Scalable and Cost Efficient Resource Allocation Algorithms Using Deep Reinforcement Learning

Mohammed Laroui *†‡, Moussa Ali Cherif *, Hatem Ibn Khedher †‡, Hassine Moungla †‡, and Hossam Afifi‡
*EEDIS Laboratory, Computer Science Departement, Djillali Liabes University, Sidi Bel Abbes, Algeria
†Université de Paris, LIPADE, F-75006 Paris, France
‡UMR 5157, CNRS, Institut Polytechnique de Paris, Telecom SudParis Saclay, France
Emails: {mohammed.laroui, moussa.alicherif}@univ-sba.dz, {hassine.moungla, hossam.afifi}@telecom-sudparis.eu
{mohammed.laroui, hassine.moungla, hatem.ibn-khedher}@parisdescartes.fr

***Abstract*—The emergence of a new generation of applications led to the appearance of new challenges that represent improvements in current communication technologies. For this, a new network paradigm's including edge computing that allows the process of data at the edge of the network. And the 5G network slicing that represents a new generation of communication increases the capacity of mobile networks by supporting the slicing technology that allows virtual "cutting" of a telecommunications network in several slices that provide high performance in terms of bandwidth and latency. Slice allocation and placement is an important networking optimization task that still painstakingly tune heuristics to get a sufficient solution. These algorithms use data as input and outputs near-optimal solutions. Thus, we are motivated by replacing this tedious process with the recent deep reinforcement learning algorithms. In this paper, we propose three approaches for Virtual Network Functions (VNFs) slices placement in edge computing (Integer linear programming (ILP), reinforcement learning (RL), and deep reinforcement learning (DRL)). Then they are implemented and evaluated. Several scenarios are considered to study the behavior of the algorithms and to quantify the impact of network size. The results show the feasibility and efficiency of the proposed techniques in terms of server utilization, placement time, and energy consumption.**

***Index Terms*—VNF, Edge Computing, Network Slicing, Optimization, Deep Learning, Deep Reinforcement Learning.**

I. Introduction

Edge Computing [1] is a new network paradigm, allows the process of data in the edge of the network by decentralizing a part of data processing from the cloud to the edge of the network. Its offers a lot of benefits including reduce latency, provide services in short time which is required in new generations of future applications.

The network slicing [2] has been considered as the main key to the rapidly progressing of 5G technology [3], it enables the slicing of the physical network into independent virtualized logical networks. Each network slice is a separated network designed to perform different requirements requested by an application. For this reason, the network slicing technology plays a central role in mobile networks [4] that allow embracing a lot of services with diverse requirements.

Network resources slicing and management problems are ubiquitous in computer systems and networks [5]–[7]. Examples include cluster scheduling, congestion control, virtual machine placement and migration, multi-commodity flow routing, combinatorial optimization and so on [8]. In this context, ETSI standardises VNF placement and slicing frameworks in edge computing. Despite the huge effort in designing the overall VNF reference architectures (i.e., in management and orchestration), ETSI VNF manager modules still suffer from the legacy heuristics used to design scalable and adaptive resource slicing and management approaches. In this paper, three approaches for finding optimal and heuristic VNF slices placements are proposed (ILP, Q-learning, and DRL) in fog/edge computing taking into account network slicing facility. In these methods, a single decision-maker optimizer receives heterogeneous parameters related to the system (CPU, GPU, Storage), the network (BW), and the cost/QoE. Then, it tries to place VNF slices on the embedded infrastructure while maximizing the overall system and quality performances and minimizing the total placement cost. It represents the cost of offloading VNF slices on optimal edge servers. Moreover, different reservoir computing capacities are considered to quantify the behavior of the proposed algorithms and to compare their efficiency in terms of placement cost, execution time, and energy consumption. Then, to deal with dense and complex networks the proposed algorithms are implemented and evaluated under different network scales (small and large scale).

The rest of the paper is organized as follows: In section II, we study the relevant work in the field of network optimization and service placement. Section III reviews the reinforcement learning and deep reinforcement learning. Section IV reviews the context formulation and the proposed architecture. In section V we formulates the exact, RL, and DRL models for VNF slices placement. The performance evaluation presented in section VI. And finally, we conclude the paper in section VII.

II. Related Work

This section highlights the relevant VNF placement optimization algorithms in 5G based virtualized edge computing context.

