Optimizing Edge Resource Allocation for Sustainable and Latency-aware Applications

Nour-El-Houda Yellas
SAMOVAR, Telecom SudParis, Institut Polytechnique de Paris
Palaiseau, France
Email: firstname.lastname@telecom-sudparis.eu

Yann Dujardin and Nancy Perrot
Orange Research
Chatillon, France
Email: firstname.lastname@orange.com

***Abstract*—The advent of Network Function Virtualization (NFV) and virtualized Content Delivery Network (vCDN) has revolutionized the deployment of resources at the edge of the network, offering a more efficient alternative to traditional CDN architectures. However, this approach introduces the challenge of resource limitations at the edge, making effective resource allocation a critical issue. This paper tackles the problem of placement of virtual network functions (VNF) by proposing a planning strategy to assign end-users access points to edge servers where vCDN functions are deployed, ensuring compliance with Service Level Agreement (SLA) while minimizing the energy consumption. We show that the problem is NP-hard and then propose a Mixed Integer Linear Program (MILP) to formulate our problem, making use of a non-linear energy model from the literature to estimate the energy footprint. We evaluate the proposal leveraging real traffic demand data from a nationwide mobile operator to model realistic network conditions. Additionally, we investigate the impact of varying the number of edge servers on the overall energy footprint. Our results demonstrate the effectiveness of the proposed optimization strategy in reducing energy consumption while maintaining the required quality of service compared to a baseline approach.**

***Index Terms*—vCDN, energy consumption, resource allocation, optimization.**

I. Introduction

In the era of increasing digital consumption, the demand for high-quality and low latency in content delivery has exploded, driving the need for deploying efficient and scalable Content Delivery Networks (CDN). As part of the network virtualization paradigm, virtual CDN (vCDN) represents a promising solution to cope with the massive amount of traffic demands, especially for content delivery and applications with strict performance, latency, and bandwidth constraints, through the dynamic deployment of caching nodes on edge servers [1]. This approach leverages the Network Function Virtualization (NFV) and Multi-access Edge Computing (MEC) paradigms while allowing Mobile Network Operators (MNOs) to scale their resources based on the demand fluctuation, thereby minimizing the energy consumption.

The deployment of vCDN services entails several challenges, such as the orchestration of the vCDN components (i.e., Virtualized Network Functions or VNFs) on the MEC servers where the traffic uncertainty and system constraints should be taken into consideration. Another challenge is the management of routing systems, where the use of an SDN controller is suggested to ensure efficient content delivery from the edge of the network. Finally, the optimization of content management strategies is an important task in order to maintain a balance between cache capacity and content availability.

In this paper, we focus on the challenge of energy-efficient resource allocation and placement of VNFs that ensure the content delivery services. In fact, the dynamic patterns of traffic demands introduce substantial uncertainty, making it difficult to predict future demands and optimally allocate resources in real-time or near real-time. To address these challenges, we propose a MILP formulation for orchestrating the decision of the vCDN function placement that effectively cope with traffic fluctuations and guarantees the best possible Quality of Service (QoS) while minimizing energy consumption.

The contributions of this paper are as follows.

- We propose a MILP formulation to efficiently plan the placement of the vCDN-related VNFs at the edge under CPU resource constraints while jointly considering the assignment of the MNO Base Stations (BSs) to the active VNFs. Additionally, our model considers QoS constraints, that is, end-to-end latency, and minimizing the overall energy consumption.
- We take into consideration the traffic demands dynamics, so the orchestration decision should adapt to the current traffic demands.
- We present preliminary results where we assess the effectiveness of our proposal using real-world traffic demands, and we compare it to a greedy nearest fit placement algorithm from the literature.

The remainder of this manuscript is organized as follows. In Section II, we give an overview of existing works. We present the problem statement, the complexity proof and the VNF placement model in Sections III IV and V. We present our results in Section VI. We draw conclusions in Section VII.

II. Related works

Numerous studies in the literature explore the use case of virtualized CDN, and can be classified based on: (i) the deployment environment, including datacenters and cloud infrastructures [2]–[4], and edge-fog environments [5], (ii) the virtualization of CDNs aligned with NFV architecture [6] and (iii) the optimization of content routing using SDN paradigm.

