A Reinforcement-Learning Self-Healing Approach for Virtual Network Function Placement

Marios Avgeris
Dept. of Systems and Computer Eng.
Carleton University
Ottawa, Canada
MariosAvgeris@cunet.carleton.ca

Aris Leivadeas
Dept. of Software Engineering and IT
École de Technologie Supérieure (ÉTS)
Montréal, Canada
aris.leivadeas@etsmtl.ca

Ioannis Lambadaris
Dept. of Systems and Computer Eng.
Carleton University
Ottawa, Canada
ioannis@sce.carleton.ca

***Abstract*—Modern networking paradigms like Service Function Chaining (SFC) allow for services to be broken down to a series of ordered and interconnected Virtualized Network Functions (VNFs) that can be hosted in generic servers in Edge-Cloud datacenters. Nonetheless, a critical issue arises, when a hardware or software failure occurs and the VNFs of an SFC need to be repositioned, allowing to autonomously bring the system back to its normal operation, a process called self-healing. In this paper, a distributed methodology is proposed that aims to address this challenge, considering the requirements of all involved actors. Specifically, a Reinforcement Learning (RL) based algorithm is proposed that allows to iteratively optimize and determine an SFC healing solution upon a datacenter failure. As a second stage, a revenue-driven resource allocation mechanism is integrated, to resolve the contention for resources in an already functional datacenter that potentially occurs due to the repositioning. Various simulation scenarios prove the efficiency of our proposed resilient healing mechanism.**

***Index Terms*—SFC, Optimization, Self-Healing, Edge/Cloud Computing, Reinforcement Learning, Resource Allocation.**

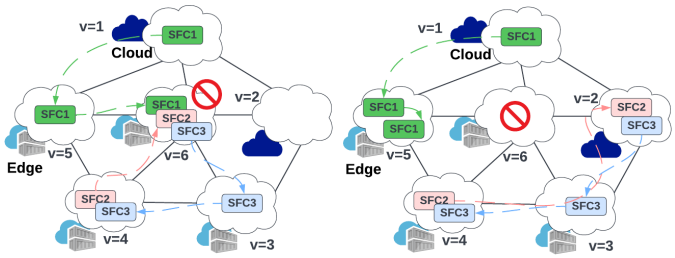
I. Introduction

The new and upcoming 5G and IoT networks make the Quality of Service (QoS) and Experience (QoE) requirements stricter, while the densification of applications and devices has resulted in an unprecedented need for automatic network service delivery. Thus, modern networks should be able to be flexibly adapted and adjusted according to the types of services being requested and the requirements of the end-users. To this end, Network Function Virtualization (NFV) and Service Function Chaining (SFC) create a new network paradigm that allows the flexible deployment and configuration of network services [1], while through appropriate orchestration mechanisms [2] can guarantee the automatic lifecycle management of network services. An open challenge in this automatic management is how network services deployed as SFCs can be quickly and efficiently self-healed in case of a network failure. This is particularly important since network outages, which occur with a probability that lies between 60-99.8% [3], can affect the user experience and adversely impact the service continuity. This paves the way towards new self-healing techniques complemented with telemetry, failure detection and resolution capabilities [4].

The SFC self-healing has been recently explored in the pertinent literature. Apart from SELFNET [4] which focuses on predicting network failures, other management frameworks that enable autonomic network functionalities have been proposed, such as in [5] and [6]; runtime SFC traffic rerouting and VNF dynamic redeployment are proposed there respectively. To theoretically formulate the placement and SFC Self-Healing problems, Reinforcement Learning (RL) approaches have been proposed; in [7], the authors deploy a deep learning-based framework with the goal to minimize the weighted cost that reflects the efficiency of the solution, the deployment, and the rejections. A different approach is proposed in [8], where a service-level prediction provides a robust RL-based mechanism that can adjust to varying network conditions and heterogeneous hardware for autonomous VNF placement. With respect to RL-enabled self-healing, the study in [9] utilises a deep learning scheme to solve an NP-hard problem that is complemented with deep neural network and K-means clustering algorithms and ultimately alleviate the problem of cell outages, in a 5G context. A very similar deep learning solution is proposed in [10] but for IoT environments.

