Performance Benchmarking of Network Function Chain Placement Algorithms

Alexej Grigorjew, Stanislav Lange, Thomas Zinner, and Phuoc Tran-Gia

University of Würzburg
{alexej.grigorjew, stanislav.lange, zinner, trangia}@informatik.uni-wuerzburg.de

Abstract. The Network Function Virtualization (NFV) paradigm enables new flexibility and possibilities in the deployment and operation of network services. Finding the *best* arrangement of such service chains poses new optimization problems, comprising a combination of placement and routing decisions. While there are many algorithms on this topic proposed in literature, this work is focused on their evaluation and on the choice of reference for meaningful assessments. Our contribution comprises two problem generation strategies with predefined optima for benchmarking purposes, supplemented by an integer program to obtain optimal solutions in arbitrary graphs, as well as a general overview of concepts and methodology for solving and evaluating problems. In addition, a short evaluation demonstrates their applicability and shows possible directions for future work in this area.

Keywords: NFV, VNF Chain Placement, Optimization, Performance Evaluation, Problem Generation

1 Introduction

In modern networks, operators apply various network functions to their traffic flows, either due to their specific requirements or due to the network's policy in general. Packets are monitored, modified, or even dropped to perform these functions, such as firewalls, deep packet inspection, load balancers, and core gateways in LTE networks. Traditionally, they are implemented by special hardware middleboxes with high performance guarantees, but they also suffer from high costs, low flexibility, low scalability, and vendor dependence. These problems are addressed by the Network Function Virtualization (NFV) paradigm [1]. Hardware middleboxes are replaced by software instances, running in virtualized environments on cheap, vendor independent commercial-off-the-shelf (COTS) machines.

With the newly attained flexibility, new optimization problems arise in the context. In particular, the Virtual Network Function Chain Placement (VNFCP) problem deals with the orchestration of Virtual Network Functions (VNFs) in the network: (a) how many VNF instances are needed, (b) where are they located, and (c) which traffic flow is using which instance. Thereby, various constraints are considered, such as bandwidth and resource limitations. The optimization objective usually involves the maximization of profit or the minimization of costs, for example by reducing the number of instances.

As the VNFCP problem is NP-hard [2], most publications rely on heuristics and approximations for productive use, featuring varying objectives, constraints, and levels of detail. However, assessing the quality of attained solutions is no trivial task in itself. For example, comparing the performance of two heuristics without knowing an optimal reference placement only yields limited expressiveness. In order to overcome these problems, this work investigates different approaches towards the performance evaluation of VNFCP algorithms. In particular, it discusses the generation of parameterized artificial problem instances with known optimal solutions, providing an independent reference for comparison.

The remainder of this work is structured as follows. Section 2 formally introduces a common variant of the VNFCP problem. In Section 3, different approaches for its solution are reflected with special regard to their respective evaluation techniques. In addition, Section 3.1 includes an Integer Linear Program (ILP) that obtains optimal solutions for small problem instances. Section 4 proposes artificial problem generation by means of two concrete strategies, whose results are demonstrated in Section 5. Related work on the evaluation of network optimization heuristics is addressed in Section 6. Finally, in Section 7, we discuss remaining challenges and conclude the paper.

An exemplary implementation of the presented ideas, such as the ILP and the problem generation strategies, is also available on GitHub[1].

2 Virtual Network Function Chain Placement

This section presents a brief overview of the VNFCP problem. Note that, as different approaches consider different details of the problem, this work only provides a generic definition of the most commonly used models. For further details, please refer to the corresponding publications in Section 6.1, and in particular to [3] for a more detailed version of this specific model.

2.1 Input and Output Models

The input model usually comprises three components. The network is represented by an undirected graph $G = (V, E)$. Nodes $v \in V$ may have computational resources $v_{\text{cpu}} \in \mathbb{R}$, while links $e \in E$ have bandwidth and delay properties $e_{\text{bw}}, e_{\text{d}} \in \mathbb{R}$. The available network function types $t \in T$ require a certain amount of resources $t_{\text{cpu}} \in \mathbb{R}$ and possess similar attributes as links: $t_{\text{bw}}, t_{\text{d}} \in \mathbb{R}$. Finally, each traffic request $r \in R$ consists of source and destination nodes $r_{\text{src}}, r_{\text{dst}} \in V$, bandwidth and maximum latency requirements $r_{\text{bw}}, r_{\text{d}} \in \mathbb{R}$, and a sequence of network functions $r_{\text{c}} \in T^{|r_{\text{c}}|}$ which represent the requested function chain.

