NSPlacer: A Tool for Evaluating Service Placement Algorithms in Post-5G Networks

Masoud Taghavian, Yassine Hadjadj-Aoul, Géraldine Texier, Philippe Bertin

▶ **To cite this version:**

Masoud Taghavian, Yassine Hadjadj-Aoul, Géraldine Texier, Philippe Bertin. NSPlacer: A Tool for Evaluating Service Placement Algorithms in Post-5G Networks. CloudNet 2022 - IEEE 11th International Conference on Cloud Networking, Nov 2022, Paris, France. pp.252-256, 10.1109/CloudNet55617.2022.9978830 . hal-03911372v2

HAL Id: hal-03911372
https://inria.hal.science/hal-03911372v2

Submitted on 7 Feb 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

NSPlacer: A Tool for Evaluating Service Placement Algorithms in Post-5G Networks

Masoud Taghavian*†, Yassine Hadjadj-Aoul*‡, Géraldine Texier *†, Philippe Bertin*§
*IRT-BCOM, France; firstname.lastname@b-com.com
†IMT Atlantique/IRISA/Adopnet, France; firstname.lastname@imt-atlantique.fr
‡University of Rennes, Inria, CNRS, IRISA, France; firstname.lastname@irisa.fr
§Orange, France; firstname.lastname@orange.com

***Abstract*—Network Function Virtualization (NFV) has been a prominent shift from dedicated network devices towards reusable Virtual Network Functions (VNF). Today's network services are defined as a graph of VNFs connected via Virtual Links (VL), which are realized by placing its elements on their corresponding substrate network nodes and links. Efficient placement of the network services is one of the most important challenges introduced by NFV. We propose a demonstration of the "Network Service Placer (*NSPlacer*)", a tool to enable the evaluation of the placement algorithms. By being able to adjust various parameters, related to the substrate network, the service and the placement algorithm, and thanks to the already implemented interesting placement algorithms, our online tool can provide near-optimal and optimal results to serve as a comparison with many placement solutions. Made available to the community, our tool aims to contribute to the reproducibility of the research results by offering a set of reference algorithms to which everyone can compare their solution and even integrate it into the set of available algorithms of the tool.**

***Index Terms*—Network function virtualization, service placement, online tool.**

I. Introduction

The use of cloud computing technologies to virtualize and orchestrate network functions is no longer an option in modern infrastructures, hence the adoption of Network Function Virtualisation (NFV) in today's networks. The placement of services is one of the most important steps in NFV. It mainly consists in addressing the allocation of network resources required by the services, a problem proven to be NP-Hard in terms of complexity [1].

Placement algorithms have been studied for several years. The goal of online placement is to find a suitable placement (by optimizing an objective function, such as the minimization of the resources' consumption) for a single service request, as soon as it arrives, in a limited time. In the literature, placement algorithms range from the ILP formulations like [2] to evolutionary algorithms like [3], and heuristic methods. Using sophisticated Artificial Intelligence (AI) and Machine Learning (ML) techniques like in [4] have been gaining a lot of attention recently and they seem to be highly promising.

In [5], we proposed an online placement approach, based on Branch and Bound (BnB), allowing to apply various AI search strategies (especially A*). We demonstrated how to obtain near-optimal and optimal results with a high degree of scalability. Inspecting the effectiveness of our approach, we have struggled to find baseline solutions to which we could compare our results. We built the *NSPlacer* tool, to be served as the mentioned baseline solution, and we made extensive evaluations to inspect the effectiveness of our proposed approach.

In this paper, we present our *NSPlacer* tool (accessible online on https://nsplacer.labs.b-com.com/), which we developed for making evaluations and comparisons between placement algorithms (including our approach proposed in [5]). Since the placement problem is encountered in many areas related to NFV, *NSPlacer* may facilitate its investigation for the research community. By being able to adjust the related parameters of the substrate network, the service requests, and the placement algorithms, our online tool provides results not only to see how the topology, node/link resources and Quality of Service (QoS) requirements influence the placement problem, but also to be used as a comparison among different placement solutions. In addition, it provides an infrastructure for the researchers to evaluate their own placement solutions. As long as the parameters are compatible, the researchers can compare their results with the results obtained from our tool manually, or directly connect the implementation of their placement algorithm to our tool in order to provide the results. In order to make the evaluations between different placement algorithms more convenient, we provide a network-based web-socket API which allows a custom placement algorithm to be connected over the network regardless of its programming language [1]. We decided to make our *NSPlacer* tool public, offering the community a common tool to which they could compare their solutions. We believe that this can improve the reproducibility of the research results by providing a common environment for evaluating and comparing the algorithms.

