A GRASP-based algorithm for Virtual Network Embedding

Amine Rguez, Yassine Hadjadj-Aoul, Farah Slim, Gerardo Rubino, Asma Selmi

▶ **To cite this version:**

Amine Rguez, Yassine Hadjadj-Aoul, Farah Slim, Gerardo Rubino, Asma Selmi. A GRASP-based algorithm for Virtual Network Embedding. ISCC 2023 - IEEE Symposium on Computers and Communications, Jul 2023, Gammarth, Tunisia. pp.470-473, 10.1109/ISCC58397.2023.10218024 . hal-04368546

HAL Id: hal-04368546
https://inria.hal.science/hal-04368546

Submitted on 1 Jan 2024

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

A GRASP-based algorithm for Virtual Network Embedding

Amine Rguez*†, Yassine Hadjadj-Aoul†, Farah Slim*, Gerardo Rubino†, and Asma Selmi*,
*EXFO Solutions, Rennes, France
†Univ Rennes, Inria, CNRS, IRISA, France

***Abstract*—With the rise of network virtualization, network slicing is becoming a hot research topic. Indeed, network operators must deal with capacity-limited resources while insuring an extreme availability of services. Several approaches exist in the literature to tackle such a problem, some of them converge quickly to a local minimum, while others are not explainable and therefore do not provide the necessary guarantees for their deployment in a real network. In this context, we propose a new approach for Virtual Network Embedding (VNE) based on the Greedy Adaptive Search Procedure (GRASP). Using the GRASP meta-heuristic ensures the robustness of the solution to changing constraints and environments. Moreover, the proposed realistic approach allows a more efficient and directed exploration of the solution space, in opposition to existing techniques. The simulation results show the potential of the proposed method for solving services' placement problems and its superiority over existing approaches.**

***Index Terms*—Virtual Network Embedding, GRASP algorithm, Cloud Resource Allocation, Network Slicing.**

I. Introduction

While traditional networks have only supported a very limited number of services, this number has increased with the advent of 5G and the evolution is towards more and more complexity for the next mobile network systems. To cope with this complexity, network slicing has been introduced in 5G, although it will probably gain significance in upcoming mobile standards where massive slicing is expected.

Network slicing, which, at first glance, aims to guarantee the operating constraints of services, represents one of the most important technological building blocks to converge towards a fully automated network (i.e. Zero Touch networks) [1]. It brings several challenges, one of the most important being the placement of services. In terms of algorithms, service placement consists of embedding constrained services, often represented by graphs, on an underlying network, which is itself represented by a graph [2].

In this context, we present a new approach to services' placement. We specifically target the optimization of the Virtual Network Embedding (VNE) [3]. We define some traditional constraints like those related to system or network resources. We also define some other production-level constraints like the need for Single-root input/output virtualization (SRIOV) cards or Huge Page enabled memory. Our approach is based on the Greedy Randomized Adaptive Search Procedure (GRASP) algorithm [4]. This meta-heuristic is able to explore only viable solutions, which often leads to a pretty efficient optimization procedure.

The remainder of the paper is organized as follows. Next section discusses the related work. Section III introduces the problem of resource allocation in the cloud and the proposed algorithm. We focus especially on the main functions of our approach. Then, in Section IV, the simulation results of the proposed method are discussed. We compare its performance against those obtained with an enhanced First-Fit heuristic. Finally, our conclusions are presented in Section V.

II. Related work

Many research works addressed the problem of resource allocation and placement of network functions. We can group them into two different classes.

A. Mono-objective Approaches

The first category of approaches is the mono-objective modeling of the problem of resource allocation, that is, the fact that there is a single optimization objective.

Integer Linear Programming (ILP) techniques are widely used in this case and proved to be very effective for providing accurate decisions as it relies on mathematical models [5]. However, it suffers from heavy run-time.

