Library Progress International *Print version ISSN 0970 1052*
Vol.44 No. 3, Jul-Dec 2024: P.14181-14191 *Online version ISSN 2320 317X*

Original Article *Available online at www.bpasjournals.com*

Augmentation Of Efficient Task Offloading In Mobile Edge Environments For Mobile Applications

Ms. Archana M. S [1] and Dr. N. Anandakrishnan [2]

[1] Research Scholar, P.G and Research Department of Computer Science, Providence College for Women, Coonoor, The Nilgiri's District.
[2] Assistant Professor, Department of Computer Science, Providence College for Women, Coonoor, The Nilgiri's District.

.**How to cite this article:** Archana M. S, N. Anandakrishnan (2024) Augmentation Of Efficient Task Offloading In Mobile Edge Environments For Mobile Applications. *Library Progress International,* 44(3), 14181-14191.

ABSTRACT

Mobile edge computing (MEC) emerges as a cutting-edge technique that effectively alleviates the computational burden on mobile devices through task offloading. In the realm of Mobile Edge Computing (MEC) augmented by 5G technology, the process of delegating computing tasks from edge devices to edge servers within the edge network can significantly diminish latency. Overcoming the challenge of designing a well-balanced task offloading strategy in a resource-limited multi-user with MEC environment to address users' requirements remains a noteworthy concern. This paper introduces Greedy method based Genetic Algorithm (GA) for task offloading proportion, channel bandwidth, and Mobile Edge Servers' (MES) computing resources. The method is designed to handle scenarios where certain computing tasks can be partially offloaded to the MES. By considering the limitations imposed by wireless transmission resources and Edge servers' processing capacity, GA is employed to optimize the task completion time for users. The offloading strategy, which utilizes the combination of Greedy with GA, is evaluated and compared against both the genetic algorithm (GA) and the Greedy method through a series of simulation experiments demonstrates its effectiveness in reducing Energy Consumption, Delay time, and ensuring fairness in Resource cost for users' task completion times.

Keywords: Mobile Edge Computing (MEC), task offloading, Genetic Algorithm (GA), Greedy method

1. Introduction

Within the radio access network of cellular networks, Mobile Edge Computing (MEC) is a ground-breaking idea that brings processing and data storage closer to mobile devices and users. Due to network transmission delays, traditional cloud computing involves delivering data to centralized remote computers for processing. The "edge" of the network, or where data is generated and consumed, is where MEC deploys computational resources in contrast. MEC accomplishes this by greatly reducing latency and enabling real-time processing for a variety of applications, including augmented reality, video streaming, and IoT devices. Local edge servers at cellular base stations or access points are used by MEC as a backup plan to distant data centers. With this architecture, users will have better overall user experience, quicker response times, and better resource utilization.

Low latency, effective data offloading, and scalability are some advantages of MEC. It supports new technologies like 5G and the Internet of Things, improves application performance, and makes the most use of network resources. Mobile Edge Computing enables networks to provide services that are faster, more responsive, and more dependable by bringing processing closer to the edge of the network.

The strategic process of task offloading in Mobile Edge Computing (MEC) involves moving computational and data processing from mobile devices to edge servers that are situated at the network's edge. Applications with high computing resource requirements or low latency can benefit from this technique's improved performance and efficiency.

The following steps are part of the task offloading process: (i)Task analysis and division, (ii)decision-making, (iii)data preparation, offloading, and result retrieval. Task offloading, in general, is a vital method in Mobile Edge Computing that enhances application performance, reduces latency, and conserves resources, matching with the requirements of computation-intensive and latency-sensitive applications on contemporary

wireless networks.

The rest of this paper is organized as follows. Section 2 introduces the related work of task computing offloading. Section 3 deals with the system model of MEC architecture. Section 4 deals with the proposed model and section 5 concludes the proposed work and provides guidance for the future work.

2. Literature Review

Computing offloading is the process of moving computing duties now carried out by MDs to expanded cloud or grid platforms, supporting terminal devices in fulfilling user task demands, and redistributing computing results to designated devices [1]. Users of the MEC system bypass the conventional method of sending data to a distant central cloud for processing by using the MEC server as a middleman to call computer power closer to the local region [2].

The use of machine-learning techniques to solve the compute offloading problem has advanced significantly with the development of machine learning [3]. A Distributed Deep Learning-based Offloading (DDLO) approach was put forth by Liang et al. [4] and used many simultaneous deep neural networks to provide offloading decisions. An experience replay method was used to continuously update the network parameters. The authors only took the static network scenario into account, however the system quickly generates decisions that are close to ideal for offloading. For the complicated compute offloading problem in collaborative computing with heterogeneous edge computing servers, Li et al. [5] suggested a deep reinforcement learning approach.

