**IAES International Journal of Artificial Intelligence (IJ-AI)**
Vol. 13, No. 4, December 2024, pp. 4186∼4194
ISSN: 2252-8938, DOI: 10.11591/ijai.v13.i4.pp4186-4194

# A new approach based on genetic algorithm for computation offloading optimization in multi-access edge computing networks

**Marouane Myyara, Oussama Lagnfdi, Anouar Darif, Abderrazak Farchane**
Laboratory of Innovation in Mathematics, Applications, and Information Technology, Department of Computer Science, Faculty of Polydisciplinary, Sultan Moulay Slimane University, Beni Mellal, Morocco

**Article Info**

***Article history:***

Received Dec 29, 2023
Revised Feb 11, 2024
Accepted Mar 21, 2024

***Keywords:***

Cloud computing
Computation offloading
Genetic algorithms
Multi-access edge computing
Resource optimization
Service time minimization

**ABSTRACT**

The proliferation of smart devices and the increasing demand for resource-intensive applications present significant challenges in terms of computational efficiency, leading to surge in data traffic. While cloud computing offers partial solutions, its centralized architecture raises concerns about latency. Multi-access edge computing (MEC) emerges as promising alternative by deploying servers at the network edge to bring computations closer to user devices. However, optimizing computation offloading in the dynamic MEC environment remains a complex challenge. This paper introduces novel genetic algorithm-based approach for efficient computation offloading in MEC, considering processing and transmission delays, user preferences, and system constraints. The proposed approach integrates computation offloading and resource allocation algorithm based on evolutionary principles, combined with a greedy strategy to maximize overall system performance. By utilizing genetic algorithms, the proposed method enables dynamic adaptation to changing conditions, eliminating the need for intricate mathematical models and providing an appealing solution to the complexities inherent in MEC. The urgency of this research arises from the critical need to enhance mobile application performance. Simulation results demonstrate the robustness and efficacy of our approach in achieving near-optimal solutions while efficiently balancing computation offloading, minimizing latency, and maximizing resource utilization. Our approach offers flexibility and adaptability, contributing to advancement of MEC networks and addressing the requirements of latency-sensitive applications.

***Corresponding Author:***

Marouane Myyara
Laboratory of Innovation in Mathematics, Applications, and Information Technology
Department of Computer Science, Faculty of Polydisciplinary, Sultan Moulay Slimane University
Beni Mellal 23000, Morocco
Email: marouane.myyara@usms.ac.ma

## 1. INTRODUCTION

In recent years, the rapid proliferation of smart devices has led to an increased demand for applications such as surveillance, augmented reality, voice recognition, automated transportation, healthcare, and autonomous driving. These applications generate a large amount of data traffic, especially from tasks that require low latency and involve complex computations [1]. However, smart devices often have limited processing power and storage capacity, which makes it challenging to perform computation-intensive tasks locally [2]. To address these challenges, cloud computing has emerged as a solution by utilizing the computing and storage resources of cloud servers over a wide area network for task processing [3]. While cloud computing can effectively handle resource-intensive applications and overcome certain limitations of user devices [4], it faces difficulties with latency-sensitive applications due to congestion and packet losses caused by the increasing data generated by user devices [5].

To overcome these challenges, multi-access edge computing (MEC) proposes the deployment of edge servers at the network's periphery to improve support for low-latency applications and services while ensuring quality of service (QoS) for end-users [6]. The MEC framework consists of three levels [7]: the user device level, the edge level, and the cloud level. At the user device level, devices communicate with the local edge server through WLAN. The edge level consists of multiple interconnected MEC servers via a metropolitan area network (MAN), while the cloud level involves global cloud servers using a wide area network. Computation offloading allows devices to offload tasks to the edge servers, contributing to overall performance enhancement. Various studies prioritize computation offloading in MEC networks, focusing on the full offloading of tasks to an edge server for each user device request [8]. However, selecting the optimal offloading server is a complex task that requires adaptation to fluctuating capacities and task diversity [9]. Traditional offline optimization is unsuitable due to uncertainties in offloading requests and changing network and computing conditions. The use of MEC for task offloading represents a promising strategy for reducing network latency [10]. In particular, MEC addresses the computational offloading needs of user devices by processing tasks near the edge, rather than relying solely on a central cloud [11]. However, the task offloading problem is recognized as an NP-hard problem [12], and finding a solution poses significant challenges.

Existing research has explored computation offloading in edge computing and offered various approaches to minimize latency in MEC systems. For example, Silva *et al.* [13] introduced a hybrid cloud-based offloading scheme. In contrast, Maleki and Mashayekhy [14] considered the runtime and application movement impact for offloading in mobile edge computing. Hossain *et al.* [15] proposed a task-offloading model that balances energy consumption and user satisfaction. Heuristic techniques, such as genetic algorithms, have also been utilized for quick offloading decisions, despite potentially suboptimal outcomes. Genetic algorithms have shown effectiveness in heterogeneous systems, as demonstrated by Neema *et al.* [16], who proposed genetic techniques for dynamic scheduling and load balancing in cloud environments, and Li *et al.* [17], who utilized genetic algorithms to address task offloading proportion, channel bandwidth, and computing resources of mobile edge servers. While these studies provide valuable insights, a comprehensive approach is needed that considers system constraints, user preferences, and dynamic MEC conditions.