Regarding the virtualization of CDN functions, several works address key challenges such as VNF placement [6] [8], resource scheduling, provisioning, and scaling operations [9]. The objectives pursued in these studies vary significantly, with some aiming to minimize deployment costs, others focusing on maximizing robustness against traffic uncertainty and QoS, and a subset dedicated to enhancing energy efficiency.

For example, the authors in [6] present a novel approach to optimize content delivery in virtualized infrastructures. They first propose a vCDN architecture based on the ETSI-NFV framework where the CDN components are virtualized then they propose OPAC, a cache placement algorithm and cache migration for vCDN while taking into consideration system constraints such as user demands, system load and CPU and network conditions.

Moreover, energy efficiency in vCDN environments has been explored from multiple perspectives. Some studies, such as [2], consider CDN as a service where slices are deployed across multiple cloud domains.In [10], authors investigate replica placement techniques to mitigate arbitrary CDN server failures, with the main objective of minimizing traffic exchange within the CDN while ensuring performance guarantees under server failure scenarios. Other works like [7] examines in-depth distributed algorithms for resource allocation with a focus on learning-based approaches.

Another line of research focuses on energy efficiency in cloud-based vCDN deployments. For instance, [3], examines dynamic scaling mechanisms that deactivate servers hosting vCDN software components, such as replica servers, routers, and firewalls. Similarly, [4] addresses energy optimization in multi-data center CDN infrastructures by dynamically provisioning servers based on traffic demand during off-peak hours while minimizing the number of on/off state transitions.

In contrast to these existing works, our study incorporates the modeling of non-linear energy consumption [11] in vCDN environments while explicitly considering its impact on VNF placement strategies.

III. Problem Definition

A. Network model

We denote N the set of edge servers deployed at both core and aggregation nodes of a mobile network operator infrastructure. We consider S a set of base stations aggregating end-users traffic demands and forwarding them to the edge servers. The edge servers are responsible for computation tasks and content delivery where each edge server $n \in N$ is equipped with a limited number of CPU cores C_n and we consider that all CPU cores operate at the same frequency. We suppose that each base station $s \in S$ is connected to every edge server where each BS can route its aggregated traffic to any edge server for processing.

Each link connecting a base station to an edge server is characterized by a known propagation delay D_{sn}. This delay contributes to the end-to-end latency experienced by the end users. The system operates on discrete time-slot basis. We denote K_s^t the aggregated traffic demands at base station s and time slot t. We consider a set of end users sending their requests to the CDN edge server where the content (i.e., videos) is streamed. We consider that the streaming operation only requires CPU (or vCPU) resources. We make use of these edge servers to dynamically deploy virtualized components of CDN. For simplicity, we consider the possibility of having only one VNF-related CDN on each physical edge server. We consider that this VNF has all the CDN capabilities (e.g., streaming, caching, etc.) and is able to process the requests from all the base stations (i.e., end users) assigned to this physical server. The end-to-end delay for treating a request is composed of the RTT (Round Trip Time) delays from the base station to the edge server, in addition to the streaming (i.e., processing) time at the edge server. This temporal component is known as the session startup time and corresponds to the time interval between the requests initialization and the delivery of the content. We denote t_{byte} the time needed to process one byte of traffic demands on a given edge server. We pose θ as the maximum session startup time that can be accepted, after this deadline we consider that the QoS of the end user is violated. Note that in our work we only consider CPU resources. However, the model can be easily extended to integrate other types of resources (e.g., RAM, disk, etc.).

B. Problem statement

We define our VNF Placement Problem (VNF-PP) as follows. Given a set of edge servers N with limited CPU capacity, a set of base stations S, a set of traffic demands generated from each BS $s \in S$ and for each time slot $t \in T$, a set of VNFs to be deployed to support CDN functions[1]. Our problem consists in determining the (i) set of edge servers on which the VNFs will be deployed, and (ii) the assignment decision of base stations to edge servers where the traffic is processed, so that:

- The end user requests are processed within the tolerated delay, and the CPU capacity is respected;
- The energy consumption at the edge servers is minimized.