The offloading decision was optimized by the algorithm based on the task's characteristics and the network's real-time status in order to reduce task delay; however, the authors only took task delay into account and ignored energy expenditure. Using the DDQN (Double Deep Q Network) technique to dynamically produce offloading decisions, Zhou et al. [6] applied deep reinforcement learning to examine the joint optimization problem of computing offloading and resource allocation in dynamic multiuser MEC systems.

To increase RIS-UAV network capacity, reference [7] presented a DDQN-based trajectory and phase shift optimization algorithm. With regard to building mobile node incentive mechanisms and content-caching strategies for D2D offloading, a novel incentive-driven and deep Q network-based method (IDQNM) was put out in reference [8].

The source [9] suggested a reward system based on reverse auction and delay limits. Two optimization techniques—the greedy winner selection method (GWSM) and the dynamic planning winner selection method (DPWSM)—were suggested with the goal of maximizing mobile network operators' income. The DQN algorithm was enhanced, computing task completion times were slashed, and terminal energy consumption was decreased thanks to a collaborative optimization method T. Yang and J. Yang [10] developed. By modeling resources as "fluids" and allocating them via an auction procedure, Zhu et al. [11] suggested a dynamic resource allocation technique based on K-means. The edge server's throughput was increased, and the transmission delay was decreased. Task dependency has been ignored by the aforementioned methods, despite the fact that it is a real phenomenon in real-world applications. The following are some sample research findings on dependent task offloading mechanisms. A computational-offloading method based on genetic algorithms was proposed by Dong et al. [12].

This technique used delay and energy consumption as assessment criteria, encoded the position and sequence of work offloading, and constantly optimized the task offloading choice using variation and crossover operations. However, it did not take edge server resource allocation into account. [13] investigated the fine-grained offloading problem with several users and servers. The authors suggested an enhanced Non Dominated Sorting Genetic Algorithm (NSGA-II) with the aim of decreasing the average delay since they saw the fine-grained offloading of Internet of Things (IoT) devices as a multi constrained goal optimization problem. A delay-acceptance-based offloading technique for multiuser jobs was put forth by Mao et al. [14].

The plan began by using a non-dominated genetic algorithm to find the best solution for each user in a single-user scenario. Next, a probabilistic selection mechanism and non-dominated judging scheme improved convergence speed. Finally, an adjustment plan based on the notion of time delay acceptance in stable matching was proposed. With dependent task scenarios, the multiuser offloading issue was resolved. To lower the overall energy consumption of IoT devices, Liu et al. [15] introduced an energy-efficient collaborative task-offloading algorithm based on semi-definite relaxation and stochastic mapping. This approach generated offloading decisions for dependent tasks in static network environments.

3. System Model

Our investigation revolves around the scenario encompassing multiple users and several MEC servers. Across the entire network, User Equipments (UEs) establish connections via base stations and wireless channels. To furnish computational services to UEs, multiple MEC servers are strategically positioned alongside these base stations. In contrast, cloud servers are situated at the pinnacle of the core network, distant from the UEs. Unlike the uncertainty in delay and extended data transmission distances associated with mobile cloud computing employed by UEs for task offloading to cloud servers, MEC offers a more expeditious and efficient means of delivering computational services to UEs, concurrently alleviating the strain on core networks [16]. Our study exclusively concentrates on the challenge of computation offloading between the user layer and the edge layer. In this context, users have the option to delegate tasks for processing to either local or MEC servers.