In this research, we propose a genetic algorithm-based approach for offloading in MEC, leveraging a dynamic genetic algorithm and a greedy strategy for performance maximization. The adaptability of genetic algorithms makes them ideal for the dynamic environment of MEC, offering near-optimal solutions without complex models [18]. Our research methodology introduces the computation offloading challenges in MEC networks, identifies gaps in task-offloading approaches through a literature review, and acknowledges the success of genetic algorithms in MEC. We detail the MEC system modeling, integrating task offloading computations and user preferences. We incorporate task and computation models representing the MEC system to ensure precision. The problem formulation centers on computation offloading optimization, to minimize the total task service time. We lay the groundwork for our genetic algorithm design, which is explained in the subsequent section. Our offloading strategy based on genetic algorithms balances exploration and exploitation for efficient convergence without sacrificing diversity. An experimental methodology outlines metric selection and test scenarios, configuring simulation parameters. We assess the performance of our computation offloading approach using metrics such as total service time, task failure rate, and computation resource utilization. Results are discussed and compared with existing approaches for validation. In conclusion, we summarize contributions, exploring future research directions, including genetic algorithm enhancements in real environments and integration into operational MEC systems.

The remainder of this paper is organized as follows, section 2 provides details of our system model and problem formulation, section 3 explains our offloading strategy using genetic algorithms, section 4 presents the performance evaluation of our approach, and discuss the results, and finally, section 5 concludes the paper and explore future research directions.

## 2. SYSTEM MODEL AND PROBLEM FORMULATION

The system model in this paper is based on a mobile/MEC/cloud architecture. When a smart mobile device (SMD) intends to execute a resource-intensive task, it can offload it to an MEC or cloud server. The MEC system comprises three levels, as shown in Figure 1. Mobile users as clients utilizing the system's resources, MEC nodes/servers near users, and the centralized cloud server with more substantial resources. MEC servers connect to the orchestrator, forming a hierarchical network. When an SMD offloads a task, the MEC server covering its location attempts to execute it. If resources are insufficient, the orchestrator delegates the remaining tasks to other servers. Each computation task is defined by length, deadline, and resource consumption. Each MEC node has virtual machines (VMs) with maximum computing capacity, including CPU, memory, and storage, allocated for task execution. Two execution modes exist for a computation task: local and remote. In the local mode, the mobile device performs the task computation, generating the result. This method is simple and efficient, requiring no data transmission to a remote server. In the remote mode, the SMD sends an offloading request to the orchestrator via the MEC node. The MEC server allocates necessary computing resources based on requirements, executes the task, and sends the result back to the SMD.

As illustrated in the MEC system framework in Figure 1, let $E_S = \{S_1, S_2, \ldots, S_p, S_{p+1}\}$ be the set of MEC and cloud servers, where $p$ is the total number of servers in the MEC architecture. Let $VM_{S_i} = \{vm_{i,1}, vm_{i,2}, \ldots, vm_{i,n}\}$ be the set of VMs, where $n$ is the total number of VMs in each MEC server $S_i$. To distinguish between VMs in MEC and the cloud server, let $VM_C = \{vm_{C,1}, vm_{C,2}, \ldots, vm_{C,m}\}$ be the set of VMs in the cloud server $S_{p+1}$, where $m$ is the total number of VMs.

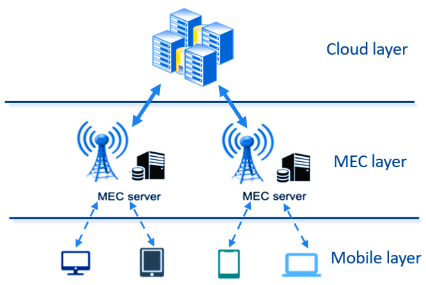

Figure 1. MEC system model with mobile/MEC/cloud architecture

### 2.1. Task model

Unlike other works [19], [20], each SMD in this study isn't confined to a single task but rather has a set of tasks. Equipped with a set of applications, each SMD's tasks are denoted by $A_t^{SMD_i} = \{\tau_{t,1}^i, \tau_{t,2}^i, \ldots, \tau_{t,j}^i\}$. Here, $SMD_i$ is the smart mobile device $i$, $t$ is the application type, and $j$ is the total tasks in application $t$. For simplicity, each task is an atomic unit of input data, not fragmented. Each task has three parameters: task length $L_{\tau_{t,j}^i}$ represents the task data amount transferred to the MEC server or the cloud, indicated by instructions. Delay sensitivity $D_{\tau_{t,j}^i}^S$ specifies task sensitivity to delay, close to 1 indicates high sensitivity. Maximum delay demand $D_{\tau_{t,j}^i}^{max}$ indicates the maximum acceptable delay for chosen server processing.