What remains underexplored in the literature, however, is the challenge of satisfying the diverse requirements coming from the various involved actors in an Edge/Cloud interplay, in a decentralized and autonomous way. Motivated to tackle this, in this work we propose a mechanism where the SFC owners independently formulate their healing strategy, while the infrastructure providers focus at maximizing their revenue through optimizing the SFC resource allocation during the healing. Our contributions can be summarized as follows:

- An RL mechanism based on Stochastic Learning Automata (SLA) is introduced to reactively respond to a site outage in an Edge-Cloud infrastructure (Sections II-III).
- We do not only emphasize on addressing the requirements of the SFCs but also on responding to the best interests of the infrastructure provider. Accordingly, we formulate a Knapsack-inspired allocation mechanism to solve this problem locally at each site (Section III).
- We propose a pipeline that successfully links the SLA and Knapsack-based mechanisms, where the healing solution is iteratively optimized and determined in a quasi real-time manner. Extensive simulation results corroborate the efficiency of our framework (Section IV).



(a) Outage (b) Result of the Self-Healing

Fig. 1: An example of the Self-Healing Process.

II. System Model

We model our Edge-Cloud infrastructure, as a graph $G = (V, E)$, where V represents the set of datacenter sites/nodes (Edge and Cloud) and E the set of links that interconnect them. Each node $v \in V$ is attributed with a vector of available resources $\boldsymbol{R_v} = [r_v^1, ..., r_v^N] \in \mathbb{R}^{1\times N}$. Regarding the SFCs, we assume that a set of S SFCs is already deployed in the infrastructure. Each SFC $s \in S$ may belong to a different vendor and consists of a number of interconnected VNFs $s = (s^1, s^2, ..., s^j)$, where s^j represents the j^{th} VNF of SFC s. Each VNF s_j is deployed as a VM which is characterized by a vector of resource demands $\boldsymbol{D_{s_j}} = [d_{s_j}^1, ..., d_{s_j}^N] \in \mathbb{R}^{1\times N}$, $s \in S$. We specifically consider three types of resources, i.e., $N = 3$, to account for the available CPU, Memory, and bandwidth capacities of the node and the VNF resource demands.

The seamless performance of an SFC relies on the proper functioning of all the consisting VNFs. For instance, even when a single VNF of an SFC is being hosted in a site that experiences an outage, the whole SFC is affected and the traffic cannot be processed in an end-to-end fashion. Accordingly, in this work we study the case of a single site outage, which however may negatively affect M SFCs, with $M \subset S$. When this happens, all the affected VNFs that belong to the M SFCs need to be relocated other functional sites. It should be noted, that when more than one VNFs of an SFC are hosted on the same faulty site, they are considered as merged and collocated processes in the same VM [11]. In this case a single VNF relocation can be considered. Thus, for the sake of simplicity, the terms VNF (i.e., s_j) and SFC (i.e., s) healing will be used interchangeably in the rest of the paper.

The healing solution can be described by a binary variable $y_{s,v}$, which takes the value 1 if the affected VNF of an SFC s is relocated to node v and 0 otherwise. For the healing to be considered successful, the new node v has to guarantee the resource requirements of the affected VNF (i.e., $\boldsymbol{R_v} \geq \boldsymbol{D_s}$). Furthermore, a VNF relocation of an SFC s should not lead to violations of the QoS relocation requirements q_s. Hence, we introduce $e_{s,v}$ to denote the additional delay that occurs by relocating VNF s to node v and $e_{s,v} < q_s$ should hold. This additional delay can be expressed by the added propagation delay or by the extra number of hops to reach the new node. As a final remark, in this work, only the outage compensation part the self-healing process is considered. Outage detection and diagnosis are part of our future work.

