


Task offloading in edge computing using integrated particle swarm optimization and genetic algorithm

Shabariram C. Palaniappan^{1*} , Priya P. Ponnuswamy¹

¹ PSG Institute of Technology and Applied Research, Coimbatore, Tamil Nadu, India

* Corresponding author's e-mail: cps@psgitech.ac.in

ABSTRACT

In the ever-evolving landscape of smart city applications and intelligent transport systems, vehicular edge computing emerged as a game-changing technology. Imagine a world where computational resources are no longer restricted to distant cloud servers but are brought nearer to the vehicles and users. Task offloading enables the computation in edge and cloud server. This proximity not only minimizes network latency but also enables a unfold of vehicles to process tasks at the edge, offering a swift and interactive response to the scenarios of applications with delay sensitivity. To deal with this constraint, an integrated methodology is utilized to enhance the offloading process. The proposed system integrates the particle swarm optimization (PSO) and genetic algorithm (GA). The integrated system optimizes task allocation by exploring the solution space effectively and ensuring efficient resource utilization while minimizing latency. In the evaluation, PSO+GA exhibits enhanced adaptability to varying task sizes, facilitating efficient offloading to the edge as needed. Energy efficiency varies between the algorithms, with PSO+GA generally showing minimal energy consumption. When compared to already existing algorithms such as Energy aware offloading, no offloading and random offloading, PSO+GA outperformed these algorithms in system performance and less energy consumption by a factor of 1.18.

Keywords: edge computing, task offloading, genetic algorithm, optimization, energy efficiency, particle swarm optimization.

INTRODUCTION

The rapid advancement of mobile devices, their apps, and the volume of data captured by them bring about considerable increases in usage of bandwidth and network core congestion. Edge Computing responds to these inadequacies by increase the cloud notion to the network's edge, where competent nodes can carry out compute-intensive operations. In recent years, edge computing has come up to help with mobile apps. For mobile applications with scarce resources, this paradigm makes use of vehicles as edge node devices to provide storage, computation, and bandwidth [1].

The continued development of Intelligent Vehicle Systems and the internet of vehicles as emerging technologies has altered vehicular edge computing (VEC). Computation offloading is a

crucial problem in the dynamic environment. Although a minority of offloading techniques are offered to improve the mobility, computational performance, priority, and offloading breakdown are rarely taken into account for optimization, making it problematic [2].

One possible solution for carrying out efficient delay-constrained applications on devices with limited computing resources is the computation offloading approach. Through computation offloading, Vehicular Edge computing integrates processing power into automobiles to provide computing services for other vehicles [3, 4]. Mobility has an impact on the communication environment, creating significant obstacles for computation offloading. The suggested system in this study looks into a task offloading scenario for edge computing, such as smart cars and roadside equipment that might

work together to pool resources. The goals of Computational Offloading are to speed up computing, conserve energy, bandwidth, and provide low latency. However, VEC offloading presents complicated resource management issues, making it mostly unavailable to the automotive industry [5]. Serverless computing has recently developed as a practical way to execute activities without the complexity of infrastructure administration [6]. The scenario makes the primary impact on task offloading in vehicular edge computing environment. The task offloading needs to consider the environment parameters. The major objectives of the proposed system are given below.

- Optimize task allocation in vehicular edge computing environments.
- Adapt to varying task sizes that allows an efficient offloading to the edge.
- Minimize the latency and improves the system performance by exploring the solution space and ensuring efficient resource utilization
- The evaluation results demonstrate that the proposed methodology outperforms existing algorithms such as Energy aware offloading, No offloading, and random offloading in terms of system performance and energy efficiency.

RELATED WORK

The field of task offloading in edge computing has witnessed significant research interest in recent years, with numerous approaches proposed to optimize resource allocation and improve system performance. A variety of techniques have been explored, including heuristic algorithms, optimization-based methods, and genetic algorithm-based approaches as follows.

Shaohua Cao et al. [7] presented a technique which combines fuzzy inference and reinforcement learning to optimize task offloading. By accurately identifying peak and low hours and effectively offloading computational requests, the method outperforms benchmarks by 24.8% in resource utilization. However, the system lacks detailed analysis of limitations and security concerns. Nevertheless, the reinforcement learning algorithm shows promise in enhancing resource utilization in vehicular edge computing networks. Zheng Zhang et al. [8] used a genetic algorithm to optimize the cost and the time of task offloading, taking advantage of the objective problem's convex feature for quick convergence. While their

strategy demonstrated superiority in task offloading delay and cost compared to alternatives. The dynamic changes in the edge computing environment were considered and relied on the assumption of a convex objective problem.