### 2.2. Computation model

The task computation model incorporates considerations for both local and remote execution times. Specifically, the local execution time of a task represents the time required for the execution of task $\tau_{t,j}^i$ on the $SMD_i$. This parameter captures the duration needed for the local processing of the task on the specific $SMD_i$:

$$T_{S,\tau_{t,j}^i}^{SMD_i} = \frac{L_{\tau_{t,j}^i}}{F_{SMD_i}} \tag{1}$$

Here, $L_{\tau_{t,j}^i}$ represents the amount of data for task $\tau_{t,j}^i$, and $F_{SMD_i}$ is the processing speed of device $SMD_i$ in million instructions per second (MIPS). The remote execution time of a task, denoted $T_{S,\tau_{t,j}^i}^{Off,S_p}$, is the time required to execute the task on a MEC or cloud server. It is given by (2):

$$T_{S,\tau_{t,j}^i}^{Off,S_p} = T_{\tau_{t,j}^i}^{Trans,S_p} + T_{\tau_{t,j}^i}^{Process,S_p} = T_{\tau_{t,j}^i}^{Up,S_p} + T_{\tau_{t,j}^i}^{Dw,S_p} + T_{\tau_{t,j}^i}^{Process,S_p} \tag{2}$$

Here, $T^{Trans,S_p}_{\tau^i_{t,j}}$ denotes the transmission time of task $\tau^i_{t,j}$ to server $S_p$, representing the time required to transfer task data from the mobile device to the MEC server or the cloud. The upload and download data transfer times are denoted by $T^{Up,S_p}_{\tau^i_{t,j}}$ and $T^{Dw,S_p}_{\tau^i_{t,j}}$ for the respective transfers of task input data from the mobile device to the target server ($S_p$) and output data from the computing server to the source mobile device. These delays are calculated using a network model integrated into the EdgeCloudSim simulator [21]. The simulator dynamically evaluates data transmission delays, considering the locations and number of mobile devices, traffic, and network types. The processing time, denoted $T^{Process,S_p}_{\tau^i_{t,j}}$, is calculated using (3):

$$T^{Process,S_p}_{\tau^i_{t,j}} = \frac{L_{\tau^i_{t,j}}}{F_{S_p,n}} \tag{3}$$

Here, the processing time represents the time required to execute the task on the MEC or cloud server, and $F_{S_p,n}$ represents the processing speed expressed in MIPS. This parameter characterizes the computational capability of the $n$-th VM on the server $S_p$, providing a measure of its efficiency in processing instructions.

### 2.3. Problem formulation

In this section, the computation offloading problem in the MEC environment is formulated. The objective is to assign a set of tasks to MEC or cloud servers to meet the QoS requirements of mobile applications and the constraints of computing resources while minimizing the total processing and transmission time of these tasks. The computation offloading decision for a task $\tau^i_{t,j}$ is represented by the binary variable $\xi_i$. If $\xi_i = 0$, the task is executed locally, and if $\xi_i = 1$, it is offloaded to a MEC or cloud server. The problem can be mathematically formulated as follows:

$$minimize \quad T = \sum_{i=1}^{N_T} (1-\xi_i) \times T^{SMD_i}_{S,\tau^i_{t,j}} + \xi_i \times T^{Off,S_p}_{S,\tau^i_{t,j}} \qquad (P1)$$

$$subjectto:$$

$$C_1: \quad \xi_i \in \{0,1\}; \quad \forall \tau^i_{t,j} \in A^{SMD_i}_t, \quad \forall i \in \{1,\dots,N_T\}$$

$$C_2: \quad \max\left(T^{SMD_i}_{S,\tau^i_{t,j}}, T^{Off,S_p}_{S,\tau^i_{t,j}}\right) \le D^S_{\tau^i_{t,j}}; \quad \forall i \in \{1,\dots,N_T\}, \quad \forall S_p \in E_S$$

$$C_3: \quad \frac{C_{Req,\tau^i_{t,j}}}{C^{vm_{i,n}}_{Target}} \le 1 \;\; \& \;\; \frac{C^{vm_{i,n}}_{Target}}{C^{vm_{i,n}}_{Select}} > 1 \;\; \& \;\; C^{vm_{i,n}}_{Target} \le Th_{S_p}; \quad \forall vm_{i,n} \in VM_{S_i}, \quad \forall S_p \in E_S$$

The problem formulation (P1) is designed to minimize the total service time of tasks, considering both local and remote execution times weighted by the parameter $\xi_i$. The binary variable $\xi_i$ decides whether task $\tau^i_{t,j}$ is offloaded ($\xi_i = 1$) or processed locally ($\xi_i = 0$). Constraints ($C_1$-$C_3$) define the conditions and limits of the problem. $C_1$ ensures that the binary variable $\xi_i$ is within the interval (0,1), indicating whether the task is offloaded or processed locally. $C_2$ guarantees that the processing and transmission time of each task does not exceed its latency sensitivity $D^S_{\tau^i_{t,j}}$. Constraint $C_3$ encompasses conditions on computing resources, ensuring that the demand and capacity of virtual resources meet predefined criteria. It concurrently regulates the load of VMson MEC servers, preventing it from exceeding the limit defined by $Th_{S_p}$.