Heuristics are certainly the most used techniques in both research and industry. For instance, see [6], where the authors proposed a refinement framework for profiling micro-services placements using a greedy heuristic. In [7], the authors presented an approach to optimize the placement of service-based applications in clouds. In their approach, they first partition the application into several parts while keeping overall traffic between different parts to a minimum and then pack the different parts into machines according to their resource and traffic demands. Although heuristics could be applied in a realistic context as they don't require a high computational power, they may stack at a local optimum and as a result, they may have difficulties in providing a good performance.

Finally, we have Artificial Intelligence-based techniques. In this class, we are mainly focusing on the adoption of Reinforcement Learning for the optimization of resource allocation designs. In [8], a placement solution based on Deep Reinforcement Learning (DRL) was introduced and was combined with a dedicated optimization heuristic. The authors' aim was to accelerate the learning process and gain in resource

usage when compared to other state-of-the-art approaches. This technique is proven to perfectly work with multi-objective goals but it's not easy to adopt it at an industrial level, as the performance of the solution is not guaranteed.

B. Multi-objective Approaches

In [9], an heuristic and a machine learning-based VM allocation method were proposed. The authors present a novel hyper-heuristic algorithm that exploits the benefits of both methods by dynamically finding the best one, according to a user-defined metric.

To easily adapt to various objectives effectively, Non Dominated Sorting Genetic Algorithm II (NSGA-II) was used in many research papers, such as [10]. In [11], a genetic algorithm was also introduced in order to improve the performance of micro-service-based applications in Edge Computing. Firstly, by preventing performance degradation caused by resource contention. Secondly, by increasing the application's availability as a means of avoiding SLA violations.

To summarize, Genetic Algorithms are very powerful for multi-objective optimization problems. But, the stopping criterion is not clear, the exploration may not be very efficient (i.e., exploration of non-viable solutions), and in some cases, it may suffer from premature convergence.

III. GRASP-based algorithm for efficient placement of VNFs

A. Problem Statement

The main problem we are concerned with in this paper is the placement of a batch of Virtual Network Requests (VNRs) on a substrate network. Each VNR is composed of Virtual Network Functions (VNFs) with CPU, RAM, and Storage demands. Some of these VNFs also have placement constraints such as enabling the Huge Page flag or the fact that they must be placed on a host that contains a specific SRIOV (Single Root Input/Output Virtualization) network interface card. These VNFs are also interconnected by links with bandwidth demands.

In addition, the substrate network has a predefined number of nodes, each one having limited resources in terms of CPU, RAM, and Storage. It might also have the Huge Page flag enabled and might have an SRIOV card integrated.

In the VNE problem, we cannot place two VNFs of the same VNR in the same host. To deploy a VNR, all of its VNFs and links between them need to be placed.

To address this problem, we propose to use the GRASP meta-heuristic, whose principles are recalled in the sequel.

B. GRASP Algorithm Overview

The Greedy Randomized Adaptive Search Procedure (also known as GRASP) is a meta-heuristic algorithm commonly applied to combinatorial optimization problems [4]. GRASP typically consists of iterations made up of successive constructions of a greedy randomized solution and its subsequent iterative improvements using a local Search.

The greedy randomized solutions are generated by adding elements to the problem's solution set from a list of possibilities ranked by a greedy function according to the quality of the solution they will achieve. To obtain variability in the candidate set of greedy solutions, well-ranked candidate elements are often placed in a restricted candidate list (RCL), and chosen at random (uniformly) when building up the solution.

The power of this technique relies on its exploration space of possible solutions. It has a parameter called Exploration Factor defined by the user. Thus, it will consider many solutions and it will try to find the best one. Here, only viable solutions are considered, unlike genetic algorithms in which the exploration is much larger. Compared to other heuristics like First-Fit, which places the services on the first node satisfying the resources' requirements, a fitness function is used, in order to choose among a list of possibilities (candidates).

C. Proposed Algorithm Principles

Unlike a genetic algorithm which is generic enough to tackle any problem, GRASP requires an important adaptation effort because the construction of the solution requires a precise knowledge of the problem to be solved.