Agarwal *et al.* [9] presented a model for SDN/NFV over 5G that can capture arbitrary VNF graphs, and allow VNF to have multiple instances. The used methodology is based on

Mr. Laroui is corresponding author.

978-1-7281-3129-0/20/$31.00 ©2020 IEEE

the MaxZ placement heuristic for CPU assignment and VNF placement, then the MaxZ is combined with a methodology for optimal CPU allocation. The numerical results show that the proposed model closely matches the optimum.

Cziva *et al.* [10] formulated a VNF placement in the Edge of the network based on optimal stopping theory to dynamically place the VNF. The dynamic scheduler is evaluated in a simulation environment using real ISP latency. The proposed solution minimizes the VNF migrations and latency. However, the constraints of chaining are not modeled.

Zhou *et al.* [11] proposed a VNF placement model, then two heuristic algorithms called VNFP and VNFP-adv are proposed to solve the problem. The first algorithm responsible for maximizing the scalability, where the second algorithm considers server utilization and path length concurrently. The performance evaluation show that the proposed algorithms reduce the packet loss ratio with higher server utilization and short path length.

Similarly, Zhang *et al.* [12] proposed a VNF placement heuristic approach over 5G network slicing framework to automatically place the VNFs in 5G network slices. They used two 5G scenarios for the simulation that prove over the founded results that the proposed solution is efficient in terms of total throughput. However, the proposed model require high computation resources in large scale networks.

Nguyen *et al.* [13] proposed a VNF placement model based on Mixed-Integer Linear Programming (MILP). The proposed model supports two-way traffic in different network services. The performance evaluation shows that the proposed MILP model reduces the VNF instance number required for deployment. As a result, the algorithm saves the total resources and reduces the end to end delay. However, the authors do not propose solutions for large scale which is important in 5G networks.

More generally, recent work in the state of the art are based on finding a way to solve the VNF placement problem using exact/heuristic optimization techniques. However, a few work has been studying the combinatorial optimization learning approach that may improve the overall network performance. Looking ahead, deploying an RL-based VNF placement in edge computing over 5G network slicing may replace exact and heuristic approaches to get sufficient solution.

III. Background

In this section, we briefly review the reinforcement learning techniques that we leverage on this paper. We refer readers to [14] for relevant surveys and detailed formulations.

A. Reinforcement Learning (RL)

We consider the general networking setting shown in Figure 4 where an agent (in the left) interacts with an environment. At each time step t, the agent observes some state s_t, and is asked to choose an action a_t. Following the action, the state transitions s_{t+1} and the agent receives reward r_t. The rewards state transitions are stochastic and are assumed to have the Markov property; i.e. the rewards and state transition probabilities depend only on the environment state s_t and the action taken by the agent a_t.

The agent can control only its actions. It has no a prior knowledge of the next state (i.e., the state, the environment would transition to) or the future reward. During the learning process, and by interacting with the environment, the agent can observe these parameters.

Reinforcement learning is an optimization process while its goal is to maximize the expected cumulative discounted reward. It can be formulated as follows:

$$OBJ = \sum_{t=0}^{\infty} \gamma^t \times r_t \tag{1}$$

The RL agent chooses actions based on a policy π, defined as a probability distribution over actions. $\pi(s, a)$ is the probability that the agent picks action a in state a.

The agent can learn a Q function that quantifies a (state,action) pair. $Q(s, a)$ represents the quality of the state s when action a is taken. It represent the accumulated reward. The Q learning can be simply described as follows:

1) Initialize the Q table with small random weights representing the benefits of performing an action a in a state s. Initially these weights are random and are totally wrong.
2) Perform an action a (let be a random action at the exploration phase).
3) Receive a reward/punishment of the current state r and the ID of the next state s'.
4) Using the Q state-action values (the table entries), select the action a' that has the maximum Q-value (i.e., $\max_{a'} Q(s', a')$).
5) Update the Q table according to the Bellman equation:

$$Q(s_{t+1}, a_{t+1}) = Q(s_t, a_t) + \alpha(r_t + \gamma \times \max_{a'_t} Q(s'_t, a'_t) - Q(s_t, a_t)). \tag{2}$$

6) Perform the selected action (i.e., according to greedy selection for example).