The demo paper is organized as follows. Section II introduces the problem of services' placement. Section III provides an understanding of the architectural solution

[1]For more information on how to connect to our tool please visit the NSPlacer-Connect section on our tool

and describes how the tool performs the placements, while Section IV gives a brief overview of the problem' parameters as well as the supported placement algorithms. Section V, provides an idea of the outcomes that can be achieved using the proposed platform. Finally, Section VI concludes this demo paper and gives perspective on future developments of our solution.

II. Placement Problem

The physical network, also called substrate network, is defined as a graph of nodes (representing the servers) and links (representing the connections), along with a set of available resources for both nodes and links and QoS metrics. Similarly, a service is defined as a graph of Virtual Network Functions (VNFs) and Virtual Links (VLs), requiring a subset of the resources and subject to QoS constraints. The placement problem deals with finding a mapping of the VNFs and the VLs of a service onto the network nodes and the paths (the sequence of network links), by satisfying the corresponding constraints (Figure 1). Note that the routing of the VLs depends on the placement, and vice versa, so we need to consider them jointly. To distinguish between different feasible placement candidates of a service over a network, we need a criteria or an objective. Service Acceptance (SA) is one of the most popular among the various studied objectives in the literature. It is defined as the maximum number of the services that can be placed successfully over the network, and it is considered as a practical generalization of resources-based multi-objective placement. We are interested in SA and we created the "Network Service Placer (*NSPlacer*)" tool, to be able to evaluate it for different placement algorithms.

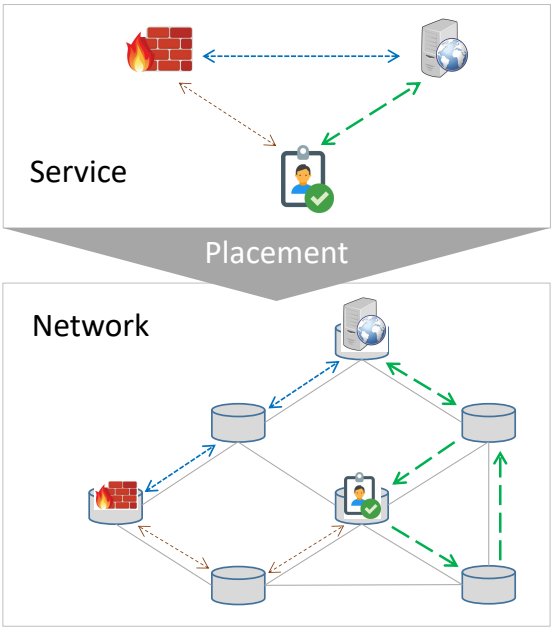

Figure 1. Placement of a service request over the network

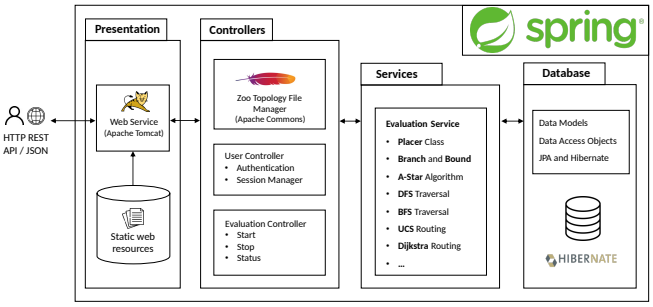

Figure 2. General architecture

III. Tool architecture and processing

Figure 2 presents a brief overview of the architecture of our tool, with the different layers and technologies considered. The user interacts with the tool's web interface via the standard REST API using JSON as the data exchange format. Our tool conforms to the general architecture of the Spring [6] framework, which decomposes the architecture into several layers, including presentation, controllers, services and data (data model and data access).