The output model contains the set of placed VNF instances $z \in \mathcal{I}$ which possess a type $z_{\text{type}} \in T$ and location $z_{\text{node}} \in V$. In addition, for every traffic

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

demand $r \in R$, the assigned route $r_{\text{path}} \in V^{|r_{\text{path}}|}$ and the used instances $r_{\text{inst}} \in (\mathcal{I} \cup \{\emptyset\})^{|r_{\text{path}}|}$ are given. Hereby, r_{inst} has the same length as r_{path}, and $\emptyset$ indicates that no VNF is applied at the respective location of the route.

2.2 Constraints

The considered constraints can be split into two categories. On the one hand, consistency between different variables must be ensured. For example, the location of the i-th instance in r_{inst} must equal the i-th node in r_{path}, or more fundamentally, for each subsequence (v_i, v_{i+1}) in r_{path}, there must be an edge in the network graph, i.e., $\{v_i, v_{i+1}\} \in E$. On the other hand, resource and delay requirements must be met. This includes CPU capacities v_{cpu} on nodes, bandwidths e_{bw} on links, capacities t_{bw} of the instances $z \in \mathcal{I}$, and maximum flow latencies as defined by r_{d}. Further details are omitted in this work, as they differ between different approaches.

2.3 Objectives

The range of considered objectives varies greatly among existing work in literature. Typical atomic objective functions include the number of placed instances, the amount of consumed computational resources, the number of active nodes, cumulative delay and number of hops for all requests, and the number of violated service level agreements. These may either be considered individually, consolidated by applying weights, or be treated by a multi-objective optimizer.

Note that the choice of objective functions determines the optimal solutions. In order to assess the quality of placements, they need to be specified beforehand.

3 Approaches for Solutions and their Assessment

This section outlines different strategies to solve the VNFCP and common ways to evaluate their performance.

3.1 Exhaustive Optimization

The most evident approach is to compute an exact solution to a given optimization problem, or the exhaustive Pareto frontier in a multi-objective context. This strategy is sometimes implemented in a simple brute force manner, but more frequently realized by Integer Linear Programs (ILPs) for performance reasons.

Given their exact results, a qualitative evaluation is not necessary for exhaustive approaches. They always provide the *best* results with regard to their respective optimization objective. However, since the VNFCP problem is NP-hard, these methods are usually limited in applicability. Hence, they can be evaluated with respect to the supported details of their model, their required computational resources, and the runtime of an optimization, while also considering the supported scale of problem input. More importantly, the results of these exhaustive solvers can be used as a reference for the evaluation of faster, empirical approaches, as explained in Section 3.3.

Table 1. Variables and indices used in the ILP formulation.

Index	Description
$r \in \{1, ..., \lvert R\rvert\}$	Traffic request.
$f \in \{1, ..., \lvert r_c\rvert + 1\}$	Function in the chain of a request.
$n \in \{1, ..., \lvert V\rvert\}$	Node in the network graph.
$e \in \{1, ..., \lvert E\rvert\}$	Edge in the network graph.
$t \in \{1, ..., \lvert T\rvert\}$	Type of the respective network function.
$i \in \{1, ..., \lvert R\rvert\}$	Instance of the respective type on the respective node.
Variable	**Description**
$c_{r,f,n} \in \{0, 1\}$	Indicates whether function f of request r is served in node n.
$z_{r,f,i} \in \{0, 1\}$	Indicates whether function f of request r is served on the i-th instance of its type.
$a_{r,f,e} \in \{0, 1\}$	Indicates whether edge e is used by function f of request r.
$m_{r,f,n,i} \in \{0, 1\}$	$c_{r,f,n} \wedge z_{r,f,i}$
$m_{t,n,i} \in \{0, 1\}$	$\max_{(r,f) \in \{(r,f) \mid \text{funct. } f \text{ of req. } r \text{ is of type } t\}} \{m_{r,f,n,i}\}$
$m_{t,n} \in \mathbb{N}$	$\sum_{i=1}^{\lvert R\rvert} m_{t,n,i}$

Example ILP. In the following, an example ILP is presented that primarily minimizes CPU utilization. It can be adjusted and used on small problem instances for the evaluation of heuristics with similar problem models.

Variables. The program is based on three types of decision variables: $c_{r,f,n}$ and $z_{r,f,i}$ indicate the location and number of an instance, and $a_{r,f,e}$ indicates the path of a request r, with f being the number of the corresponding function in the request's chain, n being a node in the network graph, e being an edge, and i the number of the instance on the respective node. In addition, several auxiliary variables are used to ease the definition of constraints and objectives: $m_{r,f,n,i}$ indicates whether the i-th instance of its type on node n is used for function f of request r. Similarly, $m_{t,n,i}$ indicates whether *any* request with a function of type t uses this instance. Finally, $m_{t,n}$ contains the *number* of instances of type t on node n. All used variables and indices are summarized in Table 1.