III. Algorithm Design

An iterative RL-based algorithm is designed, based on Stochastic Learning Automata (SLA) [12]. At each iteration, as a first step, the SFC owners decide where to relocate the affected VNFs. However, since the decision of each SFC is taken independently, the produced relocations in an iteration may not feasible. For instance, assume that in Fig. 1, at a specific iteration, all three VNFs that were located in the defective node 6 select node 5 as their healing solution, which does not have the necessary resources to accommodate them at the same time. In this case, a second stage decision should be made in the same iteration, regarding which of these VNFs will be allocated there. This local resource allocation optimization is revenue-driven and its results are fed back to the first step, where the SFC owners get to reconsider their decisions. This two-step procedure is iterated until convergence.

A. SLA-based Node Selection

We assume that the SFC providers serve as SLA agents and decide autonomously on the VNF healing, with the goal to optimize their long-term benefit [13]. This decision is affected by i) the satisfaction of the VNF resource demands in the new selected node and ii) the minimization of the additional delay occurred by the relocation. To this end, the reward gained by SFC s when healing to node v in iteration i is given by:

$$\mathscr{R}_{s,v}^{(i)} = A \cdot (1 - \frac{1}{1 + e^{-C(e_{s,v} - q_s)}}) + B \cdot y_{s,v}, \in [0, 1], \quad (1)$$

where $A, B \in [0, 1]$ are empirically selected coefficients with $0 \leq A + B \leq 1$, the fine-tuning of which guides the solution towards a desirable state. The gain coefficient $C \in \mathbb{R}^+$ assists in bringing the first term of the reward function close to 0 when the additional delay exceeds the maximum QoS relocation value (i.e., $e_{s,v} > q_s$) and close to 1 otherwise. According to [12], the probability for an SFC to select the same (and a different) node in the next iteration is updated as follows:

$$P_{s,v}^{(i+1)} = P_{s,v}^{(i)} + b \cdot \mathscr{R}_{s,v}^{(i)} \cdot (1 - P_{s,v}^{(i)}), v^{(i+1)} = v^{(i)} \quad (2)$$

$$P_{s,v}^{(i+1)} = P_{s,v}^{(i)} - b \cdot \mathscr{R}_{s,v}^{(i)} \cdot P_{s,v}^{(i)}, v^{(i+1)} \neq v^{(i)}, \quad (3)$$

where $b \in (0, 1)$ denotes the learning rate which adjusts the exploration/exploitation bias; higher b values accelerate convergence to a healing solution, which are, however, probably far from the optimum. Lower values of b yield solutions closer to optimal, at the expense of longer convergence. Convergence is achieved when a probability for each SFC s is close to 1.

B. Revenue-driven Local Resource Allocation Optimization

At the end of each iteration i, each SFC will stochastically select a node to heal to. Thus, a subset of SFCs $F_v \subseteq M \subseteq S$ is assigned to each node $v \in V$. For instance in Fig. 1b, the SFCs 2 and 3 select node 2 to be healed to, thus $F_2 = \{2, 3\}$. Following, each node decides which of these SFCs can be actually hosted and which need to find an alternative node due to resource constraints. To make this decision, each SFC $s \in F_v$, is associated with a revenue σ_s that the node will gain by admitting the particular VNF. Hence, the local resource

Algorithm 1 MKP to SKP Reduction

Input: $\boldsymbol{D_s}, \boldsymbol{R_v}, \boldsymbol{O_h}, \forall s \in F_v, \forall h \in H$
Output: $d'_s, r'_v, \forall s \in F_v$
1: $r'_v \leftarrow \boldsymbol{R_v} \oslash \boldsymbol{O_1}$ ▷ ($\boldsymbol{O_1} \rightarrow$ "smallest" VM flavor)
2: **for** $h \in H$ **do**
3: **if** ($\boldsymbol{R_v} \oslash \boldsymbol{O_h}$ is 0) **then** $w_h \gg r'_v$ **else** $w_h \leftarrow \lceil r'_v / (\boldsymbol{R_v} \oslash \boldsymbol{O_h}) \rceil$
4: **for** $s \in F_v$ **do**
5: **if** ((d'_s is $unassigned$) and ($\boldsymbol{D_s} \leq \boldsymbol{O_h}$)) **then**
6: $d'_s \leftarrow w_h$
7: **end for**
8: **if** (d'_s is $unassigned$) **then** $d'_s \gg r'_v$
9: **end for**