Homa Maleki et al. [9] address key parameters such as average offloading cost, energy consumption, and delay in transmission and processing phases. The system suggested an algorithm for task offloading in moving vehicles that combines Double Q-learning with deep reinforcement learning. The strategy aims to minimize average offloading cost by considering energy consumption and delays. User equipment is able to acquire offloading cost performance and make well-informed offloading decisions by means of deep reinforcement learning using double Q-learning. However, the result analysis lacks a detailed discussion of the limitations or potential challenges associated with this approach in vehicular environments and fails to provide a comprehensive comparison or analysis of existing strategies. Simulation results, however, show the low-cost performance of the suggested scheme in comparison to other offloading decision strategies in the literature, indicating its efficacy.

Leila Ismail et al. [10] investigated factors including crossover rate, population size, mutation rate, and termination condition. But as the volume of requests rises, there are issues because of the division of processing power, which results in violations of deadlines and processing time constraints. Moreover, the algorithm fails to consider the effects of different vehicle speeds and overlapped multi-request processing, which could influence optimization results. On the other hand, QoS-SLA based algorithm performs better than the other algorithms, with an average total execution time of 13.98 seconds and no requests that violate SLA limits.

Xingxia Dai et al. [11] considered a multi-armed bandit (MAB) architecture to handle the issues associated with inadequate offloading information. In comparison to algorithms lacking service vehicle (SeV) capacity and truthfulness awareness, the learning regret by 39% to 41%. The method enhances SeV's QoS and ensures safe V2V compute offloading. However, the algorithm does not explicitly address privacy leakage risks in V2V computation offloading, which could expose security and safety threats. Furthermore, the dynamic nature of vehicular roles across epochs and potential degradation of

QoS parameter are not fully accounted for in the algorithm, posing challenges to its applicability. Despite these limitations, the algorithm demonstrates improvements in both SeVs' QoS and safe V2V computation offloading.

Sumit Singh et al. [12] utilized genetic algorithms and binary particle swarm optimization (BPSO) to improve computation offloading in mobile edge computing (MEC) devices. The approach highlighted the superior performance of evolutionary algorithms in maximizing network operator profit through efficient resource allocation and computation offloading strategies. However, it lacks explicit discussion on the scalability and robustness of these algorithms in dynamic operational conditions. Additionally, practical challenges in real-world implementation are not thoroughly addressed. Nonetheless, simulation results demonstrate higher profitability and faster execution times, showcasing the effectiveness of evolutionary algorithms in optimizing C-RAN environments.

Yueyue Dai et al. [13] propose a novel algorithm for VEC systems, integrating load balancing and offloading to optimize server selection and resource allocation. This algorithm shows swift convergence and superior performance compared to standard solutions, enhancing VEC system efficiency. However, scalability and the impact of varying network conditions require further exploration. Despite this, numerical results demonstrate its effectiveness in improving VEC system performance.

Muhammad Saleh Bute et al. [14] propose two algorithms for efficient task offloading in VEC networks. The service vehicles selection algorithm chooses suitable service vehicles based on criteria like link lifetime and task processing cost, while the Task Offloading Decision Algorithm evaluates factors such as link lifetime and task completion time to determine task offloading. Although challenges remain, such as achieving longer link durations in highway scenarios and optimizing energy efficiency, the proposed scheme shows promising results in various VEC network conditions. Huned Materwala et al. [15] proposed evolutionary genetic algorithm and used GPU based linear Regression models to analyze the energy consumption and explored three different offloading methods namely random, genetic algorithm and no offloading and the parameters considered are the energy consumption, latency and number of requests. The

papers explore various techniques, including fuzzy inference, reinforcement learning, genetic algorithms, and deep reinforcement learning, to optimize task offloading decisions. While these approaches demonstrate promising results in terms of resource utilization, energy efficiency, and task completion time, they also face limitations and challenges. Some of the common issues include scalability, robustness in dynamic environments, privacy concerns, and limitations in addressing specific network conditions. The research finding indicates that task offloading is a crucial aspect of vehicular edge computing and that ongoing research is necessary to address the challenges and further improve the performance of these techniques.