Note that each subpath from function $f - 1$ to function f is modeled separately, with $f = |r_c|$ being the last function, and $f = |r_c| + 1$ representing the destination of the traffic request. The latter is only used by the variables $a_{r,f,e}$ for the last subpath.

Objective. The primary objective is to minimize the CPU utilization, but on a second priority, the number of hops was included, e.g., to avoid loops in the paths. Using the variables above, the objective is expressed as follows.

$$\text{Minimize} \quad 0.99 \sum_{t=1}^{|T|} \sum_{n=1}^{|V|} m_{t,n} \cdot t_{\text{cpu}} + 0.01 \sum_{r=1}^{|R|} \sum_{f=1}^{|r_c|+1} \sum_{e=1}^{|E|} a_{r,f,e} \tag{1}$$

Thereby, the weights are chosen such that the amount of hops does not impair the minimization of the primary objective, i.e., the number of used CPU resources.

Constraints. The following constraints are used by the ILP to ensure path consistency and to respect resource and capacity utilization. Given an edge $e \in$

$\{1, ..., |E|\}$, let e_1 and e_2 be the respective nodes that it connects. Further, let $L(n)$ be the set of incident edges of node n, $F(t)$ be the set of all functions (r, f) with type t, and $\tau(r, f)$ be the type t of the f-th function of the request r. Note that special cases, such as the subpath towards the final destination of a request, are omitted to retain clarity. For possible values for the indices r, f, n, e, t, and i, see Table 1.

$$\forall r, f \ : \ \sum_{n=1}^{|V|} c_{r,f,n} = 1 \quad \text{and} \quad \sum_{i=1}^{|R|} z_{r,f,i} = 1 \tag{2}$$

$$\begin{aligned} \forall r, f, e \ : \ & a_{r,f,e} \le \sum_{x \in L(e_1) \setminus e} a_{r,f,x} + c_{r,f,e_1} + c_{r,f-1,e_1} \\ & \text{and} \quad a_{r,f,e} \le \sum_{x \in L(e_2) \setminus e} a_{r,f,x} + c_{r,f,e_2} + c_{r,f-1,e_2} \end{aligned} \tag{3}$$

$$\begin{aligned} \forall r, f, n \ : \ & c_{r,f,n} \le \sum_{x \in L(n)} a_{r,f,x} + c_{r,f-1,n} \\ & \text{and} \quad c_{r,f-1,n} \le \sum_{x \in L(n)} a_{r,f,x} + c_{r,f,n} \end{aligned} \tag{4}$$

$$\forall n \ : \ \sum_{t=1}^{|T|} m_{t,n} \cdot t_{\text{cpu}} \le n_{\text{cpu}} \tag{5}$$

$$\forall e \ : \ \sum_{r=1}^{|R|} \sum_{e=1}^{|E|} a_{r,f,e} \cdot r_{\text{bw}} \le e_{\text{bw}} \tag{6}$$

$$\forall t, n, i \ : \ \sum_{(r,f) \in F(t)} m_{r,f,n,i} \cdot r_{\text{bw}} \le t_{\text{bw}} \tag{7}$$

$$\forall r \ : \ \sum_{f=1}^{|r_c|+1} \sum_{e=1}^{|E|} e_{\text{d}} \cdot a_{r,f,e} + \sum_{f=1}^{|r_c|} \tau(r, f)_{\text{d}} \le r_{\text{d}} \tag{8}$$

Hereby, Equation 2 ensures that, for every requested function, there is exactly one location and one instance assigned. Equation 3 ensures that the paths are connected: for every incident node of a link e, there is at least one other link x in use, or the node is an endpoint of this subpath. With Equation 4, all used instance locations enforce the usage of an incident link, ultimately leading to another instance location (or the request's destination) with the above constraints. The set of Equations 5, 6, and 7 respect the available CPU resources on nodes, the available bandwidth on links, and the available computational power of instances, respectively. Finally, Equation 8 maintains the service level agreements with respect to the flows' delays.

3.2 Approximation Algorithms

Approximation algorithms provide solutions with *guaranteed* quality bounds. They are designed such that their results are provably within a multiplicative (or sometimes additive) distance from the optimal solution. However, the approximability of the VNFCP problem varies greatly with its details and assumptions, such as the actual objective of the optimization and considered constraints. There are only few publications in literature that propose an approximation algorithm for the VNFCP problem in polynomial time (cf. Section 6).

Given their provable bounds, the evaluation of approximation algorithms is primarily of analytical kind and deals with their worst case performance. Assessing their expected, average performance by analytical means is difficult as the definition of an *expected* problem scenario itself is unclear. However, they can be evaluated by empirical means with respect to selected representative problem characteristics from real world scenarios. Therefore, similar approaches as described in Section 3.3 can be applied here as well.

