Distributed Learning for Application Placement at the Edge Minimizing Active Nodes

Claudia Torres-Pérez*, Estefanía Coronado*†, Cristina Cervelló-Pastor‡,
Juan Sebastian Camargo*, and Muhammad Shuaib Siddiqui*
*i2CAT Foundation, Barcelona, Spain;
Email: {claudia.torres, estefania.coronado, juan.camargo, shuaib.siddiqui}@i2cat.net
†Universidad de Castilla-La Mancha, Albacete, Spain; Email: estefania.coronado@uclm.es
‡Universitat Politècnica de Catalunya, Castelldefels, Barcelona, Spain; Email: cristina.cervello@upc.edu

***Abstract*—The main goal of application placement in Multi-Access Edge Computing (MEC) is to map their requirements to the infrastructure for desired Service Level Agreement (SLA). In highly distributed infrastructures in beyond 5G and 6G networks, meeting this need and minimizing energy use are crucial. Focusing solely on meeting SLA requirements can lead to resource fragmentation and reduced energy efficiency, as nodes utilize only a small portion of their resources. Furthermore, when multiple orchestrators govern MEC nodes, achieving optimal efficiency becomes a more complex challenge. This paper addresses the application placement problem by employing distributed deep reinforcement learning to efficiently minimize the overall cost of active MEC nodes in a distributed scenario involving multiple MEC systems. Our technique reduces the number of active nodes maintaining an average accuracy of up to 98%, meets SLA requirements, and is scalable for hosting several MEC nodes.**

***Index Terms*—application placement, edge computing, distributed learning, 5G, 6G, active nodes**

I. Introduction

Efficient application placement is crucial in Multi-access Edge Computing (MEC), especially in highly distributed scenarios. It involves mapping application components and links onto an infrastructure graph (i.e., computing devices and physical edges) [1]. The goal is to meet Quality of Service (QoS) requirements and optimize system resource utilization. However, focusing solely on QoS can lead to resource fragmentation as the number of nodes increases. To reduce costs, an alternative is minimizing active nodes to effectively lower energy consumption and satisfy resource demands. The presence of standby nodes in the infrastructure generates minimal additional expense, leading to decreasing hardware deployment and maintenance. Furthermore, inactive nodes can be activated in case of failures to accommodate the affected workload. In 6G networks, the complexity and geographical distribution of edge nodes will increase, requiring multiple MEC orchestrators to efficiently handle requests and prevent bottlenecks in different regions or domains. Population density and application requirements can vary across areas (e.g., industrial zones vs. city centers). This necessitates designing resource management algorithms specific to application types or locations, considering the diverse data and requests of each orchestrator. However, the decisions of such orchestrators would be far from optimal if applications with different requirements were to be run in that network area. The question is: *"How can application placement be tackled in distributed environments while minimizing the number of active nodes at the edge infrastructure?"*.

To reply to this question, in this paper, we propose an approach for application placement to ensure the availability of resources based on Distributed Deep Reinforcement Learning (DDRL), following a reward function given by the infrastructure telemetry (i.e., storage, RAM, and CPU resources). We propose a novel approach for MEC applications placement in highly distributed environments in beyond 5G and 6G networks, aiming to minimize the number of active edge nodes while meeting user requirements. We implement the algorithm using DDRL, in a multiple MEC system scenario fully compatible with ETSI MEC reference architecture. We evaluate the algorithm using the Simu5G network simulator [2] and extend it to introduce and implement a control federation interface that enables communication between MEC orchestrators.

II. Related work

Application placement has been a research topic in the last few years via several approaches, such as mathematical optimization and Machine Learning (ML) methods. Following the former approach, a multi-component application placement is presented in [3] to minimize the cost of running applications while the resources are allocated over an area's distributed edge servers. Conversely, the authors of [4] address the problem of Artificial Intelligence Function placement using Mixed-integer linear programming in a federated learning setting.

Reinforcement Learning (RL) and its variants demonstrate superior performance in unknown environments over heuristic methods [5]. Most existing RL works on application placement focus on centralized approaches, neglecting geographically distributed edge nodes managed by different orchestrators. For instance, the work in [6] proposes a centralized, offline multi-objective RL algorithm for application placement, aiming to minimize network impact. Goudarzi et al. employ a distributed learning approach for application placement optimization, considering Fog/Edge servers [7].