allocation optimization problem can be formulated as a Multi-dimensional 0-1 Knapsack problem (MKP) [14]:

$$\max \quad \sum\nolimits_{s \in F_v} \sigma_s y_{s,v} \tag{4a}$$

$$\text{s.t.} \quad \sum\nolimits_{s \in F_v} d_s^n y_{s,v} \leq r_v^n, \quad \forall n = 1, ..., N, \tag{4b}$$

$$y_{s,v} \in \{0, 1\}, \ \forall s \in F_v, \tag{4c}$$

which maximizes the reward of node v by admitting as many VNFs as possible, with the highest revenue σ_s, provided that the node's resources are not overprovisioned (Eq. 4b). It should be noted that an MKP, as a combinatorial problem and for its brute force search, yields a complexity of $O(|F_v| \cdot 2^{|F_v|})$, which grows exponentially and cannot be used when the number of competing SFCs, F_v, is large enough. To remedy this problem and to make our approach more real-time oriented, we reduce the dimensionality of the formulation as described below.

C. Reduction of the Optimization Problem

We transform problem 4 to a Single-dimension 0-1 Knapsack problem (SKP), by matching the resource requirements of the competing VNFs to H pre-specified VM "flavors" available in each node. Each flavor $h \in H$ is characterized by the offered resources, $\boldsymbol{O_h} \in \mathbb{R}^{1 \times N}$; in our case $N = 3$ (CPU, memory and bandwidth). Each flavor, depending on its size, is associated with a single-dimension weight w_h, that reflects how many times it can fit in a node. Hence, the available capacity $\boldsymbol{R_v}$ of a node v can be expressed as a scalar r'_v. Similarly, for an SFC, the requested set of resources $\boldsymbol{D_{s_j}}$ can also be represented as a scalar d'_s. Algorithm 1 describes the steps of the dimension reduction. We note that $\boldsymbol{a} \oslash \boldsymbol{b}$ calculates the smallest element of the Hadamard (or element-wise) integer division between the two vectors. Furthermore, the inequality $\boldsymbol{a} \leq \boldsymbol{b}$ denotes a logical operator which takes the value of 1 when all the elements of a matrix $\boldsymbol{a}$ are less than or equal to the corresponding elements of a matrix $\boldsymbol{b}$. Lastly, the operation $\lceil a \rceil$ is the ceiling operator for a scalar. We also assume that the advertised flavors $\boldsymbol{O_h}$ are sorted from the smallest to the largest and that the reduced capacity of the node becomes equal to the amount of the smallest flavor instances that can fit the node at the same time (line 1). Additionally, that each flavor's reduced weight is a number inversely proportional to the number of the VM instances of this flavor that can fit the node at the same time (lines 3-4). In lines 5-9, the smallest VM flavor $\boldsymbol{O_h}$ that can accommodate the requirements $\boldsymbol{D_s}$ of SFC s is selected and its weight w_h becomes the SFC's reduced weight d'_s. According to this transformation, the reduced SKP problem becomes:

$$\max \quad \sum\nolimits_{s \in F_v} \sigma_s y_{s,v} \tag{5a}$$

$$\text{s.t.} \quad \sum\nolimits_{s \in F_v} d'_s y_{s,v} \leq r'_v, \tag{5b}$$

$$y_{s,v} \in \{0, 1\}, \forall s \in F_v. \tag{5c}$$

The emerging SKP problem can be solved in pseudo-polynomial time with a $O(|F_v| \cdot r'_v)$ complexity when using the Bellman recursion [15]. Furthermore, the complexity of Algorithm 1 is equal to $O(|F_v| \cdot H)$. Hence, the overall complexity of the resource allocation problem is $O(|F_v| \cdot (H + r'_v))$.