3.3 Heuristics

Heuristic algorithms attempt to find sufficiently good solutions within feasible computational efforts and time limits. These approaches vary greatly in applied strategy, requirements, and runtime, and therefore, also in their results' quality. Their underlying ideas include simple greedy heuristics, pre-calculation of desired parameters, iterative improvements, relaxation of ILPs, fixing and optimizing only parts of the whole problem, and finally meta-heuristics such as simulated annealing and evolutionary algorithms. A selection of related work in this category is presented in Section 6.

An empiric evaluation of heuristic algorithms is based on a comparison of the resulting objective values with chosen reference solutions. The selection of these references affects the expressiveness of the evaluation.

Comparison with other heuristics. When comparing two or more heuristic algorithms, the main statement is which algorithm performs better than the other with respect to the selected problem scenarios and objectives. However, there is no meaningful *quantitative* estimation of the overall performance of both algorithms without an *independent* reference point. Even with statements such as *"algorithm A performs x% better than algorithm B"*, they could still both be significantly worse than the optimal attainable value, which should be reflected by the evaluation. In many cases, simple baseline algorithms are used for the comparison, which further reduces its expressiveness. Nevertheless, as only heuristics are used, the scale of the evaluation is only limited by the algorithms' capabilities, as opposed to the limitations of acquiring optimal references.

Ideal reference points. The ideal solution (or true Pareto frontier for multi-objective problems) can be used to overcome the above issues. As it is not tied to the used algorithm but only to the problem itself, it serves as an independent representative for the investigated heuristics. However, obtaining these ideal points is not trivial in itself. Exhaustive solvers as described in Section 3.1 are only applicable for small problems due to their runtime, but the investigation of more complex scenarios is usually more interesting. Hence, the generation of synthetic problem instances with predefined optimal solutions is proposed in this work. This enables an empirical evaluation of larger scaled problems with parameterized characteristics. At the same time, the generation of such problems is tied to predefined strategies and objectives. Two simple generation strategies are presented in Section 4.

4 Synthetic Problem Generation

In general, there are two conceivable techniques for the generation of problems with known solutions. On the one hand, the problem structure could be restricted in such a way that, for an aware algorithm, the computation of solutions is facilitated significantly. An example for such a strategy is given by the grid graphs in Section 4.1. On the other hand, the problem could be created in conjunction with, or even based on, the actual solution instead of dealing with its acquisition subsequently. This type of strategy is applied for the dynamic resource distribution in Section 4.2.

In any case, the presence of solutions for a problem instance requires that the objective is fixed beforehand. Consequently, most generation strategies are tied to a specific objective. On the contrary, other characteristics of the problem can be defined more flexibly. Having more parameters for the generation process provides more control of the scale and difficulty of the problem, which helps to reveal properties of the investigated algorithms.

However, in every meaningful performance evaluation, the results should be reviewed with respect to the synthetic problem's structure and its influence on them. General statements are difficult to obtain from empirical evaluations, but the observed behavior shall not only be caused by the choice of evaluation scenario. Therefore, it can be supplemented by an independent evaluation with different, smaller problem instances and optimal references obtained by an ILP, such as in Section 3.

4.1 Grid Graph Problem

The grid graph problem (GGP) is designed to be a simple multi-objective placement problem, minimizing both CPU utilization and number of hops simultaneously. Due to its simple structure, these measures are directly proportional to the number of instances and the overall delay, respectively. Therefore, they may be minimized here as well.

An overview of the graph's layout is presented in Figure 1. The positions of the nodes are based on a grid layout. There are m source and destination nodes in the network, located at the first and last stage of the graph, respectively. In each of the the n intermediate stages, there are k nodes with computational resources. Each node is only connected to its direct neighbors in the grid.

The idea of this scenario is to define very similar traffic requests, with equal properties except for their source and destination, and all traffic flowing into the same direction. The requests include n different VNF types, one for each of the inner stages. Each of them has the same properties, and a single instance of each

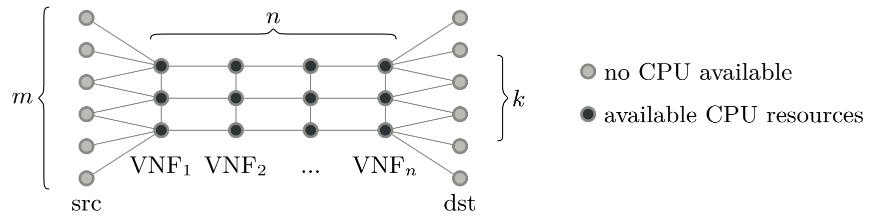


Fig. 1. Grid graph problem scenario.

type is sufficient to satisfy all traffic requests. Each node in the n intermediate stages can host exactly one instance. Each link in the network has the same delay and sufficient bandwidth to support every request $n+1$ times. Finally, each traffic demand requests all of the n VNFs in the same order and its maximum tolerated delay allows it to visit all of them, regardless of their position.