The agent continues to update the table until reaching the optimal values stored at another $Q(s, a)^*$ table. The converging state can be easily detected when the Q-values remains unchanged after the update process.

The $\max_{a'} Q(s', a')$ that we use to update Q(s,a) is only an approximation (completely wrong in early stages). Nevertheless, the approximation get more accurate in each iteration. Then, Q-function will converge and represent the optimal/true Q-value.

Figure 1 outlines the general Q-Learning algorithm.

B. Deep Reinforcement Learning (DRL)

Deep learning is a function approximation. Therefore, it can be used to represent the described Q-function. Figure 2 outlines the deep learning utilization in the context of Q-learning. The merge between deep learning and reinforcement learning (Q-learning) called Deep Reinforcement Learning (DRL). Among the main instances of DRL we cite Deep Q-learning (DQN)

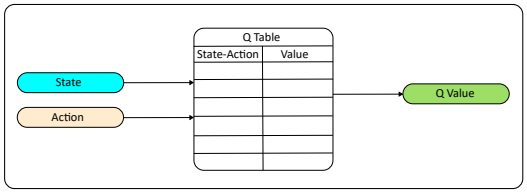


Fig. 1: Q-Learning.

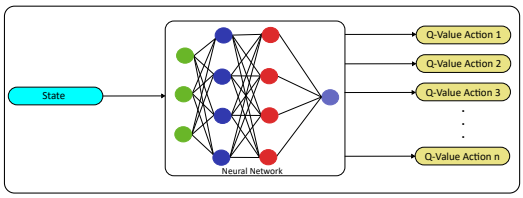


Fig. 2: Deep Q-Learning.

where the Q algorithm is used. The input of the deep neural network represent the state values and the output represent the action values. Still, the objective of the DQN agent is to find the optimal weights that approximate the Q-value at each state.

Given a transition $\{s, a, r, s'\}$, the Q-table update rule in the previous algorithm (without neural network) must be replaced with the following:

1) Performs a feed-forward pass for the current states to get predicted Q-values (states, actions) for all actions.
2) Perform a feed-forward pass for the next state s' and calculate maximum overall network outputs $\max_{a'} Q(s', a')$.
3) Set Q-value target for action to $r + \max_{a'} Q(s', a')$ (use the max value calculated in step 2).
4) For all other actions, set the Q-value target to the same as initially returned from step 1, making the error 0 for those outputs.
5) Update the weights using the back-propagation technique that minimizes the loss described as follows

$$\min_{\theta} || \, r + \max_{a'} Q(s', a') - Q(s, a; \theta) \, ||_2 \qquad (3)$$

IV. Context Formulation and the Proposed Architecture

A. Context Formulation

ETSI has recently expected the profits of the mobile edge computing (MEC) architecture to run all its entities and applications as VNFs in NFV environment [15].

The VNF placement problem in the cloud has achieved more attentiveness. It is similar to the placement of Virtual Machines (VMs), where the VNFs are composed of containers or VMs that can execute network functions. Indeed, the placement of VNFs at the network edge provides a lot of benefits including proximity to end-users which by consequence decreases the latency and enhances the quality of services.

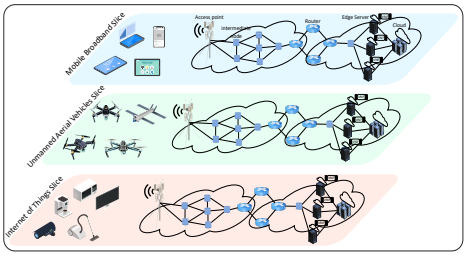


Fig. 3: VNF Slices Placement in Edge Computing Over 5G Network.

TABLE I: Summary of the most used notation in this paper.

Notations	**Definition**
$\mathcal{N}$	The number of server in the edge.
$\mathcal{M}$	The number of vnf for the placement.
w_j	The number of slices in the vnf $j \in \mathcal{M}$.
c_i	The number of cpu slots in the server $i \in \mathcal{N}$.
Decision variables	**Definition**
y_i	A binary variable indicate if the server $i \in \mathcal{N}$ is used.
x_{ij}	A binary variable that assigns the vnf $j \in \mathcal{M}$ to the server $i \in \mathcal{N}$.

5G network slicing enables the multiplexing of independent and virtualized networks, where each slice corresponds to an isolated network to perform application requirements.