IV. Numerical Results

We consider a hybrid infrastructure that consists of V Edge and Cloud sites, with the Cloud sites being more powerful, but found in much longer distance than the Edge ones. For the resources we use a three-dimensional vector of CPU, memory and bandwidth capacity, $[cores, GB, Gbps]$. We model the Edge capacity vector to range between $[2, 4, 1]$ and $[24, 48, 10]$, and the Cloud one from $[24, 48, 10]$ to $[96, 192, 40]$. The resources of the 3 available VM flavors are set as $[2, 4, 0.5]$, $[4, 8, 1]$ and $[8, 16, 2]$ respectively. We assume that the SFCs have various lengths, resource and QoS demands and they are already provisioned in the infrastructure prior to the outage. The resource requirements of each VNF lie in the range of $[1, 2, 0.5]$ and $[8, 16, 2]$. The number of SFCs and infrastructure nodes remain static throughout the simulation, while a single random node is considered to fail at each time. Our approach, however, could be easily modified to account for more outages in parallel. Finally, the learning rate is set as $b = 0.6$.

As a first experiment, the performance of the SKP reduction is evaluated. Fig. 2a illustrates the average revenue gained by a single site, when an incremental number of SFCs compete for its resources. Revenue σ_s acquired for each healed SFC s ranges between 1-10 and is proportional to the SFC resource demands. This value can be set according to the provider's revenue policy. SKP is compared with a Brute Force Optimization ("BFO") implementation, where the optimal solution is selected after evaluating all the possible healing placements. This results in the optimal revenue, though it is unrealistic, as its execution time scales exponentially with $|F_v|$ (e.g., $> 10s$ when $|F_v| = 20$). We also implement a greedy solution which prioritizes SFCs with low resource requirements, in an effort to maximize the number of allocations. For this comparison, 100 experiment repetitions were executed with fluctuating $|F_v|$ and SFC requirements and the results were averaged. Our algorithm is shown to outperform the greedy solution when the problem is no longer trivial, i.e., when not all the competing SFCs can fit the node at the same time. The difference between the collected and the optimal revenue is never greater than 10%, while the execution time is always close to that of the greedy algorithm ($< 1ms$), which acts as the baseline.

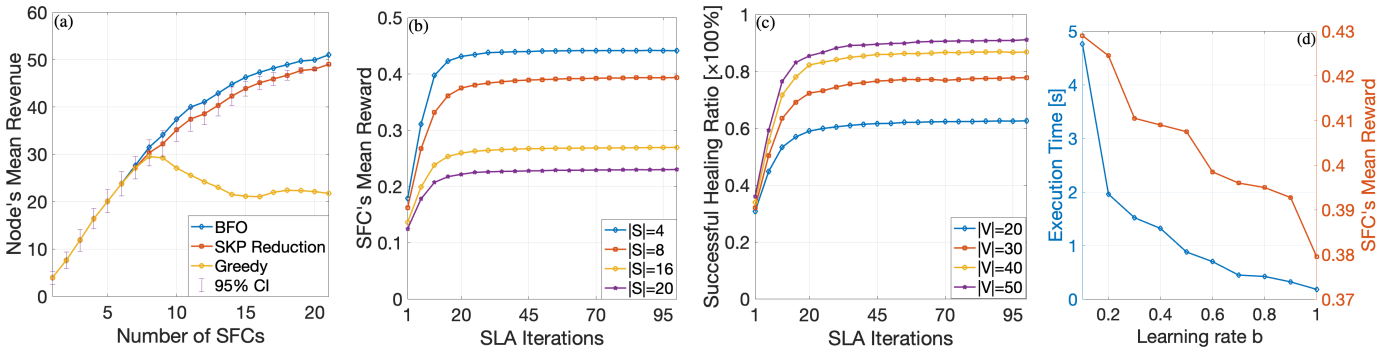


Fig. 2: Performance evaluation: (a) mean node revenue, (b) mean SFC reward and (c) healing ratio and (d) learning rate impact.