In addition to the above parameters k, m and n, the load parameter $\rho \in [0,1]$ controls the amount of requests in the scenario, i.e., there will be $\rho \cdot m^2$ traffic demands generated with random choices from the m available source and destination nodes. The dimensions of other measures, such as bandwidths and CPU resources, may be tweaked as well if necessary.

Optimal Solutions. In this example, the optimal placement is computed after the problem generation. The k horizontal rows of length n in the intermediate stages are referred to as *lanes*. All Pareto optimal solutions place all of the n functions in a straight lane, exactly in the order they are requested, from left to right. This avoids the introduction of unnecessary hops. As this is a multi-objective scenario, the CPU utilization is minimized as well, hence, all possible numbers $k' \in \{1, ..., k\}$ of occupied lanes are tested. For each such k', there are $\binom{k}{k'}$ possibilities for chosen lanes, thus, a total of $2^k - 1$ tests must be done.

The optimality of these solutions can be proved by contradiction: assuming the use of an incomplete row of VNFs at some point, there must be additional hops to reach the remainder of the requested service chain, which could have been avoided otherwise. Due to the limited space, only the idea is outlined here though.

4.2 Dynamic Resource Distribution

The idea of the dynamic resource distribution is to decouple the generation of the network graph from the traffic demands in order to enable a more versatile evaluation. In particular, any graph can be used without restrictions on its structure, including real topologies and randomly generated ones, however, the available resources and link bandwidths are altered in the process.

The intended objective is to minimize the number of placed instances on the graph. In its current implementation, the strategy is limited to a single VNF of the same type for each request, however, improvements are planned for future work. The generation process is comprised of the following steps.

1. Generate or choose an existing network topology.
2. Distribute a predefined number of instances on a predefined number of nodes in the network.
3. Generate traffic demands with random source-destination-pairs until all instances are fully used.
   (a) Pick a random source-destination-pair from all node pairs.
   (b) Pick one of the VNFs with remaining capacity that implies the smallest detour from the shortest source-destination-path.
   (c) Select the requested bandwidth for this traffic demand from a predefined range. Ensure optimality by filling instances up: If the remaining capacity of the instance would be too small to allow another traffic demand, increase the bandwidth such that it is fully utilized.
   (d) Define the maximum tolerated delay within a factor of the selected path's latency, e.g., the twofold.
4. Distribute sufficient CPU resources in the network such that all intended instances can be placed. Add further resources to allow more variation from the evaluated algorithms. This includes increasing existing resources as well as adding new resources to previously unused nodes.
5. Define the link bandwidths such that all selected paths are supported. Multiply the required amount by a predefined factor to enable more versatile results from the evaluated algorithms.

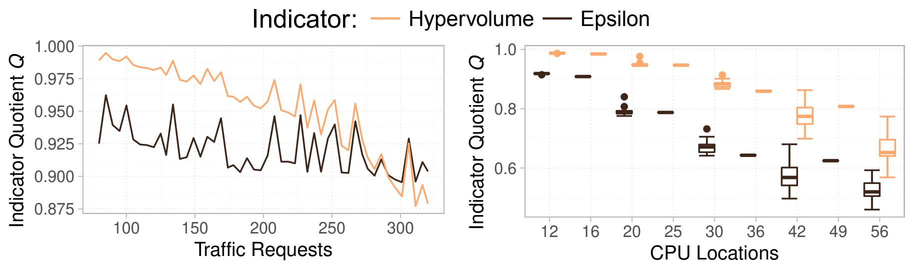
As evident from this process, the optimal solution is constructed in conjunction with the problem. More accurately, it is defined in steps (2) and (3b). Optimality of this problem-solution-pair is achieved by (3c), which ensures that all intended instances are fully utilized, and therefore, this is the minimum required number.

Despite its flexible definition, the generated problems are currently limited in variation with regard to requested VNF chains. The generation of longer chains and the possibility to support more objectives will be investigated in future work.

5 Evaluation

In order to show the benefits of synthetic problem generation, the strategies from Section 4 are demonstrated by means of a few examples. Therefore, the algorithms from [3] and [2] are applied.