IV. NP-hardness Proof

A. Defining a particular case of the original VNF placement problem

We propose *VNF-PP-0*, a particular case of the VNF placement problem described above. We consider a set of edge servers, N, and a set of base stations, S. Each base station s has a traffic demand that needs to be processed by one of the edge servers. In this model, we only consider CPU resources and that edge servers have limited capacity. To simplify the model, we assume that the processing time, y_n^t of the traffic amount associated with base station s, assigned to edge server n, is proportional to the traffic amount. Specifically, y_n^t is equal

[1]Note that for simplicity, we do not model the set of VNFs as we consider that only one VNF-related CDN can be deployed on a given edge server

to traffic demands in bytes divided by the time required to process one byte of data.

Let x_s^t be the number of CPU cores allocated to each base station s at time t.

The objective in *VNF-PP-0* is to minimize the session startup time, which in turn minimizes the propagation delay between the base station and the edge server to which it is assigned. This is achieved by assigning the base station to the nearest available edge server. The key difference between *VNF-PP-0* and the original placement problem is that in the *VNF-PP-0*, the number of CPU cores requested by each base station is known in advance, whereas in the original problem, this parameter becomes a variable that represents the total number of CPU cores requested by all base stations assigned to a given edge server n.

B. Reduction of the Generalized Assignment Problem (GAP) to the VNF-PP-0

The NP-hard GAP can be formally defined as follows. Given a set of agents and a set of tasks, the goal is to assign each task to exactly one agent, where each assignment operation has a cost and a profit. For each agent, the sum of the assignment costs should not exceed the agent budget capacity. If all costs are 1, the problem reduces to the classical assignment problem.

1) Reducing the GAP to the VNF placement problem: Let I=(N,M,A,B) be an instance of the GAP, where N is the set of tasks, M is the set of agents, A is the set of resources required by each agent to perform each task and B is the capacity of each agent.

We now consider the instance I'=(N,M,A,B,T)=(I,T) for the *VNF-PP-0* where N is the set of edge servers, corresponding to the set of agents in the GAP, M are the base stations and correspond to the set of tasks, A represents the resources needed to process traffic demands from a given BS, B is the computing capacity available at each edge server, and T is the set of time slots, which in this case is equal to 1.

For simplicity, we assume that the number of CPU cores needed to process the traffic demands is a known parameter and is directly related to the traffic amount. The objective of the GAP is to minimize the assignment cost c_{nm} where $n \in N, m \in M$. In *VNF-PP-0*, the objective is to minimize the total startup session time, which is composed of the processing time and propagation time. Since the processing time is a fixed parameter, the goal is to minimize the propagation time, which refers to the distance between the agent (edge server) and the task (base station) in a metric-based GAP. Therefore, finding a solution that assigns the traffic demands from each BS to an edge server with minimum delays for T = 1 is equivalent to solving the GAP instance with the same number of agents and tasks. Furthermore, in the original placement problem *VNF-PP*, the objective is to minimize energy consumption, which is modeled as a function of the CPU utilization. The startup session time is treated as a condition, where it must not exceed a specified threshold.

Minimizing energy consumption under these time constraints is computationally equivalent to minimizing the session startup time as the GAP is NP-hard in both its decision and optimization forms.

variables	
x_{nk}^t	binary variable, equal to 1 if the CPU core k is active at edge server n at time slot t
z_{sn}^t	binary variable, equal to 1 if base station s is assigned to edge server n at time slot t
y_n^t	processing time at edge server n at time slot t when one CPU core is used
δ_n^t	continuous variable, represents the maximum achieved session start-up time at server n and time slot t
e_n^t	continuous variable, represents the energy consumed by edge server n at time t
u_n^t	CPU utilization at edge server n and time slot t

TABLE I: Decision variables

V. VNF Placement Model

In the following, we present a mathematical programming model that corresponds to the VNF placement scheme. We use integer variables x_n^t to represent the number of CPU cores allocated to a VNF on the edge server n at time slot t. As already explained, we consider activating only one VNF per edge server to ensure CDN-related functions. We introduce variables z_{sn}^t to represent the assignment decision of the base stations to the edge servers. It takes a value of 1 if BS s is assigned to edge server n at time interval t, 0 otherwise.