SYSTEM MODEL

The system setup involves a number of user equipment (UE) and edge server (ES), with tasks being dynamically generated for each UE by a function according to the task arriving probability. These tasks vary in size between the minimum size and maximum size configured and computation density, which are crucial parameters in determining whether a task should be processed locally or offloaded. The model initializes important parameters such as computation capacity, transmission capacity for each user equipment and the edge server, computational and transmission delay which represents the responsive time and energy consumption. The model includes separate entities to store the utility function calculated which is later compared and used in strategy update in the offloading.

Computation model

In the concept of edge computing, the efficient offloading of tasks between the user equipment and the edge plays an important role in system performance optimization. If the task is done locally the utility calculation considers parameters such as computation capacity, task size, computation delay and computation density. In contrast, when the task is offloaded, the utility function takes additional parameters including the transmission capacity of the UE, computation capacity of the edge, task size, computation density, computation and transmission delay, and the current edge loads.

Communication model

In the communication model between user equipment and edge server, the decision to offload tasks is guided by key performance parameters ensuring efficient task processing. When tasks are executed locally within UEs, the primary consideration is the computational capacity and computation delay of the UE and the complexity of the task. However, when tasks are transferred to the edge server for processing, additional variables are included. These include the computation capacity of the edge server, which determines its ability to handle the computational demands of offloaded tasks effectively. Additionally, the transmission capacity of the UE is assessed to ensure smooth data transmission to the edge server. The transmission delay parameter is critical as it sets the upper limit on task completion time. Furthermore, the current edge loads are evaluated to know about the availability of resources for task processing in the edge. Considering these parameters, the communication model aims to optimize task offloading decisions and system responsiveness to ensure efficient utilization of edge computing resources.

INTEGRATED PARTICLE SWARM OPTIMIZATION AND GENETIC ALGORITHM FOR EDGE COMPUTING

An innovative method for maximizing work distribution and resource usage in edge computing (EC) settings is the PSO+GA integrated algorithm. This approach provides a comprehensive solution to the problems caused by finite computational resources and dynamic network conditions in automotive contexts by merging the particle swarm optimization (PSO) and genetic algorithm (GA). Through directing each particle towards its own optimal solution, the cognitive component promotes the exploitation of promising regions in the solution space. The solution is computed as the difference between the particle’s present location and its optimal position. Similarly, by navigation particles towards the global optimal solution that the entire swarm has discovered, the social element promotes particle exploration of new regions in the solution space. The result is calculated as the discrepancy between the particle’s present location and the optimal position globally.

Each particle’s velocity in the solution space is updated based on the social and cognitive

components, determining its direction and speed of travel. PSO iteratively updates particle locations and velocities to efficiently explore the solution space. The velocity update formula incorporates learning coefficients (c_1 and c_2) and a random number using the random function to balance exploration and exploitation:

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (pbest_i - x_i(t)) + c_2 \cdot r_2 \cdot (gbest - x_i(t)) \tag{1}$$

where: velocity update of a particle is $V_i(t + 1)$, weight is w , the acceleration coefficient c_2 , r_1 and r_2 refers the random values $[0,1]$, personal best in swarm is $pbest_i$, global best is $gbest$, and the current position is $X_i(t)$.

A sigmoid function then translates the velocity into a new position for each particle. The sigmoid function typically takes the form:

$$Position = \frac{1}{(1 + e^{-v})} \tag{2}$$

This function transforms the velocity (ranging from negative to positive infinity) into a probability between 0 and 1. This probability is then used to determine whether a bit in the binary solution vector is set to 1 (local processing) or 0 (offload to edge server). The fitness of a solution (particle position) is calculated as the total utility, often involving formulas that consider factors like processing times on local UE vs. the edge server and power consumption. The stochastic optimization algorithm known as the "genetic algorithm" draws inspiration from the principles of natural selection and evolution. The population of candidate solutions are represented as chromosome. The algorithm goes through crossover, mutation, and selection processes in order to gradually evolve towards better solutions. By simulating biological processes, the genetic operators make it easier to explore and utilize the solution space. Fitter individuals are chosen for reproduction. Selection is the process of selecting individuals from the population using fitness. In order to create children with characteristics inherited from both parents, crossover combines genetic material from specific individuals. By introducing haphazard modifications to the genetic makeup of the progeny, mutation fosters genetic diversity and delays premature convergence.

$$P_{select} = \frac{\sum Fit_{Population}}{Fit_{Individual}} \tag{3}$$

where: Probability of selection is P_{select} , Fitness of all individuals in population is $Fit_{Population}$ and Fitness of individual is $Fit_{Individual}$.