Existing works ignore the complexities of multiple MEC systems with different orchestrators and lack insights into

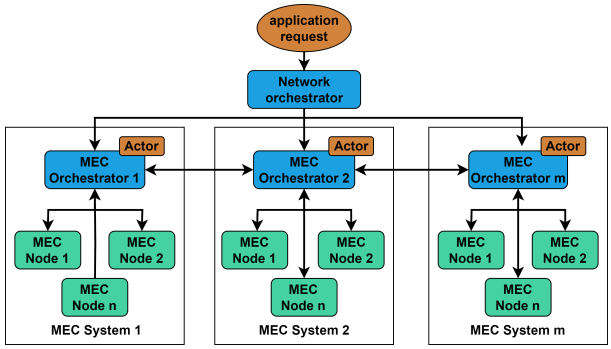


Fig. 1. Distributed MEC System architecture considered.

standardization challenges related to the ETSI MEC architecture [8]. In this regard, the authors of [9] focus on service placement load balancing in ETSI MEC architecture but without looking at cross-orchestrator communication.

To the best of our knowledge, no prior works address distributed application placement with different orchestrators in ETSI MEC-aligned environments, focusing on minimizing active edge nodes while maintaining application requirements and domain-specific data locality using DDRL approaches.

III. System Model and Problem Formulation

Figure 1 presents the system model envisioned in this work, based on a variation of the ETSI MEC architecture [8] to account for distributed MEC systems. It consists of a network orchestrator at the top level, and each MEC system comprises a MEC Orchestrator responsible for orchestration functions within the MEC system. The internal ETSI MEC reference architecture is considered in this work. However, it is omitted in Fig. 1 for readability purposes.

The network orchestrator receives an application request and forwards it to a specific MEC system. The application descriptor contains storage, RAM, and CPU requirements. The placement requires node availability in all three parameters. The primary objective is to instantiate applications while minimizing node utilization. The MEC Orchestrator, supported by a DRL model known as a worker or actor, makes management decisions within MEC nodes. The model's output represents the chosen MEC node for application placement. Since the proposed ML method is designed at the MEC system level, the training process assumes that the specific MEC system to which an incoming application request is forwarded has already been selected at the orchestration level. The interface communicating the MEC orchestrators is defined by ETSI in [10]. We include this control interface in our work to allow the DRL's parameters to be exchanged across the network.

The mathematical formulation of the application placement problem considers multiple MEC systems containing several nodes. Without loss of generality, we bear in mind that each MEC has the same number of nodes.

Let $\mathcal{M}$ denote the number of MEC systems in our network architecture and $\mathcal{N}$ the number of nodes of each one. Each node $n \in \mathcal{N}$ of each MEC system $m \in \mathcal{M}$ has available resource capacity associated with storage, RAM, and CPU, $C_{n,m} = (C^{ST}_{m,n}, C^{RAM}_{m,n}, C^{CPU}_{m,n})$. Furthermore, let $\mathcal{R}$ be a set of application requests arriving at the Network Orchestrator. Each request $r \in \mathcal{R}$ demands resource capacity related to storage, RAM, and CPU, $D_r = (D^{ST}_r, D^{RAM}_r, D^{CPU}_r)$.

We define an Integer Linear Program (ILP), called One-App-CPP, to place applications iteratively by selecting nodes with sufficient resources following a Poisson process. The objective is to optimize application placement maximizing successful allocations. Being z a binary variable to determine if the current request is placed and $x_{m,n}$, a binary variable indicating if the current request is successfully deployed at node $n \in \mathcal{N}$ of MEC system $m \in \mathcal{M}$, the constraints of the problem are the following. First, we establish that the capacity demand of the current request $D = (D^{ST}, D^{RAM}, D^{CPU})$ cannot exceed the total capacity in terms of storage, RAM, and CPU of the node $n \in \mathcal{N}$ of MEC $m \in \mathcal{M}$ at which it will be located:

$$\text{if } x_{m,n} = 1 \Rightarrow \begin{cases} D^{ST} \leq C^{ST}_{m,n} \\ D^{RAM} \leq C^{RAM}_{m,n} \\ D^{CPU} \leq C^{CPU}_{m,n} \end{cases} \quad \forall n \in \mathcal{N},\ m \in \mathcal{M} \tag{1}$$

Due to the dependency between x and z, a relationship between them must be added, having the following implications:

$$\begin{aligned} &\text{if } x_{m,n} = 1 \Rightarrow z = 1, \\ &\text{if } \sum_{\forall m \in \mathcal{M}} \sum_{\forall n \in \mathcal{N}} x_{m,n} = 0 \Rightarrow z = 0. \end{aligned} \tag{2}$$