B. The Proposed Architecture

The proposed architecture allows the placement of VNF slices at the network edge in the context of 5G network where the physical network is sliced to an isolated logical networks, each slice corresponds to an application type including IoT, Unmanned aerial vehicles (UAV), augmented reality (AR) and so on.

Figure 3 depicts our reference architecture of VNF slices placement in edge computing in the context of 5G networks.

V. VNF Slices Placement Model Over 5G

A. The exact ILP algorithm

ETSI standard does not specify how VNF slices can be deployed in a large scale network. In this subsection, we give the general formulation of the exact VNF placement algorithm in edge computing. Table I depicts the notations used in the ILP model. The single decision-making optimization is as follows:

$$\min \sum_{i=1}^{n} \alpha_i y_i \qquad (4)$$

Subject to

$$\sum_{j=1}^{m} w_j x_{ij} \leq c_i y_i, \quad i \in \mathcal{N} = \{0, ..., n\} \qquad (5)$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j \in \mathcal{M} = \{0, ..., m\} \qquad (6)$$

$$y_i \in \{0, 1\}, \quad i \in \mathcal{N} \qquad (7)$$

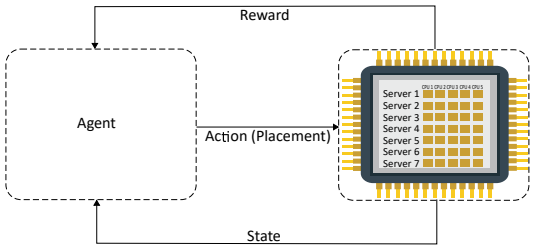


Fig. 4: RL application to VNF placement in Mobile Edge Computing.

$$x_{ij} \in \{0,1\}, \quad i \in \mathcal{N}, \quad j \in \mathcal{M} \tag{8}$$

We quote in the following the proposed decision variables used to solve the ILP formulation:

- The binary variable y_i indicates if the server i is used or not. It is defined as follow:

$$y_i = \begin{cases} 1 & \textit{if the server } i \textit{ is used;} \\ 0 & \textit{Otherwise.} \end{cases} \tag{9}$$

- The binary variable x_{ij} indicates the assignment of the vnf j to the optimal server i. It is defined as follow:

$$x_{ij} = \begin{cases} 1 & \textit{if vnf } j \textit{ is assigned to the server } i; \\ 0 & \textit{Otherwise.} \end{cases} \tag{10}$$

Equation 4 formulates the proposed objective function. It represents the total used servers. Note that each server is characterized by α_i that depends also on the server i and the operator policy. Equation 5 ensures that each server can not exceed its computing capacity represented as CPU slots. Note here that the incoming slices per VNF w_j may follow a predefined distribution in order to feed the optimization algorithm. Equation 6 ensures that each VNF can be associated with at most one server.

The above problem is NP-hard due to our combinatorial complex system and therefore the proposed exact algorithm is difficult to scale up to decide where to place VNF slices in a large scale scenario. As a consequence, efficient heuristic algorithms are proposed in the next subsection.

B. The RL algorithm

In this subsection, we present our scheme for online single resource VNF slices placement with deep reinforcement learning. We formulate the problem 4 and demonstrate how to represent it as a RL task. Then, we present our RL-based solution building on the machinery presented in the previous section.

Q-learning is the main algorithm used for model-free RL. The meaning of the Q function is to estimate the future placement reward. We describe hereafter the proposed model.

1) The proposed model: As shown in Fig. 4, we consider a computing cluster with a single resource; the CPU. VNF are jobs that arrives to the cluster with an online fashion in discrete time-steps. At each time steps, the cluster manager/scheduler chooses a VNF to place. We assume that the VNF demands (number of slices) is known upon arrival.

Algorithm 1 RL-driven VNF slices placement in Edge Computing Over 5G Network

1: **Input:** Servers, the number of slots per Server (CPUs or GPUs), the number of VNFs, and the number of slices per VNF
2: **Output:** $Q^*(s,a)$
3: Initialise Q(state,action) arbitrary.
4: Observe an initial state s (current allocation an first incoming VNF slice)
5: **repeat**
6: Select and place a VNF slice on the edge computing server
7: Observe the placement cost r and the new state s' (new allocation and another incoming VNF slice)
8:

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma \times \max_{a'} Q(s',a') - Q(s,a)) \tag{12}$$

9: s = s'
10: **until** no incoming VNF slices

2) The proposed formulation: We represent the state of the system as the current placement of VNF slices on server slots. In Fig. 4 (right side) we show the proposed state space which can be considered as an environment snapshot.