The evaluations performed by our tool start by initializing the set of nodes and links in the network with the maximum available resources, and then attempt to place services iteratively. In each iteration, a service request is created based on the specified input parameters. Then, an attempt is made to place the service on the network using the selected algorithm within the given time frame. If the algorithm successfully finds a placement, the placement is applied to the network by updating the remaining resources, before starting a new iteration. Otherwise, it completes the evaluation and reports the results. The results are shown at the end of the evaluation (Figure 3f). They include the number of services placed, the execution time to place each service request in milliseconds, and the overall amount of resources remaining (CPU, storage, and bandwidth) at the end of the evaluation in percent. The exact placements of each service request are provided in the results section which can be easily exported.

IV. Input/Output Parameters

In order to evaluate the influence of the inputs of the problem, the tool allows to choose the values of a large number of parameters classified in 3 categories.

1) Substrate Network: It is the physical network hosting the network services. Our tool accepts network topologies represented in the *Zoo Topology* [7] XML file format. New topologies can be downloaded from the zoo topology dataset, or defined as an XML file and uploaded into our tool. A graph visualization section is provided for the network topologies in order to be able to check the connectivity of the network visually. Currently, we consider *CPU* and *storage* as two resource types for each network node, along with the *bandwidth* and the *latency* for the network links (although latency is considered as a QoS metric,

we use the term resource for unification). The Figure 3a shows the interface for setting the substrate network's parameters. At the beginning of the evaluation, these input parameters represent the initial capacity (maximum amount) of the resources for each network node and link.

2) Service Graph: It specifies the service request to be placed over the network. We propose different types of topologies, including Daisy-Chain, Ring and Star. The input parameter *size* specifies the number of VNFs. Each VNF requires several units of available resources from a network node, specified by the input parameters *CPU* and *storage*. Likewise, VLs require available resources from a network link defined by the input parameters *bandwidth* and *latency*. The interface is shown in Figure 3b.

3) Placer: Our tool offers a set of input parameters to evaluate the algorithms as shown in Figure 3c. In an online placement, we place one service at a time, as soon as it arrives, but within a *Timeout* for performing the placement. The *Termination* input parameter specifies whether the algorithm should search for better placements (*Best-Found*) until the timeout is reached or it should terminate as soon as it finds a first placement (*First-Found*). The parameter *Shuffle* is used to present randomness in the service placement algorithm. The *Routing* specifies the routing algorithm, either Dijkstra algorithm or Uniform Cost Search (UCS), to calculate the shortest paths. Last but not least, the input parameter *Algorithm* chooses the placement algorithm. These are either the predefined algorithms or the connected custom placement algorithms. It is worth noting some of the predefined placement algorithms.

- *A* Bandwidth Optimized (ABO):* ABO is one of the most interesting placement algorithms that we proposed, based on A* search algorithm over BnB to find optimal placement candidates by concentrating on the bandwidth usage optimization. BnB is one of the most popular algorithm design paradigms used for solving problems with exponential complexity. BnB attracted our attention because of the large number of constraints inherent to this type of placement problem, which could be beneficial in a BnB context (the more constraints we have, the more we can prune the tree, leading to a faster search procedure). First, we formulate the problem over a BnB structure, then we start a search procedure for finding a solution, which is a complete placement of a service request over the network. We traverse the search tree, according to an order determined by a search strategy. A* is amongst the most popular AI informed search algorithms due to its completeness, optimality, and efficiency [8]. More details and the proof of the optimality of the ABO are provided in [5].
- *Fair A* Bandwidth Optimized (FABO):* Along with the optimality of ABO, increasing the chances of accepting the new service requests by avoiding network defragmentation is another important consideration. FABO is defined on top of ABO trying to address this issue by performing an optimal placement considering a fair distribution of the load over the network links.
- *DFS Bandwidth Optimized (DBO):* DBO is another placement algorithm based on a Depth-First Search (DFS)-traversal over BnB in order to quickly reach a near-optimal placement.
- *Best-Fit (BF) and Enhanced Best-Fit (EBF):* BF is one of the most well-known heuristic for service placement. By adapting BF over BnB, we could obtain other placement algorithm called EBF, which improves the results of BF remarkably.
- *Parallel:* Parallel does not define a specific algorithm, but it executes FABO, ABO and DBO in parallel, and integrates them to obtain the best results.