Real variables are introduced to model the time components. y_n^t allow to compute the processing time of a VNF deployed at the edge server n at time slot t. The value of y_n^t is proportional to the number of CPU cores allocated and the amount of traffic demands. Variables δ_{sn}^t represent the session startup time, that is, the end-to-end time spent before the end user receives the streamed content from the vCDN edge server. It encompasses both processing time and the propagation time between the base station to which the user is assigned and the edge server. Variables u_n^t denote CPU utilization of the CDN-related VNF deployed at the edge server n at time t. Finally, e_n^t allows to compute the energy consumption as a function of the CPU utilization of server n at time t.

A. Core model constraints

1) Assignement of BSs to MEC hosts: We need to determine the assignment of base stations to the available MEC hosts where each MEC host is responsible for processing the traffic demands of all end users belonging to the same BS. Also, we consider that only one VNF is deployed at the MEC server for vCDN functions. Constraints (1) impose that each base station belongs to exactly one MEC host for each time slot.

$$\sum_{n \in N} z_{sn}^t = 1 \qquad \forall s \in S, t \in T \tag{1}$$

2) Processing time definition: Constraints (2) define the processing time of a VNF as the time needed to process a given amount of data based on the IPC (Instructions executed Per Clock cycle). y_n^t represents the processing time when only one CPU core is active. Note that when multiple CPU cores are active we consider dividing y_n^t by the total number of allocated CPU.

$$y_n^t = t^{byte} \sum_{s \in S} K^{st} z_{sn}^t \qquad \forall n \in N, t \in T \qquad (2)$$

We denote Γ_n^t the processing time when multiple CPU cores are active. Constraints (3) allow to compute it.

$$\Gamma_n^t = \frac{y_n^t}{\sum_{k=0..C_n} x_{nk}^t} \qquad \forall n \in N, t \in T \qquad (3)$$

We introduce a new real variable γ_{nk}^t and we provide a step-by-step linearization of (3) using the constraints (4)-(7).

$$\gamma_n^{kt} \leq \theta x_{nk}^t \qquad \forall n \in N, k \in 0..C_n, t \in T \qquad (4)$$

$$\gamma_n^{kt} \leq \Gamma_n^t \qquad \forall n \in N, k \in 0..C_n, t \in T \qquad (5)$$

$$\gamma_n^{kt} \geq 0 \qquad \forall n \in N, k \in 0..C_n, t \in T \qquad (6)$$

$$\gamma_n^{kt} \geq \Gamma_n^t - (1 - x_{nk}^t)\theta \quad \forall n \in N, k \in 0..C_n, t \in T \qquad (7)$$

Finally, variables Γ_n^t can be written as follows.

$$\Gamma_n^{kt} = \sum_{k \in 0..C_n} \gamma_{nk}^t \qquad \forall n \in N, t \in T \qquad (8)$$

3) End-to-end delay definition: Constraints (9) allow to compute the session start up time. This parameter corresponds to the end-to-end time needed for a client (i.e., base station) to receive the requested content from the edge server. For simplicity, we consider that the propagation time between end users and base stations is negligible, left for future work. Variables $\Delta_{s,n}^t$ represent the maximum end-to-end delay experienced by an end user assigned to edge server n at time slot t.

$$\Gamma_n^t + D_{sn} z_{sn}^t = \delta_{sn}^t \qquad \forall n \in N, s \in S, t \in T \qquad (9)$$

4) Resource capacity constraints: Constraints (10) ensure that the number of CPU cores allocated to a VNF on an edge server n does not exceed its capacity.

$$x_n^t \leq C_n \qquad \forall n \in N, t \in T \qquad (10)$$

5) SLA/QoS constraints: Constraints (11) ensure that the maximum tolerated delay θ is not exceeded.

$$\delta_{sn}^t \leq \theta \qquad \forall n \in N, s \in S, t \in T \qquad (11)$$

6) Server CPU utilization: Constraints (12) estimate the amount of CPU utilization of each VNF. We define it as the ratio of non-idle time (i.e., time spent to process packets) to the total duration of the session Δ of each of the active CPU cores. Here we consider the mean duration over sessions duration.