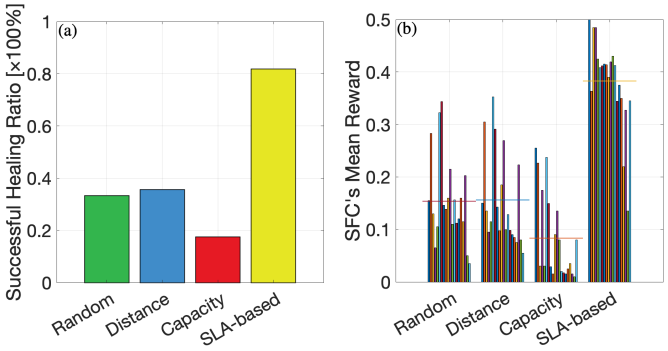


Fig. 3: Benchmarking of SFC (a) healing ratio and (b) reward.

Then, the efficiency of the proposed self-healing iterative optimisation is presented. A Monte Carlo simulation is conducted for different infrastructure and SFC settings. The results are averaged over 100 executions and presented in Fig. 2b, which illustrates the convergence performance of the SLA algorithm, for a 15-node infrastructure. For the first iteration, the healing node selection is random, as the probabilities $P^1_{s,v}$ are equal, $\forall s \in S, \forall v \in V$. As seen, our algorithm rapidly learns which new placements optimize the SFC reward, in less than 45 iterations. Nonetheless, the higher the number of the SFCs that try to heal, the lower the mean reward. This can be reasoned by the fact that it becomes increasingly difficult to find a healing solution that both minimizes the additional delay and satisfies the resource requirements for all the SFCs.

Fig. 2c presents the successful healing ratio results, which are acquired for 20 SFCs and an infrastructure size that increases from 20 to 50 sites. Once more, our algorithm learns the most suitable solution for each SFC rather quickly, while achieving a high successful healing ratio, especially for larger sizes of the infrastructure. For all the different infrastructure configurations, the healing ratio initiates from a percentage of around 30-35% and peaks close to 90%, when enough possible solutions become available (i.e., $|V| \geq 40$). For smaller sizes, it is normal to see lower healing ratios since there are not enough resources to accommodate the affected SFCs.

Additionally, the SLA efficiency is evaluated in terms of execution time, as demonstrated in Fig. 2d. For these experiments, we have set the number of SFCs to 20 and the size of the infrastructure to 35, while we increased the learning rate from 0.1 to 1. As we can see, as the learning rate increases, the exploration of the solution space becomes less exhaustive and the algorithm converges to a solution much faster. This has a negative effect on the mean reward achieved, which depicts the efficiency of the successful healing strategy. Nonetheless, for a learning rate of around 0.6 a very good trade-off is found, with an execution time of less than $1s$, which enables the SLA algorithm to find a very efficient healing solution in real-time.

Lastly, we provide a benchmark comparison against three baseline approaches, namely "Random", "Distance", and "Capacity"; for the first one, a random node is selected as the healing solution; in the second, the SFCs try to heal to the closest neighboring nodes from the defective one, while in the third, the healing solution is guided towards the nodes with the most available resources. The results presented in Fig. 3a are averaged over 100 executions for a configuration with 20 SFCs and 35 nodes. The proposed algorithm increases by more than double the successful healing ratio with respect to the baseline solutions. An interesting observation is that "Capacity" does not provide acceptable relocations, since it can only satisfy the resource but not necessarily the QoS requirements. This corroborates the fact that the SLA algorithm can find a very good trade-off between satisfying the SFCs' resource and QoS demands. This behavior is also validated in Fig. 3b, where the individual (vertical) and average (horizontal) rewards are given for every SFC and algorithm.