## 3. COMPUTATION OFFLOADING OPTIMISATION STRATEGY BASED ON GENETIC ALGORITHMS

The genetic algorithm is an evolutionary approach that simulates the process of natural evolution [22]. Applied to solve complex problems [23], genetic algorithms find extensive use in addressing NP-complete problems, including load balancing in cloud computing [24]. Their implementation involves a population of solutions, where each solution is represented by chromosomes. These algorithms utilize operators inspired by biological processes, facilitating convergence towards optimal solutions while avoiding local optima. Genetic algorithms efficiently find solutions close to optimal within a reasonable timeframe, making them a promising approach for addressing the computation offloading problem in dynamic MEC environments. Given the challenge of solving (P1) using traditional methods, genetic algorithms are employed, and the steps are outlined as follows:

– Step 1: Encoding and generation. This involves creating an initial population of potential solutions represented by chromosomes. Each chromosome corresponds to a computation offloading scheme for an application task, with genes representing possible execution locations.
– Step 2: Evaluation. To obtain an optimal assignment of tasks to computing servers, an adaptation function measures the optimality of offloading decisions. After evaluating the chromosomes, they are ranked based on their fitness, where the fitness of a chromosome $x$ is calculated as in (4):

$$f(x) = \frac{1}{(1-\xi_i) \times T^{SMD_i}_{S,\tau^i_{t,j}} + \xi_i \times T^{Off,S_p}_{S,\tau^i_{t,j}}} \tag{4}$$

This fitness function represents the inverse of the total service time of tasks, considering both local and remote execution times weighted by the binary variable $\xi_i$. The chromosomes are then ranked based on their fitness values, with higher values indicating better solutions.
– Step 3: Selection operation. The tournament selection method is used to choose the fittest chromosomes, which are retained for reproduction, while less performing ones are eliminated.
– Step 4: Crossover operation. The crossover operator combines the features of two parents to create new offspring, gradually improving the quality of solutions.
– Step 5: Mutation operation. Mutation modifies one or more genes of a chromosome, preventing premature convergence to a local optimum. Single-point mutation, altering a random gene, is used.
– Step 6: Replacement. The new offspring solutions replace the worst solutions in the current population, maintaining the population's quality and avoiding premature convergence.
– Step 7: Termination and iteration. Steps 2 through 6 are iteratively executed for a designated number of generations. The algorithm terminates upon reaching the maximum specified iterations or when the overall fitness function converges to its minimum value.

The resolution method based on these steps is described in Algorithm 1. The proposed genetic algorithm optimizes computation offloading in MEC through an evolutionary approach. In the initialization step, a population is created with chromosomes representing distinct computation offloading schemes onto MEC and cloud servers. The algorithm iterates until convergence or a preset limit. Fitness values, based on the inverse of total service time, guide chromosome selection, crossover, and mutation. A combined population of parents and offspring is formed, and the top-performing chromosomes advance to the next iteration. The algorithm concludes by selecting the optimal solution from the final population, offering a versatile framework for efficient computation offloading optimization in dynamic MEC environments.

**Algorithm 1** Genetic algorithm for computation offloading optimization

```
Require: $\tau^i_{t,j}$ : Incoming Task, $E_S$ : Set of MEC and Cloud Servers
 1: Initialization
 2: $P \leftarrow$ Generate an initial population of feasible solutions$(\tau^i_{t,j}, E_S)$
 3: repeat
 4:     Evaluation
 5:     for all $x \in P$ do
 6:         $f(x) \leftarrow$ Calculate fitness function as in (4)
 7:     end for
 8:     $P_{\text{sorted}} \leftarrow$ Sort chromosomes based on their fitness
 9:     $P_{\text{selected}} \leftarrow$ Select top-performing chromosomes from $P_{\text{sorted}}$
10:     while the number of generations is not exhausted do
11:         Crossover
12:         $P_{\text{offspring}} \leftarrow$ Generate offspring by applying crossover to $P_{\text{selected}}$
13:         Mutation
14:         $P_{\text{offspring}} \leftarrow$ Apply mutation on $P_{\text{offspring}}$
15:         Evaluation
16:         for all $x \in P_{\text{offspring}}$ do
17:             $f(x) \leftarrow$ Calculate fitness function$(x)$
18:         end for
19:         $P_{\text{combined}} \leftarrow P_{\text{selected}} \cup P_{\text{offspring}}$
20:         $P_{\text{selected}} \leftarrow$ Select top-performing chromosomes from $P_{\text{combined}}$
21:     end while
22: until Convergence
Ensure: $x$ : The best chromosome from the final population$(P_{\text{selected}})$
```

## 4. PERFORMANCE EVALUATION AND SIMULATION RESULTS

This section is dedicated to the thorough evaluation of the genetic algorithm for computation offloading optimization (GA-COO) algorithm's performance through simulations. We comprehensively consider a range of parameters, and we benchmark our results against various offloading schemes and algorithms within the MEC context. The objective is to provide a detailed analysis of how the proposed GA-COO algorithm compares and performs under diverse scenarios, shedding light on its effectiveness in addressing computation offloading challenges in MEC networks.