$$u_n^t = \frac{y_n^t}{\Delta} \qquad \forall n \in N, t \in T \qquad (12)$$

7) Server Power consumption: Constraints (13) estimate the amount of energy consumed by a server to perform its functions based on CPU utilization where P_{idle} and P_{busy} denote the power consumption when the server is idle and fully utilized. We use the non-linear energy model from [11] to track the dynamic evolution of energy where H is the calibration parameter that can be estimated from empirical analysis[2]. we used Special Ordered Set of Type 2 (SOS 2) from cplex to produce a piecewise linear approximation of the term $(u_n^t)^H)$. This approximation may overestimates the actual value. Hence, the subtraction of the term in the energy model can be less than real value; thus, the solver can underestimate optimal value of u depending on the magnitude of the approximation error where the solution may lead to more conservative VNF placement decisions. The selection of breakpoints also affects the solution quality and computational performance.

$$e_n^t = P_{min} + (P_{max} - P_{min})(2u_n^t - (u_n^t)^H) \forall n \in N, t \in T \qquad (13)$$

B. Objective

The objective in (14) aims to minimize the total energy consumed over all the physical servers over all the evaluated time slot, while respecting the maximum tolerated start-up time.

$$\min \sum_{n \in N} \sum_{t \in T} e_n^t \qquad (14)$$

VI. Performance Evaluation

In the following, we describe the traffic demands dataset as well as the different experimental settings that we use to evaluate our proposal, then we provide a performance evaluation of our model compared to a baseline approach.

A. Dataset

The dataset we used encompasses real mobile traffic data collected from a nationwide mobile operator [3]. The dataset is composed of anonymized per-user TCP record sessions and aggregated at the access point level in 10 minutes intervals, for a period of four weeks during the spring of 2024. The geographic focus includes the Paris metropolitan area, and the data covers over 1100 physical base stations.

B. Experimental setup

According to the study in [14], the reference upper bound in terms of the number of edge servers in a metropolitan area such as Paris is estimated to 20 servers[4]. Following this principle in our evaluation, we consider various configurations of edge servers positions, i.e., 20 and 30 servers, based on the geographic coordinates of base stations given in the above dataset. These sizes are chosen to demonstrate representative differences in the observed behavior of the two systems.

[2]H allows to shape the convexity of the power consumption curve w.r.t. CPU utilization.

[3]collected as part of the CoCo5G project: https://coco5g.roc.cnam.fr/.

[4]Considers activating one VM per end user.

Employing the Haversine formula, we generate edge server positions using the K-means clustering technique, where each edge server position is the centroid of a group of base stations.

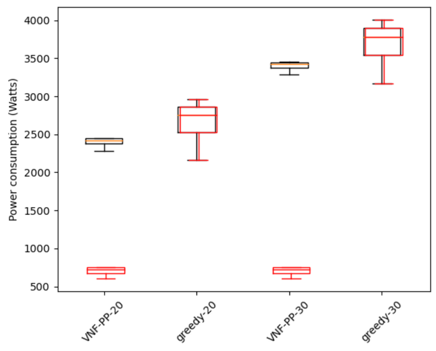

Fig. 1: Distribution of energy consumption - sum value. In black, we consider the energy consumed by servers in idle state. In red, we only consider the energy consumed by the active servers.

The number of available CPU cores on each edge server is randomly varied between 32 and 64 CPU cores, that is, the usual CPU core count in mid-range servers. The minimum and maximum power consumption $Pmin$ and $Pmax$ are set to 100 and 400 Watts, respectively. The processing time needed per byte of data t_{byte} is fixed at 1 ns while the session duration is set to the median value of session duration Δ from the above dataset. The calibration parameter H is set to 1.4. [5] We implement our model in Python and Pulp library using IBM CPLEX solver. Our experiments were conducted on a physical server equipped with 220 GB of RAM and 32 $\times$ 2.4 GHz Intel Xeon CPUs.

C. Compared methods

We use the following approaches to evaluate our proposal.

- VNF-PP: represents the linearized VNF placement MILP model (equations from (1) to (14)), using 20 and 30 edge servers.
- Greedy: a nearest-fit baseline placement algorithm that assigns each base station to its nearest available edge server for each time slot. This algorithm follows the same principle as the GRC algorithm in [13][6], using 20 and 30 edge servers.

The placement problem is solved over a period of 1 day, divided into 24 time slots, each lasting 1 hour.

