Reinforcement Learning Driven Sustainable Resource and Power Management for the MEC

Shreya K. Chari*, John S. Vardakas*†, Kostas Ramantas*, Adlen Ksentini ‡, Christos Verikoukis§¶
* Iquadrat Informatica S.L., Barcelona, Spain
† Department of Informatics, University of Western Macedonia, Greece
‡ Eurecom, Sophia Antipolis, France
§ University of Patras, Patras, Greece
¶ Industrial Systems Institute (ISI) | Athena Research Center, Patras, Greece
*{s.chari, kramantas}@iquadrat.com, †ivardakas@uowm.gr, ‡adlen.ksentini@eurecom.fr, §cveri@ceid.upatras.gr

***Abstract*—With the advent of beyond 5G applications, the execution of computationally intense tasks moves further closer to the network edge. Alongside the capabilities of a Multi-Access Edge Computing (MEC), smart decision-making considering sustainability aspects has become achievable. In this paper, a resource management technique utilizing Reinforcement Learning (RL) at the MEC is presented in order to promote power efficient solutions. *CPU* resources at the MEC are managed and distributed to several network services for their individual disposal. A direct relation between the *CPU* resources and *power* consumption at the MEC is proposed further establishing the need for efficient resource handling. A Soft-Actor Critic (SAC) approach is leveraged to learn the patterns for intelligent resource allocation minimizing the power expenditure. Further, two baseline algorithms, the Knapsack method and the proportional resource allocation scheme, are implemented to prove the dominance of the proposed RL-based algorithm. The results confirm that in the SAC-based RL implementation, the power consumption at the MEC server is lower compared to the two baseline algorithms. The promising results pave way for the deployment of RL-based algorithms for efficient performance, thus promoting green technologies at the MEC.**

***Index Terms*—Green computing, Beyond 5G, Reinforcement Learning (RL), Multi-Access Edge computing (MEC), Resource management, Power management**

I. Introduction

A rapid expansion of networks, heterogeneous services, and devices is expected as the technologies start using Beyond 5G standards. This necessitates adaptable long-lasting solutions with an emphasis on working towards greener computing [1]. Additionally, the identification of the areas within network cloud infrastructure, such as the centralized cloud, edge or the data procurement sources, can aid in determining the energy and power expenditure [2]. Another crucial aspect that can contribute to the performance enhancement of networks is the evaluation of models or algorithms using suitable metrics that can provide deeper insights [3].

MEC plays a crucial role in the efficient management of network resources by carrying out many essential computationally-intense tasks at the network edge while it enables cloud-computing potential for the network closer to the end user [4]. MEC handles cross-functional tasks at the edge such as computation offloading, caching and resource management. Some of the other pressing concerns at the MEC involve network privacy, network reliability and mobile user handover management. Thus, the biggest challenge at the MEC is to distribute, supervise and reallocate network resources efficiently. For example, the power dissipation of a server is directly linked to the number of tasks or services that are being processed in parallel and to their computational requirements. An unexpected rise in the resource demand during unforseen situations such as a large crowd gathering, or the recent global pandemic, puts extraordinary pressure on deployed infrastructures. Given the diversity that is expected in Beyond 5G use-cases in terms of data, devices and traffic, traditional algorithms are expected to perform inadequately [5]. In such a scenario, RL algorithms using the power of Artificial Intelligence (AI) can prove beneficial.

In this paper, a SAC agent which is a part of the RL subclass of algorithms is employed. The trade-off between the management of the *CPU* resources at the MEC and the effect on power consumption is studied. The key features of this work are condensed below:

- A power-consumption aware efficient resource management strategy is formulated as a Markovian Decision Process (MDP).
- The formulated strategy takes into account the *CPU* utilization for delay-sensitive services catered by the MEC. Also, the *CPU* resources are allocated to these services based on the distribution patterns learnt by the SAC algorithm.
- The proposed algorithm along with the system model is implemented as a proof-of-concept. Performance evaluation in terms of average delay and average power consumed by the MEC server and the services under consideration is further conducted.
- The results prove the superiority of the RL-based resource manager over the two implemented baselines on comparison.