With all of these considerations, the One-App-CPP model is the following:

$$\begin{aligned} \text{Maximize} \quad & z \\ \text{s.t. } x_{m,n} \ &\leq\ 1 + \frac{C^i_{m,n} - D^i}{max(C^i_{m,n})} && \forall m \in \mathcal{M},\ \forall n \in \mathcal{N} \\ & && i \in \{ST, RAM, CPU\} \\ x_{m,n} \ &\leq\ z && \forall m \in \mathcal{M},\ \forall n \in \mathcal{N} \\ z \ &\leq\ \sum_{\forall m \in \mathcal{M}} \sum_{\forall n \in \mathcal{N}} x_{m,n} \\ x_{m,n},\ z \ &\text{ binary} && \forall m \in \mathcal{M},\ \forall n \in \mathcal{N}. \end{aligned}$$

Figure 2 illustrates the iterative algorithm using the One-App-CPP model, triggered for each request arrival. Step 1 initializes the MEC infrastructure's total capacity. Steps 2 and 3 iterate to determine the placement of each application request, updating and storing available MEC resources for later analysis. The iteration concludes with the final arrival.

IV. Distributed Deep Reinforcement Learning Model For Active Edge Nodes Minimization

This section details the DRL model's role in each MEC system during training and inference, along with the DDRL technique used for distributed training. In DDRL training, a DRL model is placed together with each MEC Orchestrator,

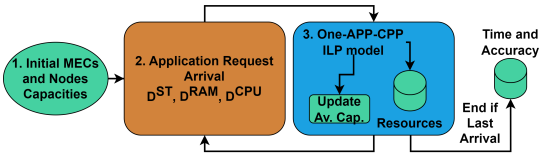


Fig. 2. Iterative Optimization Model for one Episode.

considering local node availability and application requirements as inputs for learning. This approach enables weight consideration across different domains. The application placement problem is described as a Markov Decision Process (MDP). The decision process is performed in all MEC systems independently of the remaining ones, employing local data. DRL is used to find the optimal policy for the MDP.

The state space S of the DRL model defined in Eq. (3) contains the availability of resources of the MEC nodes in a MEC system. The user sets up the number of nodes per MEC system before the model training. Therefore, the size of the state space array is the availability of storage, CPU, and RAM resources of the MEC system.

$$S = (C^{ST}, C^{RAM}, C^{CPU}) \tag{3}$$

where C^{ST}, C^{RAM} and C^{CPU} define the remaining storage, RAM, and CPU of all nodes.

The action space A is defined as the set of nodes in the MEC system, i.e., $A = \mathcal{N}$. The action associated with each request, a, represents the node where the request is allocated. The reward function in Eq. (4) is divided into three options. In the first option, a positive reward is provided if the node can cope with all the requirements and is not on standby. A higher normalized value of x corresponds to a lower resource capacity at the node. Then, the increasing exponential function boosts nodes with less availability ensuring the minimization of active nodes. The second option is a negative reward for placement decisions in a full node that cannot meet the requirements, to discourage such allocations. The corresponding x normalization guarantees a negative value if at least one of the resources cannot be satisfied, the associated exponential function acts as a negative amplified penalty. Similarly, in the third option, we wish to deter requests from being allocated to an idle node while non-idle nodes with available capacity exist. The reward will be 0 in that case.

Let $max(C_n^i)$ and C_n^i be the maximum initial capacity and the current capacity of the node n according to the resource $i \in b$, being $b = \{ST, RAM, CPU\}$, respectively. Given a request r demanding D_r^i resources $\forall i \in b$, we define the reward associated with an action $a = n$, w^n as:

$$w^n = \begin{cases} 10 \cdot 2^x - 1 & \text{if all } C_n^i > D_r^i \text{ ; } x = \sum\limits_{\forall i \in b} \frac{max(C_n^i) - C_n^i}{max(C_n^i)} \\ 10 \cdot (2^x - 1) & \text{if any } D_r^i > C_n^i \text{; } x = \sum\limits_{\forall i \in b} \left(\frac{C_n^i - D_r^i}{max(C_n^i)} - 1 \right) \\ 0 & \text{if } C_n^i = max(C_n^i)\ \forall i \in b. \end{cases} \tag{4}$$

Distribution enables parallelization of DRL models, making it suitable for scenarios with independent data management across domains. The architecture described in Section III benefits from distributed RL/DRL algorithms, allowing efficient learning and multitasking. Our DDRL algorithm adopts the ApeX approach [11], with multiple actors collecting diverse data sets from distinct MEC system environments. ApeX includes the concept of distributed prioritized experience replay owed to, intuitively, some experiences can contribute more to the agent's learning than others' experiences.