Table I
Evaluation results of placing services over BT-Europe

Algorithm	Services(#)	Time(ms)	Remained BW(%)
FABO	120	435	2.70
ABO	110	13	10
DBO	113	8	1.62
EBF	58	10	4.05
BF	26	9	59.59

V. Example of the use of the tool's output

To demonstrate a use-case of the tool and make comparisons between several placement algorithms, we conducted several evaluations, and the results are summarized in Table I. Let's try to place service requests having 4 VNFs connected in Daisy-Chain topology (each VNF demanding 1 unit of CPU and 1 unit of storage, and each VL demanding 1 unit of bandwidth) over the network with Bt-Europe topology. We considered 1000 units of CPU and storage available for network nodes, and 10 units of bandwidth available for network links. We want to evaluate different placement algorithms. To do so, we set the input parameters, start the evaluation and collect the results (see figures 3e and 3f) for each execution and add the corresponding row in the table reftable:ERPS. Based on these results, we were able to show that FABO could place the highest number of services over the network but required considerably more time on average than the other algorithms. Although DBO could place a little more services than ABO, ABO could preserve much more remaining bandwidth at the end of the evaluation. In ABO, placed services uses less bandwidth on average than DBO, since ABO is guaranteed to be optimal, which is not the case for DBO. Although BF algorithm might perform well when we do not have significant constraints on bandwidth, it is obviously not a suitable algorithm when we have serious bandwidth constraints.

VI. Conclusions and future work

In this demo, we presented our online tool "Network Service Placer", as infrastructure for the research community to make evaluations with their own or predefined