### 4.1. Experimental evaluation

The GA-COO algorithm's experimental assessment aims to minimize task response time and failure rates, considering mobile application constraints and MEC/cloud server utilization. Metrics analyzed include task response time (processing and transmission time), computational task failure rates, and resource utilization. Results are compared with four algorithms: i) DCOA-ST [25]: a greedy approach minimizing task service time while adhering to sensitivity constraints, ii) Random [26]: random task assignment to available VM, serving as a reference, iii) Utilization-based [27]: allocation based on real-time VM utilization with predefined thresholds, iv) Network-based [27]: allocation considering network characteristics, including bandwidth, and v) Hybrid [27]: an approach combining real-time VM utilization and network characteristics.

### 4.2. Simulation environment

To assess GA-COO, EdgeCloudSim [21] was chosen for its realistic simulation of decentralized environments, precise computing architecture modeling, and accurate workload generation. Task parameters, generated randomly using EdgeCloudSim's workload generator, defined simulated applications (e.g., augmented reality, health, infotainment, and heavy computation) with specific characteristics. These applications varied in upload/download sizes, task lengths, and VM utilization. Delay sensitivity ranged from 0.9% to 0.1%, and maximum delay requirements varied from 0.5 seconds to 2 seconds. The MEC architecture included 14 MEC servers, 1 cloud server, and a variable number of SMDs. Each MEC server had 8 VMs per host, the cloud server had 4 VMs per host, and SMDs had 1 core per VM. VM CPU speed was specified in GIPS, with values of 10 for MEC servers, 100 for the cloud server, and 4 for SMDs. Experiments were conducted using EdgeCloudSim in JAVA on a computer with 4 Intel Core i7-9600U processors at 2.59 GHz and 8 GB of RAM.

### 4.3. Simulation results

With the specified parameters, we evaluate our genetic algorithm's performance. Figure 2 displays computation task service times relative to the number of SMDs, showcasing GA-COO's efficiency. Results reveal a linear increase in service time with growing SMD numbers, highlighting challenges associated with limited shared computing resources. GA-COO maintains competitive service times, ranging from 1.1 seconds for 100 SMDs to 1.95 seconds for 2300 SMDs. In comparison, the DCOA-ST algorithm exhibits notable performance, ranging from 0.8 seconds to 2.5 seconds for the same SMD values.

Simulation results of failure rates in Figure 3 provide insights into the robustness of different approaches concerning SMD numbers. GA-COO maintains a low and stable failure rate up to 2100 SMDs, then experiences a notable increase to 12% for 2300 SMDs. DCOA-ST maintains a low failure rate, slightly increased to 20% for 2300 SMDs. In contrast, Random exhibits a rapid growth in the failure rate, reaching 61% for 2300 SMDs. Hybrid shows a progressive increase to 28%; Network-based maintains stability up to 2100 SMDs, then increases to 18% for 2300 SMDs. Utilization-based demonstrates a significant increase to 46% for 2300 SMDs.

Analyzing Figure 4 reveals trends in average VM utilization for MEC servers. GA-COO maintains stability at 25% for 1900 SMDs, persisting at 24% for 2100 and 2300 SMDs. DCOA-ST's performance remains consistent, slightly increasing to 19% for 2300 SMDs. In contrast, Random experiences rapid growth, reaching 61% for 2300 SMDs. Hybrid progresses to 69.5%, indicating sensitivity to high loads. Network-based stays stable up to 1900 SMDs, then increases to 44% for 2300 SMDs. Utilization-based increases to 59% for 1900 SMDs, slightly decreasing to 49% for 2300 SMDs.

In terms of cloud VM utilization, depicted in Figure 5, GA-COO maintains gradual usage, ranging from 0.1% to 3.9% for 100 to 2300 SMDs. DCOA-ST increases from 0.1% to 8.1% for 100 to 2300 SMDs. Random shows a slight increase, from 0.3% to 1% for 100 to 2300 SMDs. Hybrid maintains minimal usage, from 0.01% to 1.25% for 100 to 2300 SMDs. Network-based displays lower usage, from 0.3% to 2.1% for 100 to 2300 SMDs.