Regarding the execution time, the VNF-PP achieves an average solving time of 300 seconds per time slots whereas the greedy is faster and produces a solution in less than one second ($\sim$ 0.5 s) for each time slot. Furthermore, the VNF-PP uses a lower number of servers compared to the greedy for both infrastructure sizes. This can be explained by the fact that

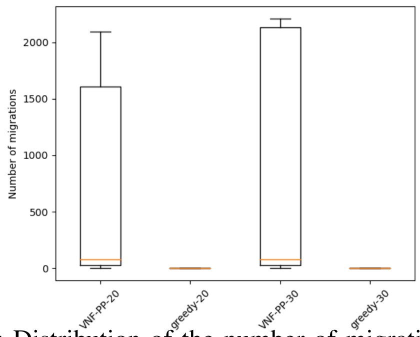

Fig. 2: Distribution of the number of migrations.

VNF-PP tends to use a low number of edge servers as long as the threshold delay is respected. On the other hand, the greedy approach uses a higher number of servers (i.e., total number of available servers) as it aims to find the nearest server for each base station regardless of the total energy consumed.

D. Energy consumption

In Figure 1, we evaluate the power consumption[7] of the two different approaches. We present the distribution of the highest energy consumption over the edge servers during one day and as per 1 hour time slots (i.e., 24 time slots). The VNF-PP achieves lower total energy consumption. This can be explained by the fact that our proposal aims to find a trade-off between the energy consumption and the startup session time (i.e., the propagation and computation time) while respecting the imposed time threshold. In contrast, the greedy approach aims to assign each base station to its nearest available server. When increasing the infrastructure size, both approaches have higher total energy consumption, as idle servers are also accounted (minimum power consumption to stay in the up state).

Furthermore, greedy and VNF-PP have two different behaviors. The greedy has a higher total consumption where the load is balanced across the servers. On the other side, the VNF-PP concentrates the load on a lower number of servers, which results in higher peaks at some servers while keeping the overall energy consumption lower.

E. Number of migrations

In Figure 2 we present the distribution of the total number of migrations, that is, the switching of the assignment of a base station from one edge server to another. The greedy algorithm refers to the lowest number of migrations for all instances. This algorithm assigns base stations to their nearest edge server, resulting in similar assignment plans over time. The only case where a migration operation happens is when the nearest server has no more available capacity pushing the BS to be assigned to the next nearest server. When increasing the number of servers to 30, VNF-PP produces a higher number of migration operations. This can be explained by the fact that when the available capacity increases, it gives more freedom to assign the load of base stations.

[5]If the penalty function H is not calibrated properly, the placement can be adversely affected by under-penalizing moderate CPU utilization, leading to overloaded nodes, forced violations of constraints, increased energy consumption, and higher infrastructure and communication overhead.

[6]https://github.com/GeminiLight.

[7]Here, power and energy consumption are used interchangeably and refers to the power consumption.

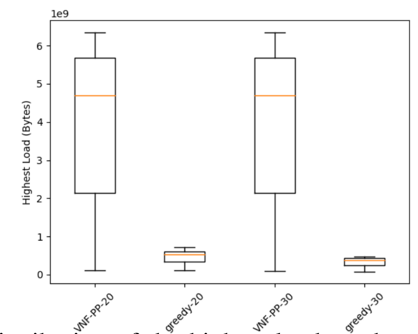

Fig. 3: Distribution of the highest load on the servers.

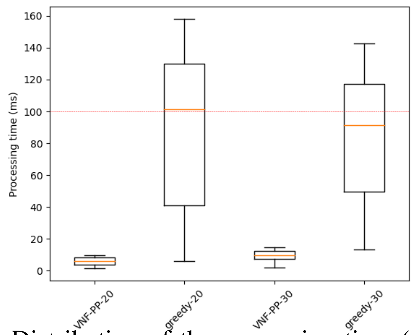

Fig. 4: Distribution of the processing time (ms).

F. Highest load on the servers

Figure 3 presents the highest load achieved on the used servers. The highest values correspond to the VNF-PP: VNF-PP concentrates the loads on a lower number of servers, which results in high CPU utilization at some servers and very low utilization on the rest of the servers. It utilizes a low number of servers to a certain extent while respecting the time threshold, resulting in a higher load concentrated on fewer servers. In contrast, greedy has the objective of minimizing the distance between each BS and the selected server, resulting in a higher number of used servers with a sparser load across the servers.