V. Performance Evaluation

A. Simulation Setup

This work uses the Simu5G simulator [2] for performance evaluation, an open-source framework for 5G networks that supports MEC applications onboarding. We integrate Simu5G with an API REST on the host machine to enable real-time information exchange between the simulation environment and Python libraries, facilitating data access for predictions and result analysis. The MEC Orchestrator relies on an API REST server and a Python framework to decide on application instantiation on MEC nodes with the trained ML model. Although Simu5G enables the design of MEC scenarios comprising a MEC Orchestrator and several MEC nodes, it does not support communication between MEC Orchestrators. In this work, we have extended Simu5G to include the interface described in Section III, which facilitates communication between multiple MEC Orchestrators and enables the exchange of parameters of the DDRL workers during training.

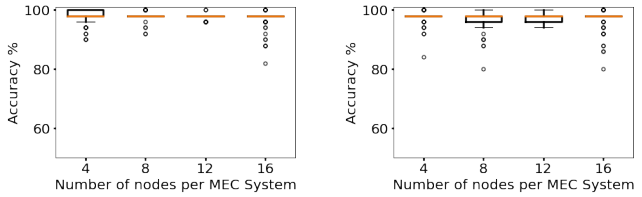
B. Methodology

The DDRL algorithm is tested in diverse simulation scenarios, evaluating its performance as the number of nodes and MEC systems increases. The scenarios consist of interconnected MEC systems managed by MEC Orchestrators. Orchestrators share model parameters during training to optimize worker models. Workers use shared parameters for re-training and performance improvement. While nodes within each MEC system have the same computational capabilities, they differ across MEC systems.

The evaluation involves episodes with new application arrivals. Each UE instantiates an application with random storage (512MB to 32GB), RAM (512MB to 4GB), and CPU (1 vCPU to 4 vCPU) capacity requirements. The application generation follows a Poisson distribution with an arrival rate of 15 apps/s, taking as reference a vehicular safety use case in [12]. The simulation ends when all applications are instantiated, or MEC node capacity is reached, whichever happens first. The evaluation is performed by varying the number of nodes in 2 MEC systems (4, 8, 12, and 16) with 50 applications to assess performance as nodes increase and by expanding to 4 and 6 MEC systems, each with 4 nodes and using 50 applications as input.

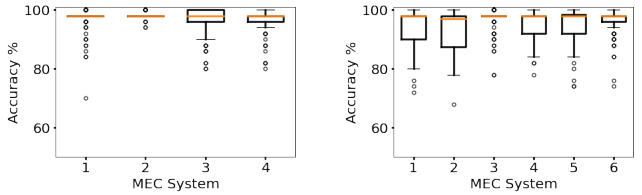
C. Performance Results

During training and inference, each model worker is placed in a MEC system environment with initial capacities specified by different vendors. In the testing phase, all worker models



(a) Accuracy in MEC System 1. (b) Accuracy in MEC System 2.

Fig. 3. Accuracy for different numbers of nodes per MEC system with 2 MEC systems and 50 applications.



(a) Accuracy for 4 MEC systems. (b) Accuracy for 6 MEC systems.

Fig. 4. Accuracy for different number of MEC systems with 4 Nodes per MEC system and 50 applications.

within each MEC system are exposed to the same environment for inference, evaluating their performance with unseen initial capacities. Each MEC system's DRL model independently determines the node for application allocation without knowledge exchange between MEC orchestrators.

Accuracy is assessed to measure the probability of the model predicting the action associated with the highest reward value among available nodes to meet the objective of minimizing the number of active nodes. Figure 3 depicts the accuracy results for the scenario with 2 MEC systems and an increasing number of nodes (ranging from 4 to 16). Note that the total number of applications arriving in the system is fixed at 50. To ensure statistical validity, each specific configuration is tested 100 times, including a 95% confidence interval. As the number of nodes increases, the average accuracy remains consistent. However, there is a slightly higher likelihood of obtaining accuracy values below the average. This is because the system may opt to allocate an application to the next node when the current one reaches its capacity. In such cases, if a new application is placed on the next node instead of the previous node with available space, it does not contribute positively to accuracy. However, this event does not significantly affect system performance, as accuracy consistently remains above 97% for all cases. The accuracy results when increasing the number of MEC systems for 4 and 6 are presented in Fig. 4. As the number of MEC systems increases, the algorithm requires more processing, and synchronization failures may occur, leading to slightly lower accuracy values. However, the decrease in the median accuracy is negligible compared to the results shown for the same number of nodes in Fig. 3.