To better explain our approach, we present the proposed algorithm functioning in detail. As explained in Algorithm 1, a GRASP algorithm contains mainly two steps. The first one is to construct a solution. This greedily constructed solution is the input for the second step, which will try to find a better solution in the neighborhood of the constructed solution, based on some criteria. In our case, we define revenue to cost R/C as the choice criteria of the best solution. Finally, system and network resources are updated on the substrate node.

Algorithm 1 GRASP Algorithm for VNRs placement

```
1: function GRASP(SN, VNR, λ)
2:     NodesPlaced ← False
3:     LinksPlaced ← False
4:     bCost ← 0
5:     for k ← 0 to nbNodes do
6:         valid, sol, cost ← solConstruction(λ, VNF)
7:         if valid then
8:             NodesPlaced ← True
9:             LinksPlaced ← True
10:        else
11:            return {}, bCost
12:        end if
13:    end for
14:    valid, bCost, bSol ← LocalSearch(VNR, sol, cost)
15:    if valid then
16:        updateResources(bSol)
17:    end if
18:    return bSol, bCost
19: end function
```

The construction phase of the GRASP algorithm, starts by choosing randomly a VNF from the VNE. Then, it filters the

nodes of the substrate nodes regarding the constraints of this VNF. After that, it randomly chooses a feasible host node because we can't calculate the R/C for the first VNF.

To place the second VNF, we do the same as for the first, but here we have to compute the R/C for all the candidate solutions. This R/C rate is calculated based on the placement of the first VNF.

So, we have a list of candidate solutions C with their costs $c(.)$. Each solution e with its cost that satisfies the following equation is placed in the RCL.

$$c(e) \geq C_{\text{max}} + \lambda(C_{\text{min}} - C_{\text{max}}), \text{ with } \lambda \in [0, 1].$$

When the greediness value $\lambda = 0$, the solution is purely greedy (as we are selecting only the best solutions with cost C_{max}) and if $\lambda = 1$, the solution is purely random (as we are selecting all the solutions, since C_{min} is the lowest cost solution).

Finally, the Construction Phase function returns a randomly chosen solution from the RCL. The Local Search phase of the GRASP algorithm starts by filtering the directly connected nodes with the substrate node of the constructed solution. Then, it calculates the placement cost for each candidate solution. If it can't find a better solution, the first constructed solution is returned. After that, the function returns the best solution with the maximum R/C and its cost.

This process can be repeated several times to find better solutions. Indeed, the random aspect of the algorithm means that each iteration explores new solutions.

IV. Performance Evaluation of the GRASP-based Algorithm

A. Simulation Settings

We assume a placement request for a batch of services. Each simulation will run the placement of 10 services and then will increase the number of services until reaching 200 requests. We considered only 5 iterations, and the different experiences were repeated 20 times to build confidence intervals.

The demands of a request in system and network resources are randomly generated. The special taints of Sriov and Huge Page are generated with a fixed probability of 0.3. Also, each node of the substrate network would have these taints randomly, with the same probability.

The system and network capacities were the same during all the simulations. In fact, our goal here is not to compare the approaches in different load conditions.

To run the simulations and judge the performance of our approach, we developed our own *python* simulator. Besides, to save room, we show only the results for the BtEurope[1] topology which contains 24 nodes interconnected by 37 edges. The results obtained for other network topologies of the same nature were very similar.

The experiments were made with the same graph configurations and the same capacity constraints in terms of CPU, RAM, Storage and Bandwidth (BW).