The paper ahead is arranged as follows. Section II points out some of the contributions from the state-of-the-art work in this domain. In Section III, the system model used for the experiments is explained. Section IV lays out the targeted constrained optimization problem. The MDP based problem

formulation is derived in Section V. Section VI lays the details for the conducted experiments. The results are examined in Section VII before concluding the work in Section VIII.

II. Related Work

Numerous efforts have been made in the direction of balancing energy and power dissipation at data centers. Optimization models both on software and hardware levels have furnished significant outcomes. Some of these works are mentioned in brief here on.

The authors in [6] minimize energy by making use of a customized process that can classify task requests arriving at the MEC host server. A multi-constrained energy efficient optimization model is proposed by the authors in [7] for Industrial Internet-of-Things (IIoT) networks. The research work in [8] suggests a joint partial-offloading and power allocation algorithm positioned on Lagrangian-constrained optimization and Johnson method. With this approach, the authors attempt to optimize energy consumption and task execution delay. The authors in [9] utilize a binary-search water-filling algorithm for maximising resource allocation to a power-constrained MEC Internet-of-Things (IoT) network. A cost-effective edge allocation algorithm is proposed by the authors in [10] for a MEC with limited power capacity.

The research idea in [11] uses Deep Reinforcement Learning (DRL) to allocate resources to multimedia broadband services with dense traffic. Deep Deterministic Policy Gradient (DDPG) is applied by the authors of [12] to optimize efficient resource allocations and offload computation cooperatively. Computation offloading and minimising the execution of tasks queue using actor-critic learning while considering energy constraints is the primary focus of the work presented in [13]. In [14], a Deep Q-Network (DQN) optimizes the user offloading data ratios and MEC-computational resources while promoting green computing.

From the aforementioned works, it is noted that a system-level approach assessing the relation between different services within a MEC is yet to be accounted for. Each network resource being managed at the MEC impacts the power consumption in their own way. At the same time, network performance metrics such as the delay has to be monitored continuously without deterioration. With multiple objectives to be achieved, using AI algorithms to establish a relation between the resources and the services at the MEC seems fitting. Hence, this paper sets the groundwork for an advanced scalable AI-based solution that can adjust to growing user preferences and shared network resource requirements.

III. System Model

A communication network framework that can encapsulate the broad system level framework spanning from the MEC connected to the core cloud to the end users at the RAN domain was required for the proposed methodology. As a result, the system model in Fig. 1 is considered for realizing the experiments described in later sections.

A single MEC server M hosts integral offloaded network

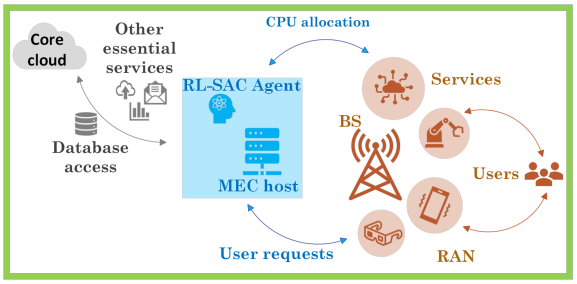

Fig. 1. System model

service s that belongs to a set of broader network services S and supplies computational resources to applications such as Virtual Reality (VR), smart homes, AI/ML models at the end devices and remote control robots. The MEC server M further holds the RL agent A that consequently interacts with the services $s \epsilon S$ to learn the network demands. At time t, the services $s \epsilon S$ place their resource demands in terms of CPU requirements to the MEC M. The CPU needs per service $s \epsilon S$ requires c computational resources, while C is the total computational capacity of the server. Computational resources c at MEC server M is expected to vary dynamically attributing to the dynamic behaviour of the tasks offloaded by the end users. To avoid shifting the focus from resource management at the MEC M, all the services in $s \epsilon S$ are admitted into the MEC M. Conversely, none of the $s \epsilon S$ is left without resource allocation as it would violate the unspoken assumption which is that the services or tasks are offloaded to M because it had the computational capacity to host. However, resource provision to each of the individual resource demand put forth by $s \epsilon S$ is based on the decisions made by the RL agent A.