Multi-Objective Quality Indicators. For the evaluation of multi-objective problem scenarios such as the grid graph problem, all n obtained solutions $s_{1..n} \in \mathbb{R}^n$ are aggregated into a single quality indicator value $q \in \mathbb{R}$ and compared directly. As this accompanies a loss of information, multiple types of indicators that represent different performance aspects are usually used. In this work, the hypervolume I_H and epsilon indicator I_ϵ are applied [4–6]. Hereby, the hypervolume indicator measures the volume in the solution space enclosed by the returned set and a reference point, while the epsilon indicator measures how much a reference solution set (e.g., the real Pareto frontier) must be stretched so it becomes worse than the evaluated set.



(a) Influence of the traffic requests. (b) Influence of the CPU locations number.

Fig. 2. Influence of GGP's parameters on the placement quality of MO-VNFCP.

For a direct comparison with the optimal values, their ratio is computed. Note that for the hypervolume, bigger values indicate a better performance, while the opposite applies to the epsilon indicator. Hence, the indicator quotients for heuristic solutions S and optimal solutions O are defined as follows.

$$Q_H(S,O) := \frac{I_H(S)}{I_H(O)}; \qquad Q_\epsilon(S,O) := \frac{I_\epsilon(O)}{I_\epsilon(S)} \tag{9}$$

Thereby, all values are within the range $[0,1]$, where 1 represents the optimal performance with respect to this indicator.

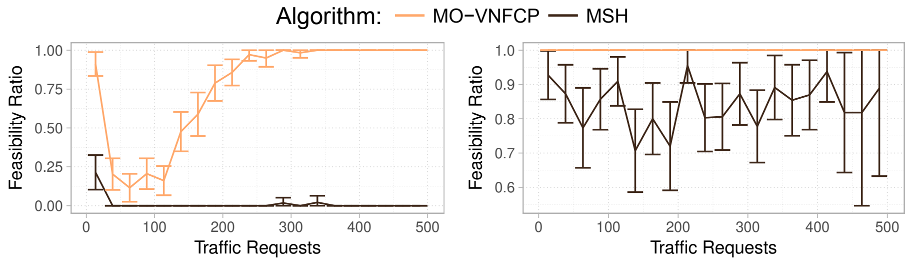
5.1 Grid Graph Problem

With the GGP scenario, the effects of different parameters on the performance of the MO-VNFCP algorithm [3] are investigated. Firstly, Figure 2 contains the resulting indicator ratios.

Note that, without the optima available for comparison, displaying the pure indicator values in a similar way would be significantly less expressive. By increasing the parameters of the scenario, the optimized problem instances differ between two runs, and so does their difficulty. As the optima change in a similar way, they are used to provide a relative view on the performance and a consistent overall representation of attained quality.

Figure 2a displays the influence of the number of traffic requests on the indicator quotient. The load parameter ρ is increased from 0.2 to 0.8 with $m = 20$ source and destination nodes, resulting in $\rho \cdot m^2 = 80$ to 320 requests. The parameters for the intermediate stages are $k = 4$ and $n = 3$. Both indicators show a decreasing trend in quality values, however, the hypervolume is affected stronger than the epsilon indicator, which implies that they indeed represent different aspects of performance.

In Figure 2b, the number of nodes in the intermediate stages is varied instead, with $k \in [3,7]$ and $n \in [4,8]$. This leads to an increased number of locations with CPU resources available for the algorithm to choose from, but also increases the length of the embedded VNF chains. The figure displays box plots which enclose the first and third quartile of attained indicator values. The whiskers extend to



(a) Feasibility ratios with multiplier 1.5. (b) Feasibility ratios with multiplier 5.0.

Fig. 3. Feasibility ratios with different bandwidth multipliers.

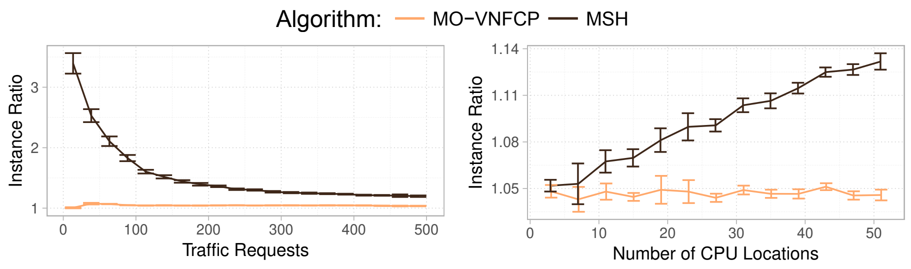
the furthest point, but at most to the 1.5-fold of their boxes' height. Similarly to before, increasing the scale of the problem leads to a decrease in solution quality shown by both indicator types. However, increasing the intermediate nodes from 12 to 56 has a greater impact on both indicators compared to raising the number of requests from 80 to 320. This analysis helps to identify the algorithm's capabilities and shows opportunities for improvement.