[1]http://www.bt.net/info/europe.shtml

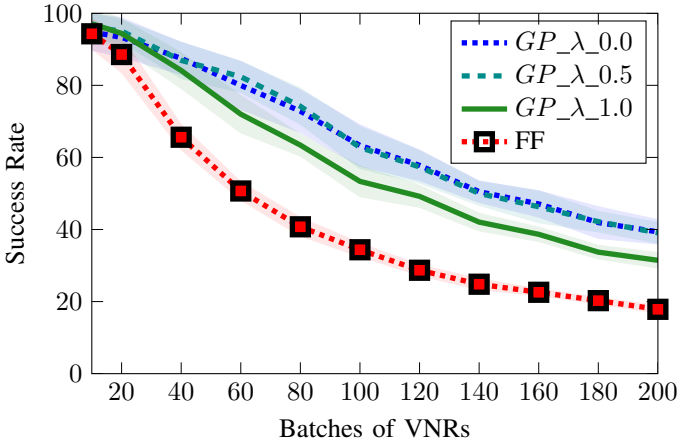

Fig. 1. GRASP performance vs First Fit with 3 VNFs

We compared our algorithm with the First-Fit heuristic in four different cases when VNRs are composed of 3, 4, 5, and 6 VNFs. We wanted to see the impact of the complexity of the request on the placement decisions made by the algorithm. Our First-Fit algorithm, unlike other standalone First-Fit algorithms, chooses a solution after filtering the nodes of the substrate network. It doesn't explore non-viable solutions. Thus, it will provide significantly better results than the original algorithm.

B. Simulation results

After running the simulations, we computed confidence intervals for all the results and added them to the plots.

1) Success rate comparison: First we compare the simulation results in terms of placement Success Rate (or acceptance rate) of the GRASP-based algorithm with greediness values $\lambda = \{0.0, 0.5, 1.0\}$ against First-Fit. It's the number of successfully placed services for a batch of generated virtual network requests over the total number of generated services to be placed.

Figure 1 shows that our algorithm performs better with VNRs containing 3 VNFs in terms of acceptance rate. We can see that the greediness value λ has only a slight impact on the performance of our algorithm.

Finally, Figure 2 shows a better illustration of the domination of our GRASP-based solution with its different values of λ against the modified First-Fit heuristic. Obviously, our approach gains especially when dealing with more complex VNRs with 5 or 6 VNFs.

2) R/C rate comparison: To further evaluate the performance of our algorithm, we compare the variance of the Revenue to Cost metric against the number of services to place. The R/C ratio is defined as R/C $= requiredBW/usedBW$. It is calculated based on the network resources' consumption when the VNR is successfully placed. It means that it computes the VNR request in bandwidth over the actually used bandwidth when it is placed. So, a perfect placement of a virtual link will use one substrate link. Making a maximum and optimal value for R/C = 1.