IV. Problem Formulation

A. Key Performance Indicators

Task execution delay incurred by the services and power consumption is used in this work as the Key Performance Indicators (KPI) for monitoring the performance of the MEC server M. The delay is calculated using the equation, $d = g\frac{c_s}{c_a}$. Here g is the gain factor which controls the impact of resource demand for a service c_s and allocated resource c_a. On the other hand, power consumption is modelled as $P = P_{idle} + (P_{max} - P_{idle})\frac{c_s}{c_a}$ [1]. The assumption here is that the MEC server M consumes power even when its not managing services $s \epsilon S$. P_{idle} is the power consumed in the idle state and P_{max} is the power consumed when its catering to the services. The values for P_{idle} and P_{max} can vary as they follow a normally distributed curve with mean values selected as per the specifications in [15]. Both the KPIs are selected after careful search within the state-of-the-art as observed in the works [1] and [16]. The parameters considered while calculating the KPI's are mentioned in Table I.

B. Constrained resource and power management

The resource and power management at the MEC M is posed as an optimization function $O(t)$, which is expressed

TABLE I
KPI PARAMETERS

Delay		
Gain factor	0.1	
Power (W)		
	Mean	*Stdev*
Max	700	200
Idle	150	50

by (1). As introduced in the system model, $s \epsilon S$ is the finite number of services hosted by the MEC M with a preference value w_s. These preference values w_s are based on the prioritization of the service that is, if some services need to be prioritized over the others if the situation arises. We aim at maximizing the difference between the CPU resources represented as R_s and the power consumption P_s for all S services.

$$O(t) = \max \sum_{s=1}^{S} w_s (R_s - P_s)(t) \tag{1}$$

The objective function in (1) is followed by a set of restrictions. Equation (2) refers to the the overall resource capacity of the server M. The CPU resources c_a allocated to different services cannot exceed C which is the maximum CPU distribution capacity of the MEC.

$$\sum_{s=1}^{S} c_a \leq C \tag{2}$$

Resource over-allocation is monitored by equation (3). In addition, L_s is the incoming resource demand load from $s \epsilon S$ and R_s is the allocated resources. α is the permissible resource satisfaction parameter used for overseeing resource over-allocation.

$$L_s \leq \alpha R_s \tag{3}$$

The KPI's presented previously are assumed to be associated with Service Level Agreements (SLA). Additional constraints including the KPI's are added to keep the objective realistic. Each service inside the MEC M produces a delay d_s within the network as in (4). The delays d_s are expected to be less than the maximum acceptable limit D_s for each individual service. The values for D_s are known from accessible communication network standards [17]. Similarly, the total power consumed by each individual service $\sum_{s=1}^{S} p_s$ needs to be proportional to the capacity limit C and power limit P of the server. The consumption index $\frac{R_s}{C}$ ensures that the services using fewer resources of the total capacity C are not the ones contributing the most to the power limit P. It is to be noted that the values for d_s and p_s are calculated as per the KPI section defined previously.

$$d_s \leq D_s \tag{4}$$

$$\sum_{s=1}^{S} p_s \leq \frac{R_s}{C} P \tag{5}$$

V. MEC Resource and Power Management

A. RL problem formalized as MDP

Every RL agent has to interact iteratively with an environment to learn the agent's RL policy. In this work, the environment is formulated as a MDP in the form of a tuple $(S_t, A_t, T_t, R_t, \gamma)$. The agent interacts with a continuous state space $s_t \epsilon S_t$ and makes continuous action decisions $a_t \epsilon A_t$. The agent moves from one state s_t to the next state s_{t+1} knowing the actions A_t while relying on the probability transition function $T(t) = P(s_{t+1}|s_t, a_t)$. The reward received by the agent on proceeding with an action a_t is represented as $r_t \epsilon R_t$, where R_t is set of immediate rewards awarded. The influence of the reward r_t is determined by the discount factor γ. The discount factor γ can vary between 0 and 1, a value closer to 1 implies that the future rewards hold as much importance as the immediate rewards. The state space, action space and reward function used for the work in this paper are defined below:

- **State space:** The state space at time t comprises of the maximum CPU capacity of the server C. It further contains CPU demand c_s, CPU allocation c_a, the delay incurred d_s and the power consumed p_s by each service $s \epsilon S$.
- **Action space:** The action space comprises of the random CPU allocations for each service $s \epsilon S$ that the RL agent makes given the state space S_t and the transition probability $T(t)$ as defined earlier.
- **Reward:** The reward function r_t is dependent on several components to guide the RL agent's policy learning, as described in (6). The first term of (6) is in favour of the rewarding the agent every time it takes the correct action. The function $f(c_d)$ is dependent on the difference between the allocated resources c_a and the resource demand c_s for the services $s \epsilon S$. Within the function f the difference $c_d = c_a - c_s$ is multiplied with appropriate preference value w_s as introduced in the problem formulation. The second term of (6) which is $\sum \Omega(c_s, d_s, p_s)$ is used for punishing the agent every time it violates any of the constraints mentioned in the problem formulation. $\Omega(c_s, d_s, p_s)$ is a function dependent on the resource demand c_s, delay d_s and power consumed p_s as per the set constraints. Just like the rewarding factor, within $\Omega(c_s, d_s, p_s)$ there are breach values multiplied to respective constraints based on the service preference.

$$r_t = \frac{f(c_d)}{S} - \sum \Omega(c_s, d_s, p_s) \tag{6}$$

B. Intelligent solution using RL

A perceptive model-free RL agent that works in continuous space is needed to solve the problem of interest in this work. SAC maximizes the objective function $J(\theta)$ represented by $J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[\sum_{t=0}^{\infty} \gamma^t (r(s_t, a_t) + \alpha H(\pi_\theta(.|s_t)]$ to learn the policy $\pi_\theta(a_t|s_t)$. It further uses entropy regularization in the form of $\alpha H(\pi_\theta(.|s_t))$ while α controls the regularization. $\gamma^t(r(s_t, a_t))$ is the immediate reward to be gained with γ,

TABLE II
SIMULATION PARAMETERS

Parameter	Value
Learning rate	0.0001
Optimizer	Adam
Activation function	ReLu
Discount factor (γ)	0.9
Trade-off coefficient (α)	0.9
Batch size	64
Episodes	20
Steps	1024

the discount factor driving it. SAC proves to be advantageous as it brings the much required stability by exploiting two Q-network $Q(s_t, a_t)$ approximators [18]. Hence, SAC is chosen as the RL agent for making resource allocation decisions based on the computation demand behaviour set forth by the network services.

VI. EXPERIMENTS

A. Implemented Framework

The system model as explained in Section III was simulated using Python 3.11 [19]. The RL-SAC agent implementation was completed using Tensorflow 2.15 [20]. The specific parameters used to tune the agent are referred to in Table II. A dataset that is suitable for the system model and for the RL agent was required. The Google cluster usage-traces proved useful in training the RL agent [21]. Google's data centers information is classified into different tasks with different priorities along with their CPU utilization values. The CPU utilization and task classification is of particular interest to this work to emulate different services being hosted by MEC M. Reasonable pre-processing steps and assumptions have been made throughout the simulations as some of the parameters related to data center's information is confidential and is not provided by Google or any similar enterprise. Nevertheless, the dataset satisfies the requirements of this work and aids in completing the proof-of-concept simulations.

B. Experiments performed

The RL SAC agent is trained for 20 episodes before the agent reaches a learning saturation. An episode in a RL algorithm signifies for how long the agent interacts with the environment and the dataset. In this work, each episode was further divided into 1024 sub episodes in the form of steps. The agent then draws the necessary state S_t, next state S_{t+1}, action A_t and reward R_t values from the replay buffer to train on. The details of the implementation is provided in the pseudo-code algorithm 1 [18]. For performance comparison of the algorithm 1, two baselines are also additionally implemented. For the simulations, 4 network services are hosted at the MEC M. Each of these services have diverse delay requirements as established in the system model section. The average delay experienced and the average power consumption by each of the service is observed for the last 5 episodes in the results section. The overall average power consumption of the MEC server M is also calculated.