5.2 Dynamic Resource Distribution

With the dynamic resource distribution scenario, the behavior of the algorithms MO-VNFCP [3] and MSH (Multi-Stage Heuristic) [2] are compared. Note that, while this scenario only considers the number of instances, both algorithms still try to optimize other measures simultaneously, such as the number of hops. All measurements were conducted on a real topology, namely the Germany graph [7].

As an initial quality measure, Figure 3 displays the influence of the number of traffic demands on the feasibility ratio (i.e., the relative amount of feasible solutions within all optimization runs) for two different bandwidth parameters with 95% confidence intervals. In Figure 3a, the available link bandwidths were set to the 1.5-fold of what the optimal solution required. Here, MSH did not return a feasible solution in most cases as it tends to congest links. For MO-VNFCP, increasing the scale of the problem improved its solvability as with more traffic requests there are also more available resources in the graph, allowing more variation in the selected paths. In Figure 3b, MO-VNFCP is always able to return a feasible solution, while there is no significant dependency observable for MSH. Its 95% confidence intervals range from 0.6 to 1.0 for multiplier 5.0. The following measurements were also conducted with this configuration.

Figure 4 shows the number of used VNF instances in relation to the optimal value. In Figure 4a, the influence of the number of requests is shown. Interestingly, the *relative* performance of the MSH improves significantly with larger problems. This is caused by a broad distribution of instances by the heuristic, and its relative quality improves when the optimal placement also requires more instances. On the other hand, Figure 4b shows the dependency on the number of nodes with available CPU resources. The more choices exist, the more difficult the problem becomes, but the optimal number of instances stays the



(a) Influence of traffic requests. (b) Influence of CPU locations.

Fig. 4. Influence of problem parameters on the number of instances.

same. Hence, the relative performance of the MSH deteriorates in this example. Figure 4 also shows that in both cases, the performance of the MO-VNFCP heuristic was very close to the optimal solution with no significant dependency on the investigated scale. Taking its longer runtime into account, it presumably requires larger or more complex scenarios to reveal its behavior, e.g., by adding more VNFs to the requested chains.

6 Related Work

This section provides an overview of publications that deal with different aspects of the VNFCP problem and focuses on the problem instances that are used for the quality assessment of the proposed algorithms. Furthermore, we present works that discuss methodologies for generating synthetic problem instances whose optimal solutions are known beforehand.

6.1 VNF Chain Placement

In [8], the authors provide one of the first formal problem statements for the VNFCP. Their evaluation is performed on a network comprised of four core routers, five switches, and 10 edge nodes. Since an ILP-based approach is utilized, optimal results are obtained. Unfortunately, such approaches can not be applied to large problem instances due to the fact that the solution space of the VNFCP grows exponentially.

For this reason, the authors of [9, 10] propose heuristics in addition to ILP-based algorithms. As they analyze only a subset of all possible placements, these heuristics are capable of handling large problem instances. As a downside, they provide no guarantees with regard to the optimality of returned solutions. In order to evaluate the heuristics, the authors generate synthetic topologies using the Barabási Albert (BA) model [11] with up to 1,000 nodes and randomly generated demands. In a similar fashion, both an ILP-based approach as well as a heuristic are proposed in [12]. However, only a small network of 12 nodes and 3 function chains is considered. In [3], a multi-objective approach is used to handle conflicting objectives. A set of solutions is continuously improved similarly to the simulated annealing approach. The results are evaluated by comparing them with those of another heuristic from literature, by means of three real-world topologies and artificial demands.

In contrast to optimizing the placement of all demands simultaneously, Bari et al. [2] propose an algorithm that adds newly arriving demands to the current placement and instantiates new instances on demand. The evaluation is performed on real world networks whose size ranges from 12 to 79 nodes. Similarly, Sahhaf et al. [13] consider the dynamic scenario and evaluate their algorithm using two network graphs from the Internet Topology Zoo [14].

Rather than addressing the entire VNFCP, [15,16] address subtasks like routing of demands or mapping and scheduling them to existing VNF instances, respectively. Both approaches work in the dynamic scenario. While the former uses different graph generation models like BA or Waxman [17], the latter does not require a topology due to the assumption that delays between nodes are negligible. Furthermore, demand arrivals follow a uniform distribution and are composed of random permutations of available VNF types.

In summary, most works in literature use either synthetic or real world graphs in conjunction with artificial demand sets in order to evaluate their proposed VNFCP heuristics. However, in the context of large problem instances, optimal solutions can not be determined and thus, a quantitative statement regarding the performance of heuristics is not possible.

Approximation Algorithms. A special case is provided by the few approximation algorithms in this context [18,19]. They prove deterministic bounds for their results and may therefore omit the empirical evaluation of their algorithms.