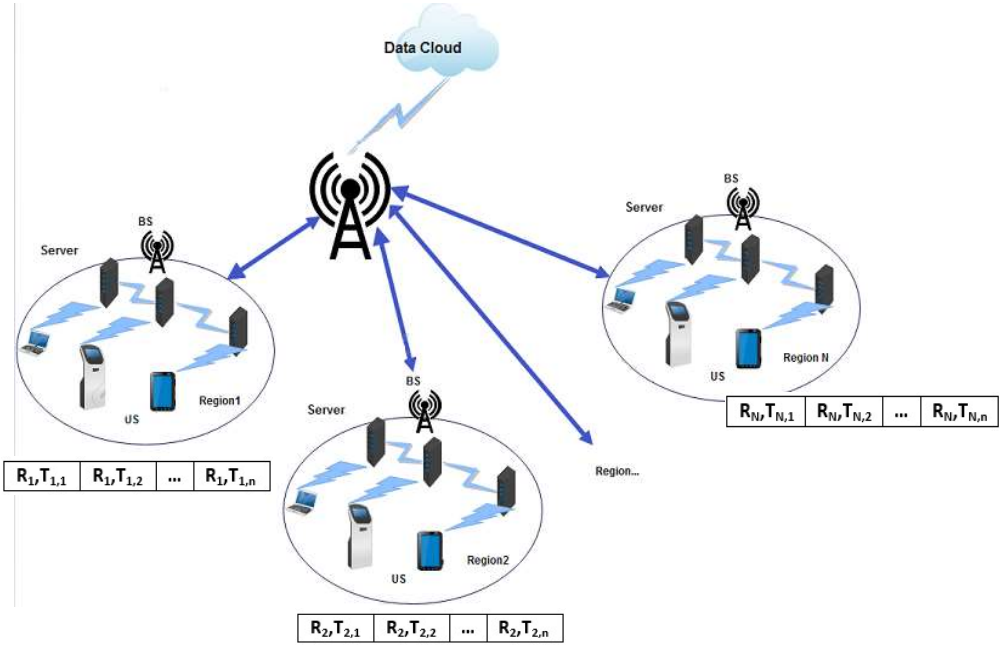

Figure1: MEC System Architecture

Figure 1 illustrates the comprehensive System Architecture. Within this architecture, the uppermost tier comprises a cloud server engaged in communication with the base station via a dedicated link. Multiple base stations are distributed throughout the entire network. Adjacent to each base station, there exists an array of compact MEC servers, facilitating inter-MEC server communication. At the end-user smart device level, tasks have the option to undergo direct execution on the local server or to be transmitted through the data transfer unit to a proximate MEC server for remote computation.

The system of entire network is partitioned into several discrete and independent regions. Our focus centers on the computation offloading conditions within a specific region, and we operate under the subsequent assumptions. Within each region, there are P User Systems (US) and E MEC servers. Each US_i (where $i \in \{1, 2 \ldots P\}$) undertakes the offloading of computational tasks to MEC_j (where $j \in \{1, 2 \ldots E\}$) utilizing wireless communication links.

Every user is tasked with executing computational assignments. Let $M_{i,j} \in \{0, 1\}$ denote the number of integers used to determine the offloading strategy for task j on mobile device i. Specifically, when ($M_{i,j} = 0$), task j on mobile device i is executed locally; conversely, when ($M_{i,j} = 1$), task j on mobile device i is transmitted to the MEC server via a wireless channel.

3.1. Problem Formulation:

The objective of this study is to make informed decisions regarding task offloading within a subset of a workflow application, focusing on minimizing energy consumption, delay, and resource costs. Numerous current

methodologies overlook the consideration of energy consumption and resource costs for tasks or subsets in both cloud and edge environments.

The selected set M is represented as $\{m_{1,1}, m_{1,2}, m_{1,3} \ldots, m_{A,B}\}$, and it serves as the offloading profile for managing computational tasks across all mobile device users. As per the principles of the channel propagation model, each user receives a dedicated bandwidth portion from the overall system allocation.

The rate at which the amount of tasks transmit from User systems (US_i) to MEC server (MEC_j) can be defined as

$$d^i(tp^i) = Blog\left(1 + \frac{tp^i h^{i,j}}{p_\theta^{i,j} \omega_0}\right) \quad \text{----- (1)}$$

Where, B is the channel bandwidth; tp^i represents transmission power of US_i ; $h^{i,j}$ is the channel gain from $US_i$to MEC_j ; $p^{i,j}$ represents the path between $US_i$and MEC_j ; θ is the path loss exponent and ω_0 represents the channel noise function

In the scenario where all mobile devices opt to transfer their computational tasks to the wireless channel simultaneously during a computation offloading interval, there exists a constraint on the achievable data rate denoted as R.

$$\sum_{i=1}^{A} \sum_{j=1}^{B} M_{i,j} d^i \leq R \quad \text{----- (2)}$$

Each user i utilizing a mobile possesses N computational tasks. These tasks have the option to be processed either locally on the device or remotely on the MEC server using wireless communication. Additionally, we denote $US_{i,j}$ as the individual on mobile device i who is seeking the execution of computation j.

Each computational task, denoted as j, is characterized by the tuple $(Ct^{i,j}, TW^{i,j})$, where $Ct^{i,j}$ represents the required number of CPU cycles to accomplish task j, and $TW^{i,j}$signifies the overall volume of off loadable data. The computation of all offloaded tasks can be performed locally with respect to the time taken for the tasks are expressed by

$$Total\ time_{local}^{i,j} = \frac{Ct^{i,j}}{CVol_{local}^{i}} \quad \text{----- (3)}$$

Where, $CVol_{local}^{i}$ represents the Computing volume of the mobile 'i' performing locally.