Algorithm 1: SAC training for the MEC

Initialize replay buffer $\mathcal{D}$, policy parameters θ
Set target networks ϕ_1, ϕ_2
for *episode=1 to N* **do**
  Reset environment and get initial state S_t
  for *steps=1 to M* **do**
    Select $a_t \epsilon A_t$
    Gather response from the environment for a_t
    Collect next state s_{t+1}, reward r_t, done sign d
    Store $(s_t, r_t, a_t, s_{t+1}, d)$ in the replay buffer $\mathcal{D}$
    if *sampling step* **then**
      Randomly sample a minibatch
      $\mathcal{B} = \{(s_t, r_t, a_t, s_{t+1}, d)\}$
      Calculate the Q-function
      $r_t + \gamma(min\mathcal{Q}(s_{t+1}, a_{t+1})) - \alpha log\pi_\theta(a_{t+1}|s_{t+1})$
      Perform gradient descent and update
       Q-function
      $L \leftarrow \frac{1}{|B|}\Sigma((\mathcal{Q}_\phi(s_t, a_t) - y(r, s_{t+1}))^2$
      Update policy
      $\frac{1}{|B|}\sum_{s \epsilon B}(min\mathcal{Q}(s, a_\theta(s))) - \alpha log\pi_\theta(a_\theta(s)|s)$
      update target network parameters
      $\phi_{1,2}^{targ} \leftarrow \tau\phi_{1,2}^{targ} + (1-\tau)\phi_{1,2}$
    end
  end
end

C. Baselines

The proposed RL algorithm in this work is compared against the following two baseline algorithms:

- **Knapsack problem:** The Knapsack problem is a widely used approach for resource allocation in general. Within the Knapsack approach, a set of objects with associated weights have to be packed adhering to a capacity limit. It serves the purpose for the work in this paper as a resource management problem with preferences is being solved. The authors of [22] and [23] have made use of the Knapsack algorithm for solving network edge-related issues thus emphasizing on its relevance.
- **Proportional distributor:** In this approach, a randomly chosen service from all the services hosted by the MEC is assigned resources proportional to their preference values. This method ensures that the services with higher preference value is assigned resources before the others. Energy proportionality has been encouraged for a significant time such as in [24] and boosts prioritization of tasks at the edge.

VII. RESULTS

Average delay and average power consumption at the MEC is used as the main KPIs for evaluating the effectiveness of the suggested algorithm in this work. To begin with, in Fig. 2, the average delay experienced by each of the 4 services being hosted at MEC M is calculated as presented in the KPI section. Clearly, the RL-SAC algorithm incurs extremely small

delay compared to the Knapsack and proportional distributor algorithms. The results observed affirms that the underlying cause for better delay values while using RL-SAC can be directly attributed to a remarkably better resource allocation scheme. Extremely stable SAC agent is able to dynamically learn policies that can vary along with the varying heterogeneous network loads compared to the other two algorithms.

The results in Fig.3-Fig.7 are related to the power consumption at the MEC M. The average power consumption by each service $s \epsilon S$ is being shown in Fig.3. A similar pattern as the average delay is observed where the RL-based solution performs better by enabling the services to consume lesser power in comparison to the others. Identically, the resource allocation strategy is learnt using sophisticated Q-networks within the SAC agent which are far more advanced than the Knapsack or proportional distributor's decision making strategy. Moreover, the experience replay buffer included in the SAC algorithm exploits the reward-punishment mechanism by favouring the samples with higher rewards.

In Fig.4-Fig.7 the power consumption by each individual service $s \epsilon S$ is shown respectively. The power consumption is shown with respect to the *CPU* demands that were put forward by these services. Further, the *CPU* demands are in the form of utilization ratio and are divided into different bins for representation purposes. It is observed that for each demand bin the RL-SAC gives a lower power consumption value compared to the baselines. A separate bin when the *CPU* utilization is zero is added, to highlight the fact that an idle server can consume power as well. Hence, it can be concluded that the proposed resource and power management algorithm in this work is not only able to bring down the overall power consumption at the MEC but also at an individual service level as well.