The action space is the placement function of VNF slice on a server slot. The placement takes into consideration the server capacity (in terms of slots). In fact, the agent will not place the VNF on a server slot if it is occupied by running another VNF. The action is mono-type where the agent process VNFs one by one until processed all the incoming user requests.

The proposed reward is the placement cost of such a VNF. It is measured as the number of server used after performing an action. It is formulated as follows

$$R_t = \begin{cases} 100 \times i & \textit{if } i \textit{ servers are opened} \\ 0 & \textit{Otherwise.} \end{cases} \tag{11}$$

As shown in the above equation 11 the main objective of the agent is to minimise the total opened servers. In other words, the agent objective will be minimizing the total discounted cumulative rewards.

In Algo. 1 we describe the pseudo-code of our proposed RL algorithm.

C. The DRL algorithm

In the DRL algorithm we try to use the deep neural network to approximate the above RL model. A succession of layers of neural networks are used to map the input (the state) to the output (the action). In Algo. 2 we describe the pseudo-code of our proposed DRL algorithm.

In the deep neural network approach, we have used the Stochastic Gradient Descent (SGD) algorithm to perform the training of the DQN. Moreover, hyper-parameter tuning are needed to decide about the optimal deep neural network configurations such as epoch number, optimizer algorithm, and action selection strategies. The DRL approach consists to reduce the complexity of the ILP and RL techniques by reducing the number of iteration to be considered in our optimization.

VI. VNF Slices Placement Performance Evaluation

The performance of the proposed models has been evaluated using three different environments (small, medium, large). Fur-

Algorithm 2 DRL-driven VNF slices placement in Edge Computing Over 5G Network

Input: Servers, the number of slots per Server (CPUs or GPUs), the number of VNFs, and the number of slices per VNF
Output: $Q^*(s, a)$
Initialize a reply memory D
Initialise action-value Q with random weights
Observe initial state s
repeat
    Select an edge computing server a .
    - with probability ϵ select a random edge computing server
    - Otherwise select the server that has the $\max_{a'} Q(s, a')$

    Place the VNF slice on the selected edge computing server a.
    Observe the placement cost r and the new state s' (new allocation and another incoming VNF slice)
    store the experience $\{s, a, r, s'\}$ in the replay memory.
    sample a random transition from the replay memory.
    Calculate the target for each mini-batch transition $(r + \gamma \times \max_a Q(s', a'))$
    Train the Q network using the following loss

$$Loss = \frac{1}{2} * (r + \gamma \times \max_{a'} Q(s', a') - Q(s, a))^2 \quad (13)$$

    s = s'
until no incoming VNF slices

TABLE II: Servers Configuration.

Environments	The number of servers	The number of cpu in servers (slots/servers)
Environment 1 (Small)	5	5/5
Environment 2 (Medium)	10	3/3, 4/3, 5/2, 6/2
Environment 3 (Large)	15	3/2, 4/3, 5/4, 6/3, 7/3

TABLE III: VNF Configuration.

VNF Configuration	The number of vnf	The number of slices in vnf (slices/vnf)
Configuration 1	5	2/3, 3/2
Configuration 2	10	1/3, 2/3, 3/4
Configuration 3	15	1/8, 2/5, 3/1, 4/1

ther we have used CPLEX optimization tool [1] to implement the ILP model given the linearity of our constraints. Furthermore, TensorFlow, an end-to-end open-source Artificial Intelligence (AI) platform [2] is used to configure and implement the deep learning models, where the EdgeCloudSim [16] is used to evaluate the efficiency of the three models in large scale networks.

A. Environment Scenarios

For each environment we define a set of parameters including *the number of servers, the number of CPUs for each server* as mentioned in table II. Then, we specify *the number of VNFs, the number of slices for each VNF* as depicted in table III.

B. Key Performance Indicators (KPI)

We propose three KPIs in order to assess the efficiency of the proposed algorithms: *placement time*, *server utilization*, and *energy consumption in the edge*. The former indicates the efficiency of the proposed models in terms of placement time, resources utilization. The latter measures the energy consumed during the VNF slices placement process.