Regarding ILP, the algorithm accuracy is always 100% for scenarios considering 1 to 16 MEC systems and 2, 4, 8, 12, or 16 nodes for each one with different application request arrivals. Combinatorial optimization problems solved by ILP are generally known to be NP-hard problems, making them challenging for online applications due to high computational time. Using the same DDRL configuration parameters, the average computing time to place one application is tens of ms; meanwhile, the DDRL algorithm achieves an average time of 0.2 ms, outperforming the ILP technique.

VI. Conclusions and Future Work

We propose an approach for MEC application placement in distributed environments managed by multiple orchestrators. Our method aims to minimize active nodes while ensuring SLA compliance and system performance. The approach aligns with ETSI MEC reference architecture and seamlessly incorporates a DDRL model with ApeX system to address the problem. Simu5G is used for the evaluation, extended to support standard control interface across orchestrators. Results show the model's adaptability to varying initial resource availability, achieving accurate application placement. Future research will explore the impact of volatile infrastructures.

Acknowledgements

This work has been performed in the framework of the EU's H2020 project AI@EDGE (101015922). The authors would also like to acknowledge the CERCA Programme/Generalitat de Catalunya, the EU "NextGenerationEU/PRTR", MCIN, and AEI (Spain) under project IJC2020-043058-I.

References

[1] F. A. Salaht, F. Desprez, and A. Lebre, "An Overview of Service Placement Problem in Fog and Edge Computing," *ACM Computing Surveys*, vol. 53, no. 3, pp. 1–35, Jun. 2020.

[2] G. Nardini, D. Sabella, G. Stea, P. Thakkar, and A. Virdis, "Simu5G–An OMNeT++ Library for End-to-End Performance Evaluation of 5G Networks," *IEEE Access*, vol. 8, pp. 181 176–181 191, Sep. 2020.

[3] T. Bahreini and D. Grosu, "Efficient Algorithms for Multi-Component Application Placement in Mobile Edge Computing," *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2550 – 2563, Oct. 2022.

[4] N.-E.-H. Yellas, B. Addis, R. Riggio, and S. Secci, "Function placement and acceleration for in-network federated learning services," in *Proc. of CNSM*, Thessaloniki, Greece, Nov. 2022, pp. 212–218.

[5] R. Solozabal, J. Ceberio, A. Sanchoyerto, L. Zabala, B. Blanco, and F. Liberal, "Virtual network function placement optimization with deep reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 292–303, Feb. 2020.

[6] R. Eyckerman, P. Reiter, S. Latre, J. Marquez-Barja, and P. Hellinckx, "Application Placement in Fog Environments using Multi-Objective Reinforcement Learning with Maximum Reward Formulation," in *Proc. of IEEE NOMS*, Budapest, Hungary, Apr. 2022, pp. 1–6.

[7] M. Goudarzi, M. S. Palaniswami, and R. Buyya, "A Distributed Deep Reinforcement Learning Technique for Application Placement in Edge and Fog Computing Environments," in *arXiv:2110.12415*, Oct. 2021.

[8] ETSI, "Multi-access Edge Computing (MEC); Framework and Reference Architecture," European Telecommunications Standards Institute, Group Specification (GS) MEC 003, Mar. 2022, version 3.1.1.

[9] B. Brik, P. A. Frangoudis, and A. Ksentini, "Service-Oriented MEC Applications Placement in a Federated Edge Cloud Architecture," in *Proc. of IEEE ICC*, Dublin, Ireland, Jun. 2020, pp. 1–6.

[10] ETSI, "Multi-access Edge Computing (MEC); Federation enablement APIs," European Telecommunications Standards Institute, Group Specification (GS) MEC 040, Feb. 2023, version 3.1.1.

[11] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, and D. Silver, "Distributed Prioritized Experience Replay," in *arXiv:1803.00933*, Mar. 2018.

[12] 5GPPP, "5G PPP use cases and performance evaluation models," https://5gppp.eu/wp-content/uploads/2014/02/5GPPPusecasesandperformanceevaluationmodeling_v1.0.pdf, Accessed on 09.05.2023.