V. Conclusion

In this paper we studied the problem of SFC self-healing, when a datacenter site hosting multiple VNFs of various SFC fails, in an Edge-Cloud environment. A distributed RL algorithm based on Stochastic Learning Automata was introduced, which allows the SFC providers to autonomously select a possible healing solution based on a stochastic process that iteratively converges to a final solution. This algorithm was enhanced with a resource allocation optimization formulation that considers the available resources of the datacenters and maximizes their revenue from the healing process. The experimentation showed that the proposed algorithm maximizes the attained revenue for the infrastructure provider and the successful healing ratio for the SFC providers, in real time. Our future work will lie on failure diagnosis and detection, in order to introduce a complete, robust self-healing mechanism.

REFERENCES

[1] A. Leivadeas, M. Falkner, I. Lambadaris, and G. Kesidis, "Optimal virtualized network function allocation for an SDN enabled cloud," *Computer Standards & Interfaces*, vol. 54, pp. 266–278, 2017, sI: Standardization SDN&NFV.

[2] "Network Functions Virtualisation (NFV); Management and Orchestration ," ETSI, European Telecommunications Standards Institute, GS NFV-MAN 001, 2014. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/nfv-man/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf

[3] A. Asghar, H. Farooq, and A. Imran, "Self-healing in emerging cellular networks: Review, challenges, and research directions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 1682–1709, 2018.

[4] J. P. Santos, R. Alheiro, L. Andrade, A. L. Valdivieso Caraguay, L. I. Barona Lopez, M. A. Sotelo Monge, L. J. Garcia Villalba, W. Jiang, H. Schotten, J. M. Alcaraz-Calero *et al.*, "SELFNET Framework self-healing capabilities for 5G mobile networks," *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 9, pp. 1225–1232, 2016.

[5] A. M. Medhat, G. A. Carella, M. Pauls, M. Monachesi, M. Corici, and T. Magedanz, "Resilient orchestration of service functions chains in a NFV environment," in *2016 IEEE conference on network function virtualization and software defined networks (NFV-SDN)*. IEEE, 2016, pp. 7–12.

[6] S.-Y. Song and F. J. Lin, "Dynamic Fault Management in Service Function Chaining," in *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2020, pp. 1477–1482.

[7] J. Pei, P. Hong, M. Pan, J. Liu, and J. Zhou, "Optimal VNF placement via deep reinforcement learning in SDN/NFV-enabled networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 263–278, 2019.

[8] M. Bunyakitanon, X. Vasilakos, R. Nejabati, and D. Simeonidou, "End-to-end performance-based autonomous VNF placement with adopted reinforcement learning," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 2, pp. 534–547, 2020.

[9] X. Yang, P. Yu, L. Feng, F. Zhou, W. Li, and X. Qiu, "A deep reinforcement learning based mechanism for cell outage compensation in 5G UDN," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2019, pp. 476–481.

[10] J. Guo, Z. Wang, X. Shi, X. Yang, P. Yu, L. Feng, and W. Li, "A deep reinforcement learning based mechanism for cell outage compensation in massive IoT environments," in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE, 2019, pp. 284–289.

[11] A. Leivadeas, G. Kesidis, M. Falkner, and I. Lambadaris, "A Graph Partitioning Game Theoretical Approach for the VNF Service Chaining Problem," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 890–903, 2017.

[12] C. Unsal, "Intelligent navigation of autonomous vehicles in an automated highway system: Learning methods and interacting vehicles approach," Ph.D. dissertation, Virginia Polytechnic Institute and State University, 1997.

[13] M. Diamanti, E. E. Tsiropoulou, and S. Papavassiliou, "Resource Orchestration in UAV-assisted NOMA Wireless Networks: A Labor Economics Perspective," in *ICC 2021-IEEE International Conference on Communications*. IEEE, 2021, pp. 1–6.

[14] A. Fréville, "The multidimensional 0–1 knapsack problem: An overview," *European Journal of Operational Research*, vol. 155, no. 1, pp. 1–21, 2004.

[15] D. Pisinger, "Where are the hard knapsack problems?" *Computers & Operations Research*, vol. 32, no. 9, pp. 2271–2284, 2005.