G. Highest processing time

Figure 4 shows the distribution of the highest processing time over edge servers. For all the infrastructure sizes, VNF-PP has the lowest processing times as a time constraint on the end-to-end session startup time should always be respected and for all the tested instances, put to 100 ms (i.e., represented by a red line). The greedy algorithm has higher processing times. It limits the usage of CPU capacity to process the request as it has no time constraints. Finally, increasing the number of servers improves processing time in the greedy algorithm but can slow it down under VNF-PP. The latter minimizes active CPU cores to minimize energy usage where the load is higher on a per-server basis compared to the greedy approach. The greedy approach on the other hand exploits available servers more freely.

VII. Conclusion

In this paper, we addressed the problem of assigning base stations to edge servers while simultaneously determining the placement of VNFs in the context of virtual CDN over multiple time slots. To assess the computational complexity of our problem, we introduced a complexity proof of a simplified version of the problem where we have shown its NP hardness. We proposed a MILP formulation that models the assignment and placement decisions while introducing a linearized representation of a non-linear energy consumption model. Through preliminary evaluation using a real-world dataset, we have shown the effectiveness of our proposal and its superiority with respect to a state-of-art nearest-fit placement algorithm.

Limitations and Future Work: As solving the proposed MILP comes at a relatively high execution time, further research may explore the integration of pre-processing techniques, such as a clustering algorithm to group together the time slots as a pre-processing step to decrease the computation overhead and improve scalability. The spatial dimension can also be considered where the optimization model can be executed independently across geographically segmented regions, such as per Tracking Area Code (TAC) or administrative zones. Finally, a comprehensive benchmarking including state-of-the-art metaheuristic methods (e.g., genetic algorithms, reinforcement learning) is planned as part of our future work where we could relax existing assumptions, such as permitting multiple VNFs per edge server.

Acknowledgment

This work was fully funded by the ANR CoCo5G (contract nb: ANR-22-CE25-0016) project. We would like to thank Cezari Ziemlicki from Orange for his support.

References

[1] ETSI, ”Network Functions Virtualisation (NFV); Use Cases,” ETSI GR NFV 001 V1.3.1, Mar. 2021.
[2] I. Benkacem, et al., ”Optimal VNFs Placement in CDN Slicing Over Multi-Cloud Environment,” in IEEE Journal on Selected Areas in Communications, vol. 36, 2018.
[3] Liao, Dan, et al. ”Energy-efficient virtual content distribution network provisioning in cloud-based data centers.” in Future Generation Computer Systems 83 (2018).
[4] Ge, Chang, et al., ”Energy-aware data center management in cross-domain content delivery networks.” in IEEE Online Conference on Green Communications 2013.
[5] T. Taleb, et al., ”CDN Slicing over a Multi-Domain Edge Cloud,” in IEEE Transactions on Mobile Computing, vol. 19, no. 9, 2020.
[6] Ibn-Khedher, Hatem, et al. ”OPAC: An optimal placement algorithm for virtual CDN.” in Computer Networks 2017.
[7] Doostmohammadian, Mohammadreza, et al. ”Survey of distributed algorithms for resource allocation over multi-agent systems.” Annual Reviews in Control 59 (2025): 100983.
[8] Papaioannou, Thanasis G., et al. ”Virtual CDN Providers: Profit Maximization through Collaboration.” in IEEE Globecom 2019.
[9] Llorca, Jaime, et al. ”Joint content-resource allocation in software defined virtual CDNs.” in ICCW 2015.
[10] S. U. Khan, et al., ”Robust CDN replica placement techniques,” in IEEE International Symposium on Parallel & Distributed Processing, 2009.
[11] Xiaobo Fan, et al., ”Power Provisioning for a Warehouse-Sized Computer.” In ISCA 2007.
[12] Sun, Gang, et al. ”The cost-efficient deployment of replica servers in virtual content distribution networks for data fusion.” in Information Sciences 432 (2018): 495-515.
[13] L. Gong, et al., ”Toward profit-seeking virtual network embedding algorithm via global resource capacity,” in IEEE INFOCOM 2014.
[14] A. Ceselli, et al., ”Mobile Edge Cloud Network Design Optimization”, in IEEE/ACM Trans. on Networking 25(3):1818-1831, 2017.