GA progresses towards optimal solution and explores the solution space using successive generations of selection, crossover, and mutation. Effective search of the solution space is ensured by genetic operators and selection pressure, which maintain a balance between exploration and exploitation. PSO+GA integrated algorithm combines the strengths of PSO and GA to optimize task allocation in VEC environments. In this approach, PSO is employed for global exploration of the solution space, refining solutions iteratively and guiding particles towards promising regions. Meanwhile, GA introduces genetic diversity to the search space, promoting the discovery of more efficient task allocation strategies through selection, crossover, and mutation operations. The proposed system navigates the solution space to find optimal task allocation strategies by integrating PSO and GA. The synergy between PSO's swarm intelligence and GA's evolutionary principles enhances the system ability to adapt to dynamic VEC environments, minimizing latency and maximizing resource utilization.

PSO is well-suited for global exploration of the solution space, while GA excels at local exploitation of promising regions. By integrating both algorithms, the PSO+GA approach achieves a balance of the exploitation, maximizing the likelihood of finding optimal solutions. Vehicular networks are characterized by dynamic conditions, including varying traffic and network conditions. The PSO+GA algorithm's adaptability allows it to the environment, continuously optimizing task allocation and resource utilization. PSO's swarm

intelligence and GA's genetic diversity accelerate the convergence process, enabling faster identification of optimal solutions. The iterative refinement and genetic exploration mechanisms work in tandem to efficiently search the converge towards high-quality solutions. The PSO+GA algorithm is scalable and flexible, capable of accommodating diverse problem domains and evolving requirements (Figures 1–3).

SIMULATION SETUP

The simulation is tailored for the study of task offloading in VEC and it is done in python 3.10 using the NumPy library which is particularly suited for handling large, multi-dimensional arrays and executing mathematical functions on these arrays. The important attributes for the simulation which are listed in Table 1. The VEC Simulation class encapsulates the core logic of the simulation. The class orchestrates Initialization, evaluates the utilities for the UE and Edge Server based on capacity, task size, delay and energy consumed and the perform the computational offloading based on the algorithms implemented.

To introduce stochasticity, the simulation employs the random library, which simulates the probabilistic nature of task arrivals and the decision-making processes of UEs and edge servers. During execution, the simulation progresses through multiple episodes, where tasks are generated, utilities are computed, and strategies are updated iteratively. Upon completion, the

Algorithm 1 Integration of Particle Swarm optimization and Genetic Algorithm

```

1: Input : Configuration parameters for UEs and Edge Servers
2: Output : Strategy profiles for UEs and Edge Servers, Task processing rates
3: procedure SIMULATION
4:   Initialize configuration parameters
5:   Initialize simulation instance with configuration
6:   Run simulation
7: end procedure
8:
9: function Run_Simulation
10: for episode ← 1 to N_EPISODE do
11:   Generate tasks
12:   if episode is a multiple of (N_EPISODE/5) then
13:     Evaluate performance
14:   end if
15:   if PSO performance > GA performance then
16:     Run PSO
17:   else
18:     Run GA
19:   end if
20: end for
21:

```

Figure 1. Integration of particle swarm optimization and genetic algorithm

Algorithm 2 Particle Swarm Optimization

```

1: procedure PSO
2:   Initialize best global fitness as  $-\infty$ 
3:   Initialize swarm of particles
4:   for all particles in swarm do
5:     Initialize particle's best position and fitness
6:   end for
7:   for iteration  $\leftarrow 1$  to PSO_MAX_ITER do
8:     for all particles in swarm do
9:       Update particle's velocity
10:      Update particle's position
11:      Calculate fitness
12:      if fitness > particle's best fitness then
13:        Update particle's best position and fitness
14:      end if
15:      if fitness > best global fitness then
16:        Update best global position and fitness
17:      end if
18:    end for
19:  end for
20: end procedure