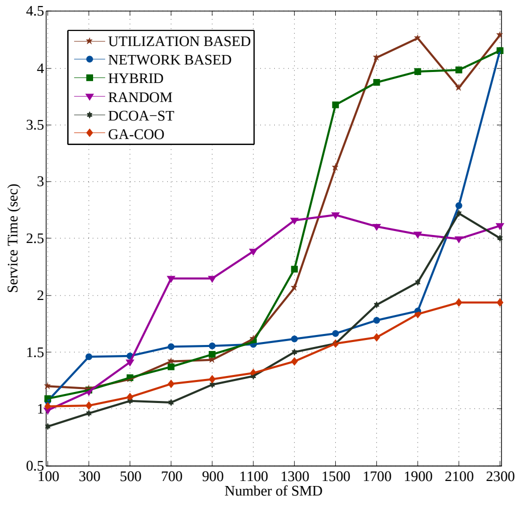

Figure 2. Comparison of average service times for different offloading approaches with various SMDs

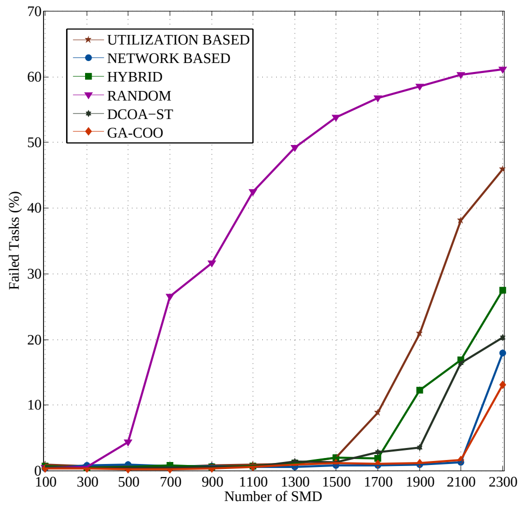

Figure 3. Comparison of failure rates among different offloading approaches for various SMDs

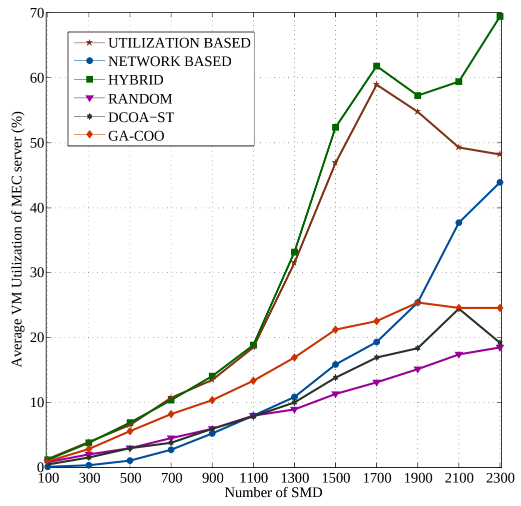

Figure 4. Average VM MEC servers utilization as a function of the number of SMDs

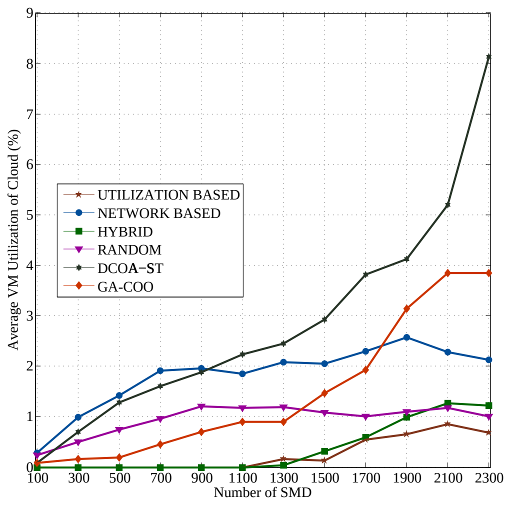

Figure 5. Average VM cloud utilization as a function of the number of SMDs

The simulation results confirm the stability and effectiveness of the GA-COO approach for computation offloading in MEC networks. GA-COO maintains consistent performance under increasing workloads, with a linear increase in task service times and a low, stable task failure rate even under high loads. The utilization of VMs in MEC servers remains stable, indicating efficient resource management in the presence of significant workloads. In response to the identified challenges, GA-COO demonstrates competitive performance with short task service times, low failure rates, and stable resource utilization, indicating efficient resource management, particularly under heavy workloads.

## 5. CONCLUSION

The proposed GA-COO employs a genetic algorithm to optimize computation offloading in MEC networks. Comparative evaluations, involving various reference offloading schemes, unequivocally establish GA-COO's effectiveness in minimizing task service times and failure rates, all while considering the constraints of mobile applications and computational resource utilization. GA-COO consistently outperforms traditional methods and certain reference algorithms, especially in scenarios with a substantial number of SMD. This robust performance underscores the significance of tailored offloading strategies that account for application specifics and system constraints. This work not only introduces a novel computation offloading approach in MEC but also aligns with broader goals of optimizing edge computing systems for diverse applications in an increasingly interconnected and data-intensive landscape. Future research prospects involve not only enhancing GA-COO's performance in real-world environments but also considering specific factors such as quality of experience and mobility. Furthermore, crucial investigations into GA-COO's impact on other aspects of MEC networks, including energy consumption and security, are imperative for a comprehensive understanding of its implications.