C. Performance Evaluation

We have proposed three different environments as depicted in Tab. II. We mention here that we proposed three environment instances to define the model size and select the appropriate approach. Moreover, for the sake of better quantify each environment instance, we have added three VNF configuration as presented in Tab. III.

Fig. 5 shows the server utilization according to the proposed approaches (the exact ILP, the Q-Learning, and the Deep Q-Learning (DQN)).

Fig. 5a show the server utilization in small environment. We show that the Q-learning and DQN give the optimal solution (ILP) of vnf slices placement.

Figure 5b depict the server utilization in medium configuration, we show that the DQN give an efficient solution compared to the Q-learning for the configuration 1, where for the configurations 2 and 3, the Q-learning and the DQN give the same server utilization.

Figure 5c display the server utilization in large environment, we show that the DQN give an efficient solution compared to the Q-learning.

Figure 6 depict the performance evaluation of the vnf slices placement in the edge using ILP, Q-Learning, and DQN in large scale networks.

Figure 6a display the placement time for the three models, we show that when the number of vnf slices increase, the time increase exponentially for the ILP compared to Q-learning and DQN which prove the problem of the ILP model in large scale networks. In addition, the placement time of DQN is more efficient compared to Q-learning model.

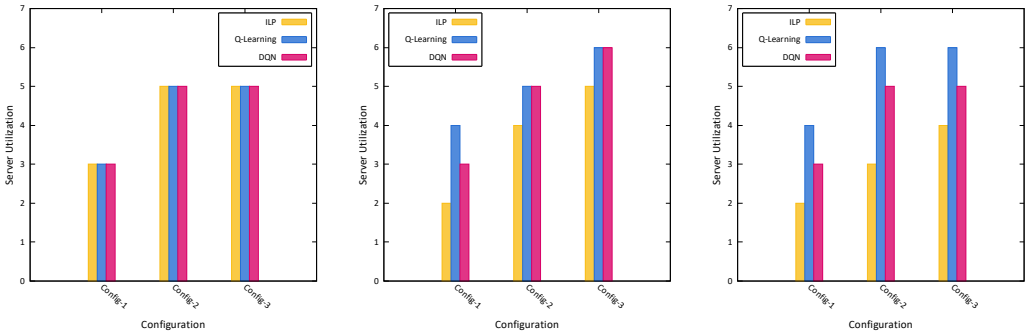
Figure 6b show the server utilization for Q-learning and DQN in large scale networks. We show that the DQN give efficient solution in terms of server utilization compared to the Q-learning model.

Figure 6c show the energy consumption in the edge for ILP, Q-learning and DQN in large scale networks. We show that when the number of vnf slices is less than 400, the ILP give an efficient energy consumption compared to Q-learning and DQN. Where when the number of vnf slices is greater than 400, the energy consumption of ILP is very high compared to the Q-learning and DQN because of the high computation resources used by the ILP. In addition, the DQN is the efficient solution in terms of energy consumption.

VII. Conclusion

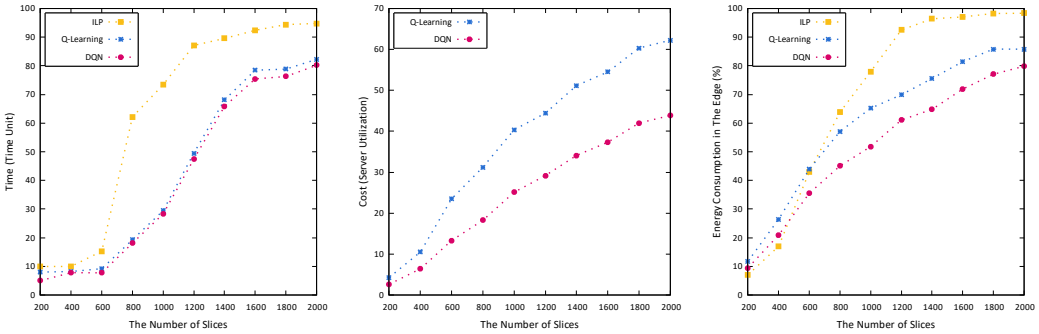
In this paper, we have proposed scalable and cost efficient VNF slices placement algorithms in edge computing using linear programming techniques in small scale and (deep) reinforcement learning algorithms in large scale. The proposed algorithms are modeled, implemented and evaluated under different network and computing scenarios. Results show the efficiency of the proposed approaches in both network scales in terms of server utilization and energy consumption metrics.