The results achieved strengthens the rationale that there is a strong correlation between the network resource distribution and power consumption at the edge server. A methodical resource allocation policy not only reduces the power consumption but also avoids over provisioning of resources. There can be an overall reduction in the operational costs for the operators as the expenditure on actively maintaining and cooling large-scale server clusters will go down. In addition, the results achieved also solidifies the hypothesis that AI technologies such as the one in this work are the prospective solution in the times ahead.

VIII. Conclusions

An AI-based solution to resolve power efficiency at the network edge is proposed in this paper. An algorithmic implementation of a RL-based system model inclusive of the major components of a network infrastructure is used for validation. The implementation revolves around the fundamental notion of building a link between the edge server's *CPU* resources and the power consumption by the services hosted by it. RL-SAC exhibits an enhanced performance at the network edge server in terms of *CPU* resource management and power management in comparison to the standard Knapsack and proportional distributor algorithms. The significant performance improvement particularly in terms of power consumption while using the RL based SAC agent emphasizes the importance of system-

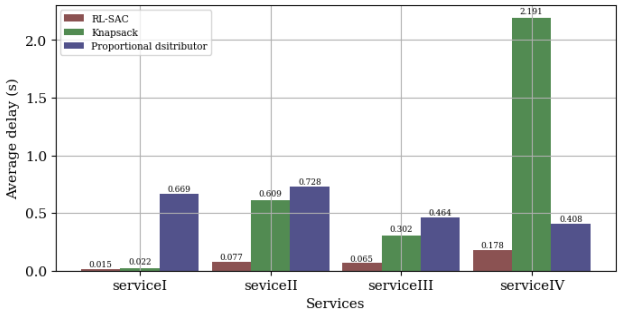

Fig. 2. Average delay experienced by hosted services at the MEC

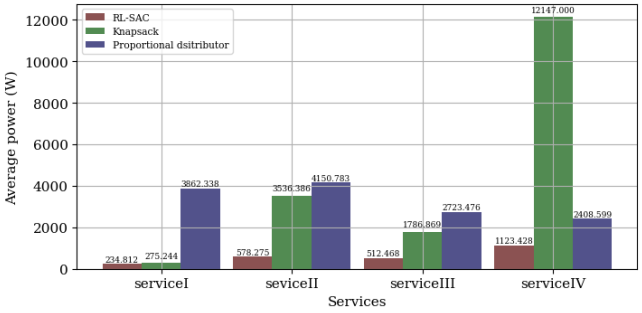

Fig. 3. Average power consumption of hosted services at the MEC

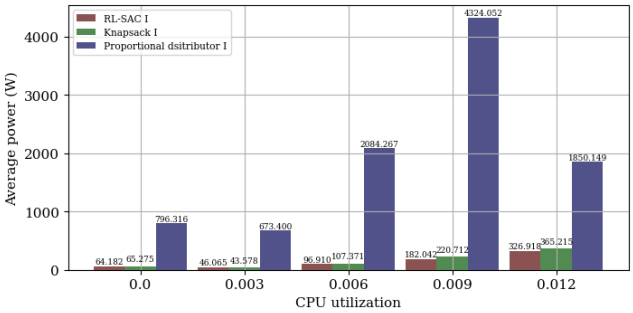

Fig. 4. Average power consumption of service I at the MEC

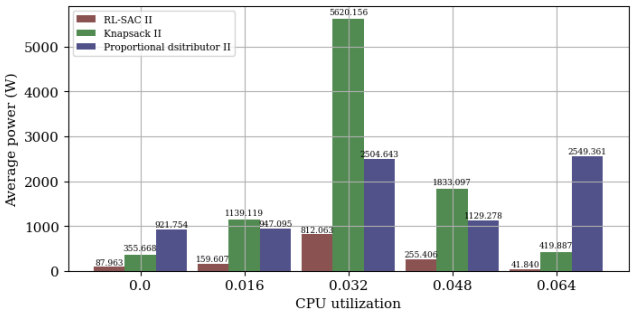

Fig. 5. Average power consumption of service II at the MEC

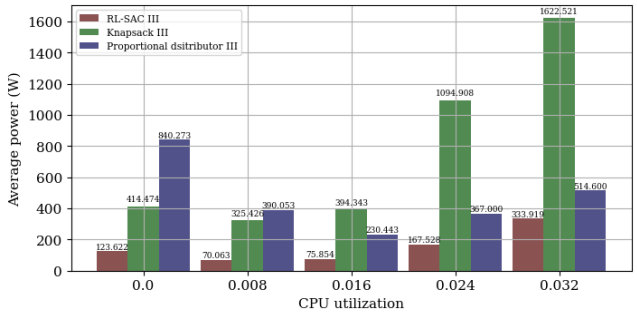

Fig. 6. Average power consumption of service III at the MEC

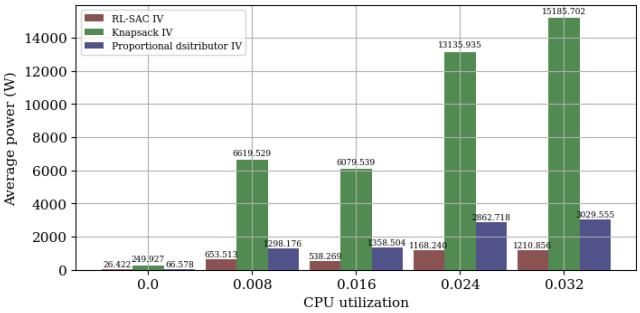

Fig. 7. Average power consumption of service IV at the MEC