## REFERENCES

[1] N. E. Nwogbaga, R. Latip, L. S. Affendey, and A. R. A. Rahiman, "Attribute reduction based scheduling algorithm with enhanced hybrid genetic algorithm and particle swarm optimization for optimal device selection," in *Journal of Cloud Computing*, vol. 11, no. 15, 2022, doi: 10.1186/s13677-022-00288-4.

[2] Q. Zhang, L. Gui, S. Zhu, and X. Lang, "Task offloading and resource scheduling in hybrid edge-cloud networks," *IEEE Access*, vol. 9, pp. 85350-85366, 2021, doi: 10.1109/ACCESS.2021.3088124.

[3] J. Fang, J. Shi, S. Lu, M. Zhang, and Z. Ye, "An efficient computation offloading strategy with mobile edge computing for IoT," *Micromachines*, vol. 12, no. 2, 2021, doi: 10.3390/mi12020204.

[4] J. Zhang, W. Zhu, X. Wu, and T. Ma, "Traffic state detection based on multidimensional data fusion system of internet of things," *Wireless Communications and Mobile Computing*, vol. 2021, pp. 1-12, 2021, doi: 10.1155/2021/1374186.

[5] L. Long, Z. Liu, Y. Zhou, L. Liu, J. Shi, and Q. Sun, "Delay optimized computation offloading and resource allocation for mobile edge computing," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, Sep. 2019, pp. 1-5, doi: 10.1109/VTC-Fall.2019.8891166.

[6] F. Hoseiny, S. Azizi, M. Shojafar, and R. Tafazolli, "Joint QoS-aware and cost-efficient task scheduling for fog-cloud resources in a volunteer computing system," *ACM Transactions on Internet Technology (TOIT)*, vol. 21, no. 4, pp. 1–21, 2021, ACM New York, doi:10.1145/3418501.

[7] C. Sonmez, A. Ozgovde, and C. Ersoy, "Fuzzy workload orchestration for edge computing," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 769-782, 2019, doi: 10.1109/TNSM.2019.2901346.

[8] U. Saleem, Y. Liu, S. Jangsher, Y. Li, and T. Jiang, "Mobility-aware joint task scheduling and resource allocation for co-operative mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 360-374, 2021, doi: 10.1109/TWC.2020.3024538.

[9] C. Li, M. Song, M. Zhang, and Y. Luo, "Effective replica management for improving reliability and availability in edge-cloud computing environment," *Journal of Parallel and Distributed Computing*, vol. 143, pp. 107-128, 2020, doi: 10.1016/j.jpdc.2020.04.012.

[10] D. Lim and I. A. Joe, "DRL-based task offloading scheme for server decision-making in multi-access edge computing," *Electronics*, vol. 12, 2023, doi: 10.3390/electronics12183882.

[11] A. Samanta and Z. Chang, "Adaptive service offloading for revenue maximization in mobile edge computing with delay-constraint," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3864-3872, 2019, IEEE, doi: 10.1109/JIOT.2019.2892398.

[12] B. Wang, Y. Song, J. Cao, X. Cui, and L. Zhang, "Improving task scheduling with parallelism awareness in heterogeneous computational environments," *Future Generation Computer Systems*, vol. 94, pp. 419-429, 2019, doi: 10.1016/j.future.2018.11.012.

[13] J. Silva, E. R. Marques, L. M. Lopes, and F. Silva, "Jay: adaptive computation offloading for hybrid cloud environments," in *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*, pp. 54-61, 2020, doi: 10.1109/FMEC49853.2020.9144950.

[14] E. F. Maleki and L. Mashayekhy, "Mobility-aware computation offloading in edge computing using prediction," in *2020 IEEE 4th International Conference on Fog and Edge Computing (ICFEC)*, pp. 69-74, 2020, doi: 10.1109/ICFEC50348.2020.00015.

[15] M. D. Hossain, L. N. T. Huynh, T. Sultana, T. D. T. Nguyen, J. H. Park, C. S. Hong, and E. N. Huh, "Collaborative task offloading for overloaded mobile edge computing in small-cell networks," in *2020 International Conference on Information Networking (ICOIN)*, pp. 717-722, 2020, doi: 10.1109/ICOIN48656.2020.9016452.

[16] G. Neema, A. B. Kadan, and V. P. Vijayan, "Multi-objective load balancing in cloud infrastructure through fuzzy based decision making and genetic algorithm based optimization," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 12, no. 2, pp. 678-685, 2023, doi: 10.11591/ijai.v12.i2.pp678-685.

[17] Z. Li and Q. Zhu, "Genetic algorithm-based optimization of offloading and resource allocation in mobile-edge computing," *Information*, vol. 11, no. 2, 2020, doi: 10.3390/info11020083.

[18] K. Wang, X. Wang, and X. Liu, "Sustainable internet of vehicles system: A task offloading strategy based on improved genetic algorithm," *Sustainability*, vol. 15, no. 9, 2023, doi: 10.3390/su15097506.