[1] https://pypi.org/project/cplex/
[2] https://www.tensorflow.org/



(a) Environment 1 (Small) (b) Environment 2 (Medium) (c) Environment 3 (Large)

Fig. 5: Performance Evaluation of VNF Slices Placement using ILP, Q-Learning, DQN in Three Different Environments.



(a) Placement Time (b) Server Utilization (c) Energy Consumption

Fig. 6: Performance Evaluation of VNF Slices Placement in the edge using ILP, Q-Learning, DQN in Large Scale Networks.

Future work can be conducted on including a real VNF slices pattern to feed the optimization algorithms and detect anomalies in the normal distribution. Moreover, using multi-model deep learning techniques such as generative adversarial networks over time series would be beneficial in terms of placement costs and energy consumption. Moreover, we will integrate the aforementioned placement algorithm using Information-Centric Networking in Extended Edge Computing.

REFERENCES

[1] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge computing: A survey," *Future Generation Computer Systems (FGCS)*, vol. 97, pp. 219–235, 2019.

[2] M. Afaq, J. Iqbal, T. Ahmed, I. U. Islam, M. Khan, and M. S. Khan, "Towards 5G network slicing for vehicular ad-hoc networks: An end-to-end approach," *Computer Communications*, vol. 149, pp. 252–258, 2020.

[3] V. P. Kafle, P. Martinez-Julia, and T. Miyazawa, "Automation of 5G Network Slice Control Functions with Machine Learning," *IEEE Communications Standards Magazine*, vol. 3, no. 3, pp. 54–62, 2019.

[4] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y.-J. A. Zhang, "The roadmap to 6G: AI empowered wireless networks," *IEEE Communications Magazine*, vol. 57, no. 8, pp. 84–90, 2019.

[5] H. Moungla, A. Jarray, A. Karmouch, and A. Mehaoua, "Cost-effective reliability-and energy-based intra-WBAN interference mitigation," in *IEEE Global Communications Conference*, 2014, pp. 2399–2404.

[6] M. Ali, H. Moungla, M. Younis, and A. Mehaoua, "Inter-WBANs interference mitigation using orthogonal walsh hadamard codes," in *IEEE 27th annual international symposium on personal, indoor, and mobile radio communications (PIMRC)*, 2016, pp. 1–7.

[7] H. Khelifi, S. Luo, B. Nour, H. Moungla, and S. H. Ahmed, "Reputation-based blockchain for secure NDN caching in vehicular networks," in *2018 IEEE Conference on Standards for Communications and Networking (CSCN)*, 2018, pp. 1–6.

[8] H. Khedher, E. Abd-Elrahman, H. Afifi, and M. Marot, "Optimal and Cost Efficient Algorithm for Virtual CDN Orchestration," in *IEEE 42nd Conference on Local Computer Networks (LCN)*, 2017, pp. 61–69.

[9] S. Agarwal, F. Malandrino, C.-F. Chiasserini, and S. De, "Joint VNF placement and CPU allocation in 5G," in *IEEE Conference on Computer Communications (INFOCOM)*, 2018, pp. 1943–1951.

[10] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal VNF placement at the network edge," in *IEEE Conference on Computer Communications (INFOCOM)*, 2018, pp. 693–701.

[11] W. Zhou, Y. Yang, M. Xu, and H. Chen, "Accommodating dynamic traffic immediately: A VNF placement approach," in *IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[12] Q. Zhang, F. Liu, and C. Zeng, "Adaptive interference-aware VNF placement for service-customized 5G network slices," in *IEEE Conference on Computer Communications (INFOCOM)*, 2019, pp. 2449–2457.

[13] T.-H. Nguyen, J. Lee, and M. Yoo, "A practical model for optimal placement of virtual network functions," in *IEEE International Conference on Information Networking (ICOIN)*, 2019, pp. 239–241.

[14] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[15] "Mobile Edge Computing (MEC), Deployment of Mobile Edge Computing in an NFV environment," *ETSI Group Report MEC 017*, 2018.

[16] C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, p. e3493, 2018.