The energy consumed to perform the computational task locally by a mobile 'i' can be represented by

$$EC_{local}^{i,j} = v^i * Ct^{i,j} \quad \text{----- (4)}$$

Where, v^i is the amount of energy consumed in each CPU cycles by the mobile device 'i'.

According to the MEC server, a mobile device user i selects to send computational task j to aMobile Edge Server (MEC) via the wireless channel. The duration of task execution during offloading is influenced by the transmission time, encompassing the time taken by mobile device users to offload the computation task, along with the time for the task to be processed on the MEC servers—referred to as task execution time.

The complete duration of offloading, calculated as the combined value of transmission time, execution time, and the energy expended during transmission which is calculated using the given formula;

$$T\_time_{trans,exec}^{i,j} = \frac{TW^{i,j}}{d^i} + \frac{Ct^{i,j}}{MEC_{pow}^{i}} \quad \text{----- (5)}$$

$$EC_{trans}^{i,j} = tp^i \frac{TW^{i,j}}{d^i} \quad \text{----- (6)}$$

Where, MEC_{pow}^{i} is the computational capacity, quantified as the count of CPU cycles per second, allocated to user i on the designated edge server.

The local execution delay of each offloading task can be calculated by

$$Delay_{loc}^{i,j} = \frac{TW^{i,j}}{CP_{UE}^{i}} \quad \text{----- (7)}$$

Where, CP_{UE}^{i} is computing ability of UE for each task.

Each mobile device user is responsible for both execution time and energy consumption. Moreover, the management of distributing available resources on the edge server is also conducted. This issue of offloading is characterized as an optimization problem in the Greedy algorithm:

$$min\left(\sum_{i=1}^{A} \sum_{j=1}^{B} \left(EC_{trans}^{i,j} + EC_{local}^{i,j}\right)\right), \quad \text{------ Optimization Problem (OP 1)}$$

the problem and the constrained values for reducing the energy are;

$$\sum_{i=1}^{A} \sum_{j=1}^{B} M_{i,j} d^i \leq R\,, \forall i, j \quad \text{----- (Constrained 1)}$$

$\sum_{i=1}^{A}\sum_{j=1}^{B}\left(EC_{trans}^{i,j}+EC_{local}^{i,j}\right) < E_{max}$ ----- (Constrained 2)

$\sum_{i=1}^{A}\sum_{j=1}^{B} CP_{UE}^{i} \leq CP_{UE}^{max}$ -----(Constrained 3)

$\sum_{i=1}^{A}\sum_{j=1}^{B} Delay_{loc}^{i,j} \leq Delay_{max}^{i,j}$ ----- (Constrained 4)

The objective of the optimization task involves minimizing energy consumption among mobile device users through the utilization of task offloading techniques. This is subject to the constraint 1, which represents the data rate capacity. The constraint 2, ensures that energy consumption from both the Edge server and the mobile device remains below the specified maximum, denoted asE_{max}. The Constraint 3 shows that the MEC computing resources allocated to each user should not be larger than the computing resources of MEC itself and the Constrained 4 represents that the local delay always less than $Delay_{max}^{i,j}$.

4. Proposed Method

4.1. Greedy Based Genetic Algorithm (GBGA) method:

The optimization problem stated in equation (OP1) is characterized as non-convex, posing a challenge for conventional greedy approaches to solve. A heuristic technique known as the Genetic Algorithm (GA) [17] is introduced. Unlike directly manipulating individual object parameters, GA assesses numerous solutions within the exploration space. Benefitted by its robust global search capacity, GA employs genetic operations including gene selection, crossover, and mutation [18] to efficiently and precisely address intricate problems while avoiding getting trapped in local optima.

We employ the GBGA method to determine computational offloading choices due to its precise resolution of mathematical challenges and quicker processing duration compared to alternative heuristic methods. Our novel approach enhances resource usage by appropriately sequencing and allocating workflow tasks, consequently potentially reducing overall resource expenses.

4.1.1. Population Initialization and Assessment

Within the framework of the genetic algorithm, the starting point involves populating a collection of individual solutions, each comprised of genetic components. This method, a type of metaheuristic algorithm, is adept at generating high-quality solutions for various challenges, including optimization and search tasks. In this specific section, the population's initialization process is elaborated upon, alongside the subsequent determination of fitness values, which transpires as follows:

a. Initialization

During the Initialization stage, our GBGA algorithm is employed to initiate a range of potential solutions concerning the offloading of a subset of the workflow application to the edge environment. This subset encapsulates a group of dependent tasks acquired during the workflow application partitioning process, serving to uphold task interdependencies. The possible set of solutions used in the search space is called population and its composition is expounded upon as follows:

$Popu = \{Ch^1, Ch^2, \dots, Ch^m, \dots Ch^N\}$ ------ (8)

Where, N is the size of the initial population, where m=1, 2… N.