[19] H. Gu, M. Zhang, W. Li, and Y. Pan, "Task offloading and resource allocation based on DL-GA in mobile edge computing," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 31, no. 3, pp. 498-515, 2023, doi: 10.55730/1300-0632.3998

[20] M. Zhao and K. Zhou, "Selective offloading by exploiting ARIMA-BP for energy optimization in mobile edge computing networks," *Algorithms*, vol. 12, no. 2, 2019, doi: 10.3390/a12020048.

[21] 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, 2018, doi: 10.1002/ett.3493.

[22] A. Ulah, N. M. Nawi, J. Uddin, and S. Baseer, "Artificial bee colony algorithm used for load balancing in cloud computing: Review," *IAES International Journal of Artificial Intelligence*, vol. 8, no. 2, pp. 156-167, 2019, doi: 10.11591/ijai.v8.i2.pp156-167.

[23] B. Keshanchi, A. Souri, and N. J. Navimipour, "An improved genetic algorithm for task scheduling in the cloud environments using

the priority queues: formal verification, simulation, and statistical testing," *Journal of Systems and Software*, vol. 124, pp. 1-21, 2017, doi: 10.1016/j.jss.2016.07.006.

[24] Z. Zhou, F. Li, H. Zhu, H. Xie, J. H. Abawajy, and M. U. Chowdhury, "An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments," *Neural Computing and Applications*, vol. 32, pp. 1531-1541, 2020, doi: 10.1007/s00521-019-04119-7.

[25] M. Myyara, A. Darif, and A. Farchane, "Task deadline-based computation offloading algorithm for service time minimization in mobile edge computing," in *International Conference on Artificial Intelligence and Green Computing*, pp. 149-162, 2023, doi: 10.1007/978-3-031-46584-0_12.

[26] S. Fu, C. Ding, and P. Jiang, "Computational offloading of service workflow in mobile edge computing," *Information*, vol. 13, no. 7, 2022, doi: 10.3390/info13070348.

[27] V. Nguyen, T. T. Khanh, T. D. Nguyen, C. S. Hong, and E. N. Huh, "Flexible computation offloading in a fuzzy-based mobile edge orchestrator for IoT applications," *Journal of Cloud Computing*, vol. 9, no. 1, pp. 1-18, 2020, doi: 10.1186/s13677-020-00211-9.

## BIOGRAPHIES OF AUTHORS

**Marouane Myyara** received his B.Sc. in Electronic and Telecommunication Engineering in 2019 and M.Sc. in Telecommunication Systems and Computer Networks in 2021 from Sultan Moulay Slimane University, Beni Mellal, Morocco. He is currently a Ph.D. candidate at the Laboratory of Innovation in Mathematics, Applications, and Information Technology (LIMATI), Faculty of Polydisciplinary, Sultan Moulay Slimane University, Morocco. His current research focuses on improving the performance of multi-access edge computing networks, cloud computing, computation offloading, and internet of things. He can be contacted at email: marouane.myyara@usms.ac.ma.

**Oussama Lagnfdi** received his B.Sc. in Physical Matter Science in 2020 and M.Sc. in Telecommunications Systems and Computer Networks in 2022 from Sultan Moulay Slimane University, Beni Mellal, Morocco. Currently, he is a Ph.D. candidate at the Laboratoire d'Innovation en Mathématiques et Applications et Technologies de l'Information (LIMATI), Faculty of Polydisciplinary, Sultan Moulay Slimane University, Morocco. His ongoing research is focused on enhancing the performance of internet of things and mobile edge computing, artificial intelligence, deep learning, and fuzzy logic. He can be contacted at email: lagnfdi.o@gmail.com.

**Anouar Darif** received the bachelor in Informatique Electrothecnique, Electronique and Automatique (IEEA) from Dhar El Mahraz, Faculty of Sciences at Mohamed Ben Abdellah University Fez, Morocco in 2005. He received the Diplôme d'Etudes Supérieurs Approfondies in Computer Sciences and Telecommunications from Faculty of Sciences Rabat in 2007. He received the Ph.D. degree in Computer Sciences and Telecommunications from Faculty of Sciences of Rabat in 2015. He is currently a Research and a Teaching Associate in Faculty of Multidisciplinary at University of Sultan Moulay Slimane Beni Mellal, Morocco. His research interests include wireless sensor network (WSN), mobile edge computing (MEC), internet of things (IoT), cloud computing, and neural networks. He can be contacted at email: anouar.darif@gmail.com.

**Abderrazak Farchane** received his B.Sc. in Computer Science and Engineering in June 2001 and M.Sc. in Computer Science and Telecommunication from the University of Mohammed V Agdal, Rabat, Morocco, in 2003. He obtained his Ph.D. in Computer Science and Engineering at ENSIAS, Rabat, Morocco. He is currently an Associate Professor of Computer Science in the Faculty of Polydisciplinary at Sultan Moulay Slimane University, Morocco. His areas of interest are information coding theory, cryptography, and security. He can be contacted at email: a.farchane@gmail.com.