Pareto set analysis of the resulting placements. In this context, placements that are optimized according to each of these functions individually, are compared with each other. In contrast, our work deals with the VNFCP in a multi-objective manner, allowing decision makers to investigate trade-offs between feasible solutions and to define preferences given the available alternatives.

B. Subproblems of the VNFCP

Although the following approaches do not address all parts of VNFCP, they can provide useful insights into the individual steps required for developing VNFCP algorithms. For instance, the authors of [20] focus on routing of network demands while assuming that the position of VNF instances is known beforehand. In an analogous fashion, the locations of virtual machines (VMs) that can host VNFs are fixed in [21] and the task consists of mapping and scheduling demands to VMs.

Virtual Network Embedding (VNE) problems [22], [23] overlap with the VNFCP in terms of the chaining and placement aspects. However, isolation between demands is stricter in the context of VNE, i.e., multiple demands can not share the same resource. Similarly, Virtual Machine Allocation (VMA) problems [24] deal with the task of deciding which VMs to place on which node in order to balance aspects like energy efficiency and Quality of Service (QoS).

C. Special Cases

Rather than covering a subset of the VNFCP, the following publications discuss problems that can be considered special cases. By reducing the number of constraints or available VNF types, algorithms that solve the VNFCP could also be applied to these problems. One common difference to the algorithm developed in this work lies in considering only one type of VNF type, as proposed in [25], [26], and [27]. Additionally, these works aim at minimizing only a single objective function, e.g., the number of VNF instances or the costs for installation and migration of instances.

VII. Conclusion

Network Functions Virtualization (NFV) addresses many drawbacks of today's wide area networks by leveraging virtualization techniques and replacing specialized, expensive hardware middleboxes with easily scalable software instances that run on off-the-shelf servers. However, the NFV paradigm also introduces new challenges in the domain of management and orchestration that need to be addressed for optimal results.

In particular, this work focuses on the VNF chain placement problem (VNFCP) which encompasses questions regarding the number of VNF instances, their location, and the routing of network demands through service chains. Additionally, placements need to meet constraints that correspond to SLAs while optimizing factors like costs or end user satisfaction. To this end, we propose MO-VNFCP, a multi-objective heuristic algorithm that calculates a set of Pareto optimal placements with respect to several objective functions. On the one hand, this can be used to analyze trade-offs between possibly competing objectives. On the other hand, heuristics usually yield results significantly faster than ILP-based approaches. Therefore, they constitute a more feasible approach for coping with large scale problem instances and dynamic scenarios.

Apart from designing the multi-objective heuristic, we perform a case study in order to trace its solution set over time. This highlights the different stages of the algorithm, i.e., exploration in the beginning and targeted improvement of solutions in the end, and is achieved by analyzing three quality indicators for Pareto frontiers during its run time. Finally, a performance comparison with a greedy heuristic from literature is conducted. The fact that the latter is outperformed by our algorithm in the majority of scenarios demonstrates our algorithm's feasibility for tackling the VNFCP.

There are several directions for future work. First, we plan to investigate the algorithm's performance and applicability in the context of topologies and network demands that reflect additional real world scenarios as well as the scalability of individual modules within VNFs. Second, new objectives like resilience can be taken into account during the optimization. This can identify characteristics of resilient placements as well as provide information about the resulting trade-offs w.r.t. the remaining objectives. Finally, mechanisms for online optimization can build on the aforementioned insights in order to place incoming demands in a timely manner.

Acknowledgments

This work has been performed in the framework of the CELTIC EUREKA project SENDATE-PLANETS (Project ID C2015/3-1), and it is partly funded by the German BMBF (Project ID 16KIS0474). The authors alone are responsible for the content of the paper.

References

[1] Network Functions Virtualisation – Update White Paper. 2013. [Online]. Available: https://portal.etsi.org/NFV/NFV_White_Paper2.pdf

[2] M. Seufert, S. Lange, and M. Meixner, "Automated Decision Making Methods for the Multi-objective Optimization Task of Cloud Service Placement," in *1st International Workshop on Programmability for Cloud Networks and Applications (PROCON)*, 2016.

[3] B. Korte and J. Vygen, *Combinatorial optimization*. Springer, 2012.

[4] G. Dósa and J. Sgall, "First fit bin packing: A tight analysis," in *LIPIcs-Leibniz International Proceedings in Informatics*, 2013.

[5] P. Czyżżak and A. Jaszkiewicz, "Pareto simulated annealing - a metaheuristic technique for multiple-objective combinatorial optimization," *Journal of Multi-Criteria Decision Analysis*, 1998.

[6] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann, "Heuristic Approaches to the Controller Placement Problem in Large Scale SDN Networks," *IEEE Transactions on Network and Service Management*, 2015.

[7] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *Network and Service Management (CNSM), 2015 11th International Conference on*, 2015.

[8] K. Deb, K. Sindhya, and J. Hakanen, "Multi-objective optimization," in *Decision Sciences: Theory and Practice*, 2016.

[9] M. Ehrgott, *Multicriteria Optimization*. Springer, 2005.

[10] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE transactions on evolutionary computation*, 2003.

[11] J. Knowles, L. Thiele, and E. Zitzler, "A tutorial on the performance assessment of stochastic multiobjective optimizers," ETH Zurich, Switzerland, Tech. Rep. 214, Feb 2006, revised version.

[12] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler, "Theory of the hypervolume indicator: optimal μ-distributions and the choice of the reference point," in *Proceedings of the tenth ACM SIGEVO workshop on Foundations of genetic algorithms*, 2009.

[13] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici, "ClickOS and the Art of Network Function Virtualization," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2014.

[14] H. Moens and F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *10th International Conference on Network and Service Management (CNSM) and Workshop*, 2014.

[15] S. Sahhaf, W. Tavernier, M. Rost, S. Schmid, D. Colle, M. Pickavet, and P. Demeester, "Network service chaining with optimized network function embedding supporting service decompositions," *Computer Networks*, 2015.

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

[17] M. C. Luizelli, W. L. da Costa Cordeiro, L. S. Buriol, and L. P. Gaspary, "A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining," *Computer Communications*, 2016.

[18] P. Hansen and N. Mladenović, "Variable neighborhood search: Principles and applications," *European journal of operational research*, 2001.

[19] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *3rd International Conference on Cloud Networking (CloudNet)*, 2014.

[20] A. Dwaraki and T. Wolf, "Adaptive Service-Chain Routing for Virtual Network Functions in Software-Defined Networks," in *Proceedings of the 2016 workshop on Hot topics in Middleboxes and Network Function Virtualization*, 2016.

[21] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and S. Davy, "Placement and scheduling of functions in network function virtualization," *arXiv preprint arXiv:1512.00217*, 2015.

[22] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys & Tutorials*, 2013.

[23] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, "Vnr algorithm: A greedy approach for virtual networks reconfigurations," in *Global Telecommunications Conference (GLOBECOM 2011)*, 2011.

[24] Z. Á. Mann, "Allocation of virtual machines in cloud data centers—a survey of problem models and optimization algorithms," *ACM Computing Surveys (CSUR)*, 2015.

[25] M. Bagaa, T. Taleb, and A. Ksentini, "Service-aware network function placement for efficient traffic handling in carrier cloud," in *Wireless Communications and Networking Conference (WCNC)*, 2014.

[26] M. Bouet, J. Leguay, T. Combe, and V. Conan, "Cost-based placement of vDPI functions in NFV infrastructures," *International Journal of Network Management*, 2015.

[27] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R. Boutaba, "Elastic virtual network function placement," in *4th International Conference on Cloud Networking (CloudNet)*, 2015.