The individual chromosome in the population can be calculated as

$Ch^m = y_m^1, y_m^2, \dots y_j^k, \dots, y_m^n$ ------ (9)

Where j= {1, 2… n}, here n is the total number of sub values of mobile devices from the equation (9).

If $y_j^k = 0$ implies that the sub values of $US_i$is offloadedto the MEC devices and $y_j^k = 1$ implies that the sub values of $US_i$is offloaded to the server

b. Fitness Function:

Within the context of the GBGA approach, the computation of the fitness function plays a crucial role in identifying the most optimal solution from the available alternatives. In the case of each chromosome, our approach systematically assesses the fitness function, aiming to concurrently minimize both delay and energy consumption.

c. ***Fitness Score Function****:*

Our GBGA methodology assesses potential solutions to determine the optimal one by employing the fitness function. The genetic algorithm computes the fitness value through the utilization of the objective function. Within the framework of the GBGA strategy, the objective function FS (US_i) is constructed with a focus on two

components: minimizing total time and reducing energy consumption.

$$FS(US_{i,j}) = \left\{m \times \left[\sum_{i,j=1}^{t} Total\ time_{local}^{i,j} + \sum_{i,j=1}^{t} T\_time_{trans,exec}^{i,j}\right] + \left[n \times \left[\sum_{i,j=1}^{t} EC_{local}^{i,j} + \sum_{i,j=1}^{t} EC_{trans}^{i,j}\right]\right]\right\} \quad \text{------- (10)}$$

From the equation (10), the values of variables m and n undergo modifications in accordance with the attributes of the provided application. Within the genetic algorithm framework, a pair of chromosomes is chosen from the population using the roulette wheel method to execute genetic operations like mutation or crossover. The introduced genetic operation creates a new population, which in turn contributes to the formation of another population set. This iterative process persists until either the optimal solution is achieved or the predefined maximum iteration count is reached.

4.1.2. Genetic Operations

The genetic procedure is tasked with extracting the optimal solution among the potential options. Within the GBGA methodology, an efficient computation of the fitness score occurs during the selection, crossover, and mutation stages to enhance decision-making regarding offloading. The calculation of the fitness score to secure the best possible solution through various processes, encompassing selection, crossover, and mutation, is outlined below:

a. Selection: The selection of a chromosome is a crucial factor in generating subsequent populations to achieve the optimal solution. Within the GBGA methodology, the roulette wheel selection technique is employed to choose the parent chromosome. The calculation for chromosome selection in the genetic algorithm during recombination is outlined as follows:

$$Sel^{i} = \frac{FS(US_{i,j})}{\sum_{p=1}^{S} FS(US_{i,j})} \quad \text{----- (11)}$$

b. Crossover: Crossover is employed to generate a high-quality offspring. To create the next population generation with improved solution quality, the crossover phase of the genetic algorithm merges the two chosen individuals such as P^1 and P^2

$$FS(US_{i,j}^{P}) = \sum_{i,j=1}^{t}\{[m \times Totaltime^{i,j}] + [n \times EC^{i,j}]\} \quad \text{------ (12)}$$

Utilizing equation (12), the GBGA technique produces an offspring, Ch1, by evaluating the fitness value of individual genes in the parent chromosomes. Subsequently, genes with superior fitness values from the parents are replicated into the offspring.

c. Mutation: The arrangement of genes within the chromosomes is adjusted to achieve the optimal solution, aiming to minimize the waiting time of the subset. The process involves selecting gene g^s within the chromosome and sequentially examining for its immediate successor g^{suc} until the end of thequeue is reached. In the event that another gene g^r is positioned between g^s andg^{suc} .

If its precursor exists before g^s this mutation process alters the positions of both g^s and g^r .

This mutation process is iteratively applied until the position of gene g^s becomes equal to g^{r-1}.

Algorithm: GBGA Algorithm for effective task offloading:

Inputs: Population size N , No. of Iterations 'I', Mutation Probability m^p, Crossover Probability c^p.