atic and smart resource management. For future work, server maintenance power, minimising the idle and migration power can be considered [25]. Resource requirement data collected from other cloud service providers can also be experimented with to further enhance performance. Further, a multi-MEC environment involving components from the RAN domain can be experimented to have a broader point of view. Additionally, AI technologies such as explainable AI and distributed RL can also be attempted.

IX. Acknowledgement

This research was supported by the Research and Innovation Program under the AC3 (Agile and Cognitive Cloud-edge Continuum management) project (Grant No. 101093129).

References

[1] Avita Katal, Susheela Dahiya, Tanupriya Choudhury, "Energy efficiency in cloud computing data center: a survey on hardware technologies," in Cluster Computing, 25 1-31, 2022.

[2] M. Dayarathna, Y. Wen and R. Fan, "Data Center Energy Consumption Modeling: A Survey," in IEEE Communications Surveys & Tutorials, vol. 18, no. 1, pp. 732-794, First quarter 2016.

[3] Xiaotong Shao, Zhongbin Zhang, Ping Song, Yanzhen Feng, Xiaolin Wang, "A review of energy efficiency evaluation metrics for data centers," in Energy and Buildings, 271, 112308, 2022.

[4] Multi-access Edge Computing (MEC). [Online]. Accessed: March 24, 2024. Available: https://www.etsi.org/technologies /multi-access-edge-computing.

[5] M. Elsayed and M. Erol-Kantarci, "AI-Enabled Future Wireless Networks: Challenges, Opportunities, and Open Issues," in IEEE Vehicular Technology Magazine, vol. 14, no. 3, pp. 70-77, Sept. 2019.

[6] S. Thananjeyan, C. A. Chan, E. Wong and A. Nirmalathas, "Energy-Efficient Mobile Edge Hosts for Mobile Edge Computing System," 2018 IEEE International Conference on Information and Automation for Sustainability (ICIAfS), Colombo, Sri Lanka, 2018, pp. 1-6.

[7] D. Jiang, Y. Wang, Z. Lv, W. Wang and H. Wang, "An Energy-Efficient Networking Approach in Cloud Services for IIoT Networks," in IEEE Journal on Selected Areas in Communications, vol. 38, no. 5, pp. 928-941, May 2020.