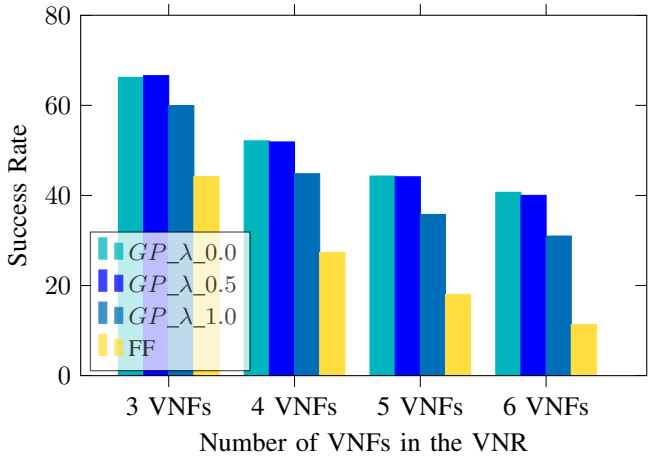

Fig. 2. Global comparison for different sizes of VNRs

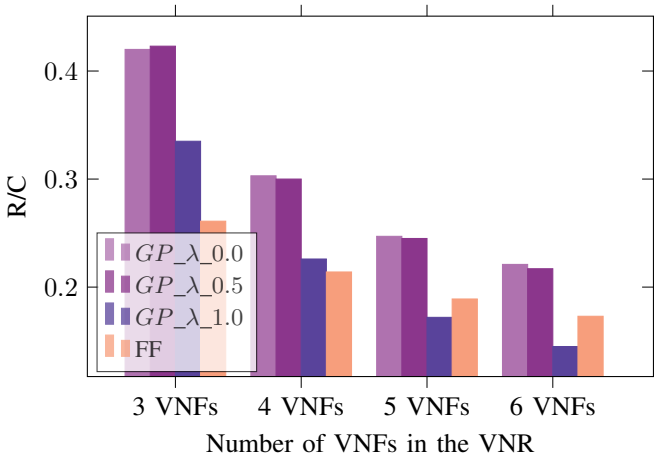

Fig. 3. R/C Variance for GRASP vs First Fit

For example, a link between two VNFs of the VNR requires 10 in terms of BW capacity. Thus, it can be placed on the substrate network over two links, either because we don't have a direct link between the two host nodes or because we don't have enough capacity on the direct link. So, it will consume bandwidth from two links instead of one. This may result in the placement failure of the next VNR.

To summarize, Figure 3 shows a better illustration of the domination of the GRASP algorithm with its different values of λ against the First-Fit heuristic, in terms of R/C. In fact, the greater the number of services to place, the larger is the gap between the placed solutions' cost with grasp and those with First-Fit.

In our approach, the greediness value λ didn't have much impact on the results. Despite the existence of randomness in the choice of the next VNF to be placed and the first substrate node, we must say that this is a totally greedy approach. That's because the construction of the solution is greedy as we try to optimize the Revenue to Cost metric when moving to the next VNF to place.

V. Conclusions

The problem of network slicing, and more specifically, Virtual Network Embedding, or VNE, remains a very active topic in 5G and post-5G networks. In this paper, we proposed a GRASP-based algorithm for VNE, and an enhanced First-Fit strategy. These two algorithms were applied after a filtering step, allowing to reduce the exploration space. This last step is inspired by the functioning of an orchestrator like Kubernetes (with a filtering step, then scoring). The results show that our algorithm largely dominates the enhanced First-Fit heuristic in terms of R/C and in terms of acceptance rate.

The algorithm introduced in this paper can be improved by enhancing its construction phase by means of a deep learning strategy or by implementing a new heuristic for the restricted candidate list of solutions. This will be explored in future work.

References

[1] C. Benzaid and T. Taleb, "Ai-driven zero touch network and service management in 5g and beyond: Challenges and research directions," *IEEE Network*, vol. 34, no. 2, pp. 186–194, 2020.

[2] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[3] 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.

[4] T. A. Feo and M. G. Resende, "Greedy randomized adaptive search procedures," *Journal of global optimization*, vol. 6, no. 2, pp. 109–133, 1995.

[5] I. Allal, K. Dhifallah, J. Penhoat, Y. Gourhant, and S.-M. Senouci, "A green mesh routers' placement to ensure small cells backhauling in 5g networks," in *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, 2017, pp. 1–6.

[6] F. Slim, F. Guillemin, A. Gravey, and Y. Hadjadj-Aoul, "Towards a dynamic adaptive placement of virtual network functions under onap," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2017, pp. 210–215.

[7] M. Selimi, L. Cerdà-Alabern, M. Sánchez-Artigas, F. Freitag, and L. Veiga, "Practical service placement approach for microservices architecture," in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, 2017, pp. 401–410.

[8] Z. Yang, P. Nguyen, H. Jin, and K. Nahrstedt, "Miras: Model-based reinforcement learning for microservice resource allocation over scientific workflows," in *2019 IEEE 39th international conference on distributed computing systems (ICDCS)*. IEEE, 2019, pp. 122–132.

[9] E. Dávalos, C. Aceval, V. Franco, and B. Barán, "A multi-objective approach for vne: Problems using multiple ilp formulations," *CLEI Electronic Journal*, vol. 19, no. 2, pp. 2–2, 2016.

[10] C. Guerrero, I. Lera, and C. Juiz, "Resource optimization of container orchestration: a case study in multi-cloud microservices-based applications," *The Journal of Supercomputing*, vol. 74, no. 7, pp. 2956–2983, 2018.

[11] T. A. Q. Pham, J.-M. Sanner, C. Morin, and Y. Hadjadj-Aoul, "Virtual network function–forwarding graph embedding: A genetic algorithm approach," *International Journal of Communication Systems*, vol. 33, no. 10, p. e4098, 2020.