Output: Finding the optimal decision from $Popu$for offloading the tasks

1 Randomly set the populations$Popu$.
2 Set the input values
3 For i=1 to N do
4 $Popu = Popu[i]$;
5 Assess the fitness value of an each individual in Ch^m using the equation (10)
6 Update the $Popu$ at each iteration
7 While stopping criterion is not met do / Until the Optimal Solution*/*
8 For i=1 to N do
9 Calculate the selection of new chromosomes using the equation (11)
10 Perform Crossover of a chromosomes using the equation (12)
11 Make to find the mutation based on the successor value of the subset g^s

12 $new[i] = Popu[i] + +$
13 *Evaluate the fitness function for* $new[i]$
14 *Select the best individuals to get the new generation of population through the mutation function*
15 *End for*
16 *End*
17 *Return optimal offloading policy*

Algorithm 1 elucidates the comprehensive procedure of the GBGA approach, which effectively governs the decision-making process for offloading communications between User devices and mobile edge servers. The initial phase of our algorithm involves populating a set of potential solutions within predetermined dimensions. Following this, parameters such as crossover probability, mutation probability, and the number of iterations are established to facilitate the identification of these potential solutions.

Next, an assessment is made by evaluating execution time, energy consumption, and time delay of the subset across diverse environments, with the aim of computing its fitness function through employment of the genetic algorithm. Within the population of possible solutions, a pair of parents is selected using the roulette wheel selection technique. By combining the genetic material of these selected parents, we generate offspring of superior quality, thereby enhancing the overall search proficiency.

The development of high-caliber offspring is achieved through the replication of genes from parents exhibiting greater fitness value, denoted as $FS(US_{i,j})$. Subsequently, the refinement of the generated offspring transpires by implementing mutation-dependent task operations inherent to the genetic algorithm. Through this mutation process, gene positioning within the chromosome is readjusted in accordance with associated decisions, ultimately minimizing waiting times.

This sequence of steps is reiterated until the stipulated condition is met, thus ensuring the iterative progression of the algorithm.

4.2. Simulation Results & Discussion

Within this segment, we perform thorough simulations to assess the effectiveness of the Greedy with Genetic algorithm across various scenarios. Our simulation environment is created using Matlab R2022a.

4.2.1. Simulation Parameter:

Simulation Parameter	Value
Channel bandwidth	150kHz
Number of tasks	500-5000
Data size of tasks	250~1000 Kb
Computation capacity of MEC server	3GHz
Computation capacity of US	0.5-1 GHz
Distance between US and MEC server	50~100 m
Distance of MEC servers	50~100 m
Number of MECs	2-7
Path loss	2
Transmission power of US	2W

4.2.2. Evaluation Metrics:

✓ Energy Consumption: Energy consumption pertains to the energy utilized while executing workflow tasks within the Mobile Edge or Cloud environment.

✓ Delay: The delay can be calculated using :

Total Delay = Propagation Delay + Transmission Delay (Mobile-to-Server) + Processing Delay + Queuing Delay + Network Delay + Execution Time + Transmission Delay (Server-to-Mobile)

✓ Resource Cost: Calculate the computational resources required to execute the offloaded task on the edge server. This can include CPU cycles, memory usage, and other processing-related costs such as Data transmission cost, Queuing cost and waiting cost.