6.2 Synthetic Problem Generation

In many areas of optimization, synthetic problem instances are used in order to perform algorithm benchmarks and tweak their performance [20–22]. To the best of our knowledge, there are no corresponding frameworks for the VNFCP, yet. However, Virtual Network Embedding (VNE) problems [23, 24] overlap with the VNFCP in terms of the chaining and placement aspects, and algorithms for problem generation are researched in a recent publication [25]. Its authors develop mechanisms for both static and dynamic scenarios whose solution is known beforehand. Since it is possible to change the number of network nodes, requests, and resources, algorithms can be analyzed in terms of aspects like scalability and behavior under different load levels.

7 Conclusion

In order to fully benefit from the flexibility of Network Function Virtualization, algorithms that tackle the arising optimization problems, in particular the Virtual Network Function Chain Placement problem, are required. Despite the large number of recent publications in this area, there is no commonly accepted standard approach yet. Therefore, an unbiased methodology for the evaluation and comparison of VNFCP algorithms is necessary to obtain meaningful statements on their performance.

This work presents an overview of the VNFCP problem, followed by possible strategies for its solution. Different concepts towards their evaluation are discussed, and the importance of optimal solutions for their comparison is emphasized. In particular, an Integer Linear Program is proposed to obtain optimal reference solutions for small problems, along with artificial problem generation strategies with known optima for the assessment of larger topologies.

Two strategies are implemented to demonstrate the applicability of the concept. Their evaluation shows the influence of problem parameters on the algorithms' performance and provides valuable insights for their improvement. However, it also reveals current limitations. Hence, future work will include the extension of existing strategies to use more complex function chains, and the support for more objective functions. Nevertheless, by using artificial problems, the expressiveness of performance assessments can be raised significantly with regard to parameterization and absolute solution quality.

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. 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, pp. 50–56.
3. S. Lange, A. Grigorjew, T. Zinner, P. Tran-Gia, and M. Jarschel, "A multi-objective heuristic for the optimization of virtual network function chain placement," in *29th International Teletraffic Congress (ITC 29)*, 2017, pp. 152–160.
4. 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*, pp. 117–132, 2003.
5. J. Knowles, L. Thiele, and E. Zitzler, "A tutorial on the performance assessment of stochastic multiobjective optimizers," Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, Tech. Rep. 214, Feb 2006, revised version. [Online]. Available: http://www.tik.ee.ethz.ch/pisa/?page=bugs.php
6. 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*. ACM, 2009, pp. 87–102.
7. Zuse-Institute Berlin (ZIB), "SNDlib: Survivable fixed telecommunication network design library," http://sndlib.zib.de/, accessed: 2017-10-29.

8. 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, pp. 418–423.
9. 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, pp. 98–106.
10. 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*, vol. 102, pp. 67–77, 2017.
11. R. Albert and A.-L. Barabási, “Topology of evolving networks: local events and universality,” *Physical review letters*, vol. 85, no. 24, pp. 5234–5237, 2000.
12. S. Mehraghdam, M. Keller, and H. Karl, “Specifying and placing chains of virtual network functions,” in *3rd International Conference on Cloud Networking (CloudNet)*, 2014, pp. 7–13.
13. 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*, vol. 93, pp. 492–505, 2015.
14. S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, “The Internet Topology Zoo,” *IEEE JSAC*, vol. 29, no. 9, pp. 1765–1775, 2011.
15. 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, pp. 32–37.
16. 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.
17. B. M. Waxman, “Routing of multipoint connections,” *IEEE journal on selected areas in communications*, vol. 6, no. 9, pp. 1617–1622, 1988.
18. M. Rost and S. Schmid, “Service chain and virtual network embeddings: Approximations using randomized rounding,” *arXiv preprint arXiv:1604.02180*, 2016.
19. T. Lukovszki and S. Schmid, “Online admission control and embedding of service chains,” in *International Colloquium on Structural Information and Communication Complexity*. Springer, 2015, pp. 104–118.
20. K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, *Scalable test problems for evolutionary multiobjective optimization*. Springer, 2005.
21. S. Huband, L. Barone, R. L. While, and P. Hingston, “A scalable multi-objective test problem toolkit.” in *EMO*, vol. 3410, 2005, pp. 280–295.
22. S. Huband, P. Hingston, L. Barone, and L. While, “A review of multiobjective test problems and a scalable test problem toolkit,” *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 477–506, 2006.
23. A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, “Virtual network embedding: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.
24. I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, “Vnr algorithm: A greedy approach for virtual networks reconfigurations,” in *Global Telecommunications Conference (GLOBECOM)*, 2011, pp. 1–6.
25. A. Fischer and H. de Meer, “Generating virtual network embedding problems with guaranteed solutions,” *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 504–517, 2016.