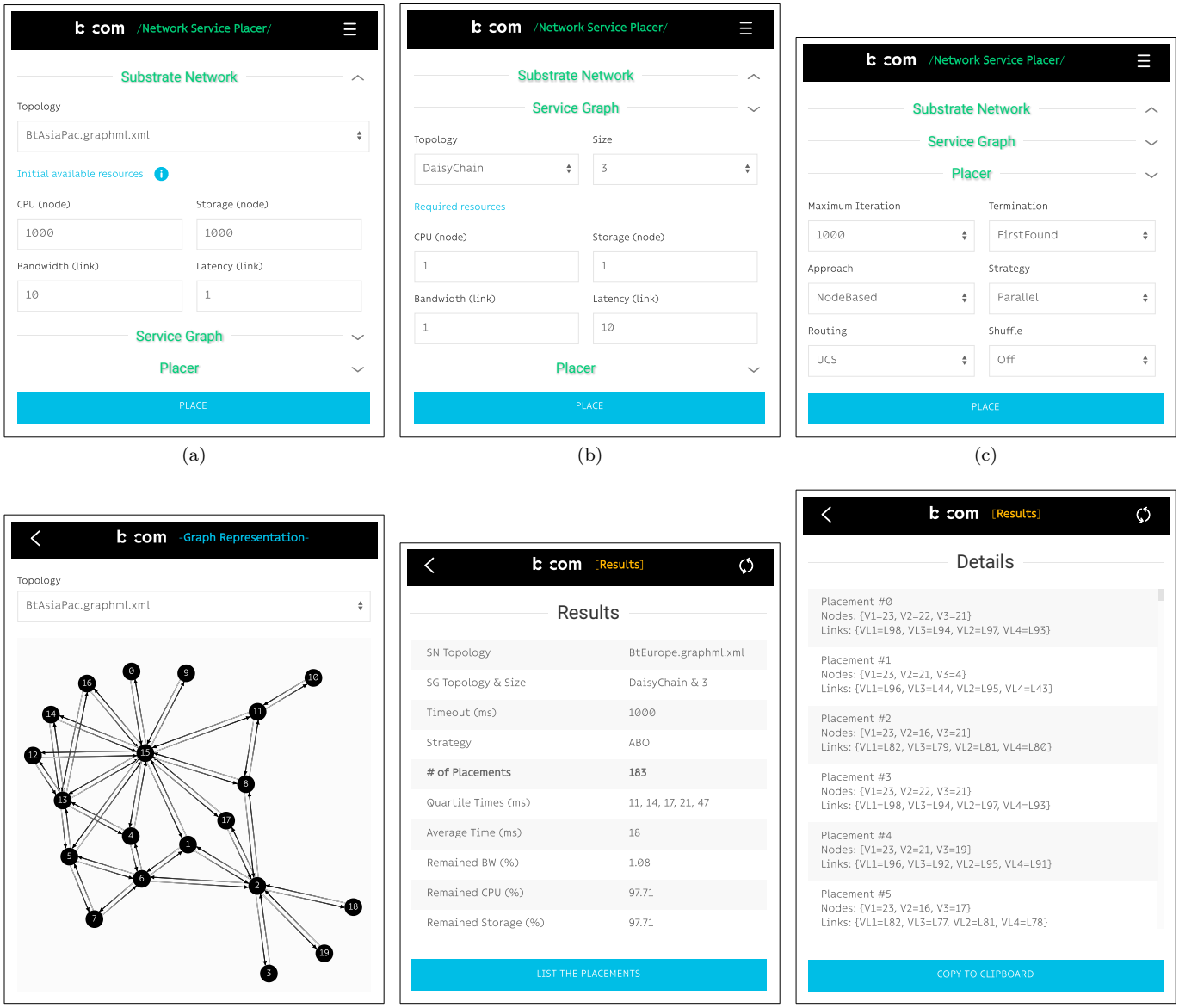

Figure 3. Some screenshots ((a) substrate network parameters, (b) service graph parameters, (c) placer parameters, (d) network graph representation, (e) overall results, and (f) placement details

placement algorithms, in order to not only facilitate comparing placement algorithms but also to be able to track the influences of the topology, resources and QoS metrics on the placement algorithms. We believe that making our tool available to the community can contribute to the reproducibility of research results. Researchers will be able to compare their own solutions to common algorithms, or even integrate them into our tool to allow other researchers to evaluate theirs in return.

In the future, we envision the possibility of performing a batch of evaluations by defining a range or random generator to randomly generate values for some input parameters. Currently, we assume that all VNFs in a service are instantiated and cannot be shared across services for simplicity. We plan to introduce this possibility in an update of our tool.

References

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

[2] Marcelo Caggiani Luizelli, Leonardo Richter et Bays, Luciana Salete Buriol, Marinho Pilla Barcellos, and Luciano Paschoal Gaspary. Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions. In *2015 IFIP/IEEE Int. Symp. on Int. Network Management (IM)*, pages 98–106. IEEE, 2015.

[3] T.A.Q. Pham, Jean-Michel Sanner, Cédric Morin, and Yassine Hadjadj-Aoul. Virtual network function–forwarding graph embedding: A genetic algorithm approach. *Int. J. of Communication Systems*, 33(10):e4098, 2020. e4098 0.1002/dac.4098.

[4] Pham Tran Anh Quang, Yassine Hadjadj-Aoul, and Abdelkader Outtagarts. A deep reinforcement learning approach for vnf forwarding graph embedding. *IEEE Transactions on Network and Service Management*, 16(4):1318–1331, 2019.

[5] Masoud Taghavian, Yassine Hadjadj-Aoul, Géraldine Texier, and Philippe Bertin. An approach to network service placement using intelligent search strategies over branch-and-bound. In *2021 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2021.

[6] Rod Johnson, Juergen Hoeller, Alef Arendsen, and R Thomas. *Professional Java development with the Spring framework*. John Wiley & Sons, 2009.

[7] Simon Knight, Hung X. Nguyen, Nickolas Falkner, Rhys Bowden, and Matthew Roughan. Zoo topology. http://www.topology-zoo.org/.

[8] Stuart Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Prentice Hall, 2002.