4.2.3. Evaluation Results:

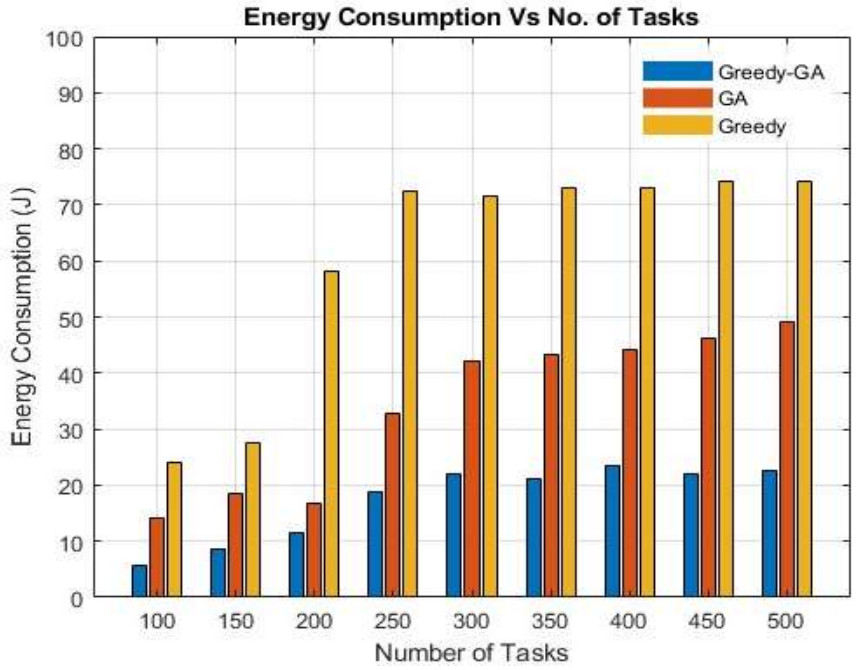

Figure 2: Energy Consumption

The utilization of the Greedy-GA method demonstrates a comparison between energy consumption and the number of tasks is shown in Figure 2. The task is derived from various applications and encompasses task ranges between 100 and 500. The analysis indicates that an increase in task rate leads to increased energy consumption. In contrast, the existing approach known as the greedy algorithm, and Genetic Algorithm maximizes energy consumption as the task quantity rises. Similarly, the Greedy-GA approach also exhibits an uptick in consumption with growing task numbers, yet this value remains lower than that of the Greedy, GA approach. This discrepancy arises from the fact that the Greedy and GA approach primarily concentrates on minimizing latency. On the contrary, our proposed methodology Greedy-GA addresses both on time and energy usage.

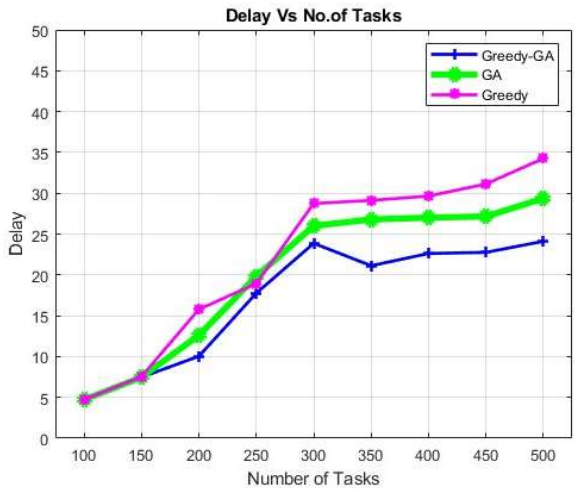

Figure 3: Delay

Figure is show as the delay with number of tasks. As the quantity of mobile devices rises, the average delay for each approach (excluding no offloading) experiences an upward trend, attributed to the potential load escalation at the edge nodes. The proposed algorithm, adept at managing uncertain edge load dynamics, manages to achieve a 7.5% reduction in average delay compared to both the Greedy method and GA when the number of mobile devices reaches 500.

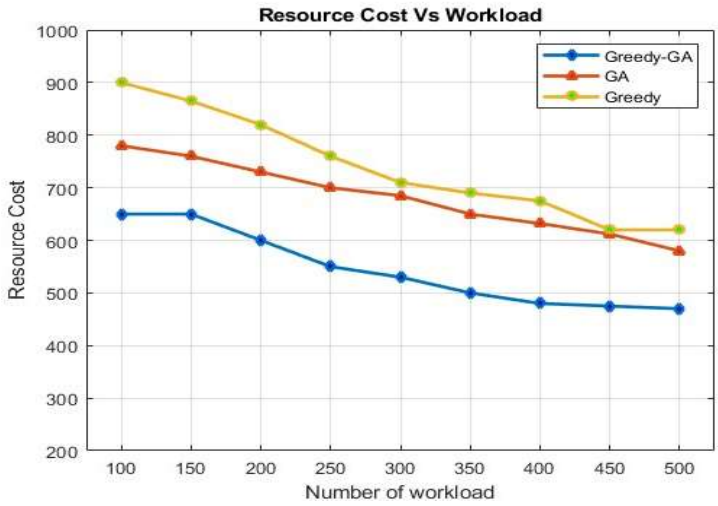

Figure 4: Resource Cost

Figure 4 depicting the curve of average resource costs across various workloads reveals an unfavorable correlation between number of workload and average resource cost. In contrast, our proposed algorithm maintains consistent stability in average resource costs, while the Greedy and GA methods exhibit notable fluctuations. As workload increases, there is a gradual decrease in resource costs, potentially leading to a point where user processing capacity cannot suffice for demands. Furthermore increased offloading quantities empower the MEC server can encourage the users with lower costs for increased task offloading, ultimately benefiting the system. At approximately 400 units of user workload, the curve for average resource cost begins to raise ground, driven by the necessity for users to align offloading with their processing capacities.

5. Conclusion:

The GBGA approach facilitates decision-making for offloading in either cloud or Edge environments. Initially, our focus is on the computation offloading of intricate, interdependent fine-grained tasks, which model user-generated mobile applications. Subsequently, our attention shifts to the optimization of execution delay and energy consumption for applications originating from User Equipment (UE). Employing a genetic algorithm, our GBGA approach strives to identify an optimal solution from a range of possibilities. However, it's worth noting that the scope of this article's scenario remains relatively straightforward.