[8] L. Li, Z. Kuang and A. Liu, "Energy Efficient and Low Delay Partial Offloading Scheduling and Power Allocation for MEC," ICC 2019 - 2019 IEEE International Conference on Communications (ICC), Shanghai, China, 2019, pp. 1-6.

[9] M. Qin, L. Chen, N. Zhao, Y. Chen, F. R. Yu and G. Wei, "Power-Constrained Edge Computing With Maximum Processing Capacity for IoT Networks," in IEEE Internet of Things Journal, vol. 6, no. 3, pp. 4330-4343, June 2019.

[10] M. Song, Y. Lee and K. Kim, "Reward-Oriented Task Offloading Under Limited Edge Server Power for Multiaccess Edge Computing," in IEEE Internet of Things Journal, vol. 8, no. 17, pp. 13425-13438, 1 Sept.1, 2021.

[11] Y. Huo et al., "DRL Driven Energy-efficient Resource Allocation for Multimedia Broadband Services in Mobile Edge Network," 2020 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), Paris, France, 2020, pp. 1-6.

[12] X. Feng, J. Jia, T. Wang, J. Li, H. Xu and Q. Li, "Deep Reinforcement Learning based Energy Efficient Resource Allocation for Wireless Powered Edge Computing Network," 2023 4th International Symposium on Computer Engineering and Intelligent Communications (ISCEIC), Nanjing, China, 2023, pp. 144-148.

[13] J. Zhang, J. Du, C. Jiang, Y. Shen and J. Wang, "Computation Offloading in Energy Harvesting Systems via Continuous Deep Reinforcement Learning," ICC 2020 - 2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 2020, pp. 1-6.

[14] Y. Yang, Y. Hu and M. C. Gursoy, "Deep Reinforcement Learning and Optimization Based Green Mobile Edge Computing," 2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 2021, pp. 1-2.

[15] Standard Performance Evaluation Corporation. Accessed: Mar. 24, 2024. [Online]. Available: http://www.spec.org/power_ssj2008/results/

[16] C. Restif, N. Ponomareva and K. Ostrowski, "A Classifier for the Latency-CPU Behaviors of Serving Jobs in Distributed Environments," 2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom), Vancouver, BC, Canada, 2015, pp. 123-130.

[17] 3GPP, 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects, Service requirements for cyber-physical control applications in vertical domains, 2023, 3GPP TS 22.104 V19.1.0 (2023-09).

[18] Soft Actor Critic. [Online]. Accessed: March 21, 2024. Available: https://spinningup.openai.com/en/latest

[19] Python. [Online]. Accessed: March 22, 2024. Available: https://www.python.org/

[20] Tensorflow. [Online]. Accessed: March 22, 2024. Available: https://www.tensorflow.org/

[21] Google cluster-usage traces v3. [Online]. Accessed: March 22, 2024. Available: https://github.com/google/cluster-data/blob/master/ClusterData2019.md

[22] Vaibhav Tiwari, Chandrasen Pandey, Abisek Dahal, Diptendu Sinha Roy, Ugo Fiore, "A Knapsack-based Metaheuristic for Edge Server Placement in 5G networks with heterogeneous edge capacities," in Future Generation Computer Systems, Volume 153, 2024, Pages 222-233, ISSN 0167-739X.

[23] C. -Y. Hsieh, Y. Ren and J. -C. Chen, "Edge-Cloud Offloading: Knapsack Potential Game in 5G Multi-Access Edge Computing," in IEEE Transactions on Wireless Communications, vol. 22, no. 11, pp. 7158-7171, Nov. 2023.

[24] D. Lo, L. Cheng, R. Govindaraju, L. A. Barroso and C. Kozyrakis, "Towards energy proportionality for large-scale latency-critical workloads," 2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA), Minneapolis, MN, USA, 2014, pp. 301-312.

[25] Felipe Rubin, Paulo Souza, Tiago Ferreto, "Reducing Power Consumption during Server Maintenance on Edge Computing Infrastructures," in Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing (SAC '23), Association for Computing Machinery, New York, NY, USA, 691–698, 2023.