Remarkably, our methodology excels at minimizing energy consumption, resource costs, and total delay compared to existing algorithms. Through simulation results, we demonstrate the efficacy of our proposed algorithm in achieving resource allocation and energy consumption reduction. This superiority is evident when compared to both Greedy and Genetic Algorithm (GA) approaches in the context of UE energy consumption reduction.

Furthermore, our ongoing research endeavors will encompass the incorporation of user mobility and dynamic computation offloading. These extensions will be pursued using optimization techniques alongside efficient offloading strategies.

References

[1] Fang, J.; Shi, J.; Lu, S.; Zhang, M.; Ye, Z. An Efficient Computation Offloading Strategy with Mobile Edge Computing for IoT. Micromachines 2021, 12, 204.

[2] Shan, X.; Zhi, H.; Li, P.; Han, Z. A Survey on Computation Offloading for Mobile Edge Computing Information. In Proceedings of the 2018 IEEE 4th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing, (HPSC) and IEEE International Conference on Intelligent Data and Security (IDS), Omaha, NE, USA, 3–5 May 2018; IEEE: Piscataway, NJ, USA, 2018.

[3] A. Shakarami, M. Ghobaei-Arani, and A. Shahidinejad, "A survey on the computation offloading approaches in mobile edge computing: a machine learning-based perspective," Computer Networks, vol. 182, no. 9, article 107496, 2020.

[4] L. Huang, X. Feng, A. Feng, Y. Huang, and L. P. Qian, "Distributed deep learning-based offloading for mobile edge computing networks," Mobile Networks and Applications, vol. 27, pp. 1123–1130, 2022.

[5] Y. Li, F. Qi, Z. Wang, X. Yu, and S. Shao, "Distributed edge computing offloading algorithm based on deep reinforcement learning," IEEE Access, vol. 8, pp. 85204–85215, 2020.

[6] H. Zhou, K. Jiang, X. Liu, X. Li, and V. C. M. Leung, "Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing," IEEE Internet of Things Journal, vol. 9, no. 2, pp. 1517–1530, 2022.

[7] H. Zhang, M. Huang, H. Zhou, X. Wang, N. Wang, and K. Long, "Capacity maximization in RIS-UAV networks: a DDQN-based trajectory and phase shift optimization approach," IEEE Transactions on Wireless Communications, vol. 99, pp. 1–1, 2022.

[8] H. Zhou, T. Wu, H. Zhang, and J. Wu, "Incentive-driven deep reinforcement learning for content caching and D2D offloading," IEEE Journal on Selected Areas in Communications, vol. 39, no. 8, pp. 2445–2460, 2021.

[9] H. Zhou, X. Chen, S. He, J. Chen, and J. Wu, "DRAIM: a novel delay-constraint and reverse auction-based incentive mechanism for WiFi offloading," IEEE Journal on Selected Areas in Communications, vol. 38, no. 4, pp. 711–722, 2020.

[10] T. Yang and J. Yang, "Deep reinforcement learning method of offloading decision and resource allocation in MEC," Computer Engineering, vol. 47, no. 8, pp. 37–44, 2021.

[11] X. Zhu, M. Wu, and G. Wang, "Dynamic resource allocation strategy based on K-means," Computer Engineering and Design, vol. 42, no. 4, pp. 901–907, 2021.

[12] D. O. Hao, Z. H. Haiping, L. I. Zhongjin, and H. Liu, "Computation offloading for service workflow in mobile edge computing," Computer Engineering and Applications, vol. 55, no. 2, pp. 36–43, 2019.

[13] C. U. Yu-ya, Z. H. De-gan, Z. H. Ting, Y. A. Peng, and Z. H. Hao-li, "A multi-user fine-grained task offloading scheduling approach of mobile edge computing," Acta Electronica Sinica, vol. 49, no. 11, pp. 2202–2207, 2021.

[14] Y. Mao, T. Zhou, and P. Liu, "Multi-user task offloading based on delayed acceptance," Computer Science, vol. 48, no. 1, pp. 49–57, 2021.

[15] F. Liu, Z. Huang, and L. Wang, "Energy-efficient collaborative task computation offloading in cloud-assisted edge computing for IoT sensors," Sensors, vol. 19, no. 5, pp. 1105–1105, 2019.

[16] Qiuping, L.; Junhui, Z.; Yi, G. Computation offloading and resource management scheme in mobile edge computing. Telecommun. Sci. 2019, 35, 36.

[17] Galletly, J. Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms; Oxford University Press: New York, NY, USA, 1999.

[18] Morris, G.M.; Goodsell, D.S.; Halliday, R.S.; Huey, R.; Hart, W.E.; Belew, R.K.; Olson, A.J. Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function. J. Comput. Chem. 2015, 19, 1639–1662.