```

Figure 2. Implementation of particle swarm optimization

Algorithm 3 Genetic Algorithm

```

1: procedure GA
2:   Initialize best global fitness as  $-\infty$ 
3:   Initialize population of individuals
4:   for all individuals in population do
5:     Evaluate fitness
6:     if fitness > best global fitness then
7:       Update best global fitness and position
8:     end if
9:   end for
10:  for iteration  $\leftarrow 1$  to GA_MAX_ITER do
11:    Create new population
12:    for all individuals in population do
13:      Select parents
14:      Perform crossover (using a crossover rate CROSSOVER_RATE)
15:      Perform mutation (using a mutation rate MUTATION_RATE)
16:      Evaluate fitness
17:      if fitness > best global fitness then
18:        Update best global fitness and position
19:      end if
20:    end for
21:  end for
22: end procedure

```

Figure 3. Implementation of genetic algorithm

Table 1. Simulation attributes

Parameter	Values
N_{UE}	20
N_{ES}	5
$N_{EPISODE}$	500
$T_{askarrival_prob}$	0.5
Minimum S_{task}	2
Maximum S_{task}	6
<i>PSO_MAX_ITER</i>	100
<i>PSO_POP_SIZE</i>	20
<i>GA_MAX_ITER</i>	100
<i>GA_POP_SIZE</i>	20
<i>GA_CROSSOVER_RATE</i>	0.5
<i>GA_MUTATION_RATE</i>	0.4

simulation produces results, offering insights into the efficacy of task processing within the VEC framework. In addition to the setup, it's vital to highlight certain parameters, such as the crossover rate, mutation rate, and probability of replicator dynamics, which are constants within the simulation. These values are determined through extensive exploration of their efficiency within the current scenario, as depicted in Figure 4 of the simulation framework.

In Figure 4a, determining the ideal crossover rate, it was observed that the system performed efficiently when the crossover rate in the Genetic Algorithm was fixed at 0.6 which is indicated in the line graph. The mutation rate,

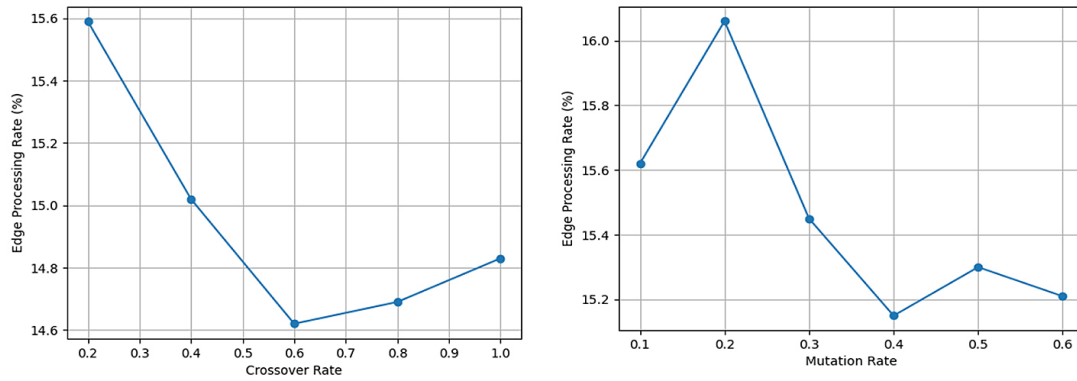


Figure 4. (a) edge processing rate vs crossover rate (b) edge processing rate vs mutation rate

illustrated in Figure 4b, is a parameter controlling how likely individual chromosomes change during evolution. The iteration gives 0.4 is the optimal mutation rate.

RESULTS AND ANALYSIS

Based on the simulation setup and parameter values in Table 1, the number of tasks executed in User Equipment is more than the number of tasks executed in edge as only the resource intensive tasks are offloaded to the edge server. While offloading parameters such as delay, capacity, energy consumption is also considered. Figure 5 represents the number of tasks executed in each user equipment and the overall tasks processed locally is 4254 out of 4975 and number of tasks processed

at edge is 721 out of 4975 which gives a task distribution rate of 85.51 % and 14.49% respectively.

The number of tasks executed in local and edge by varying the computation and transmission capacity is illustrated in Figure 6. The result indicates the transmission capacity factor is increased by 0.4, the number of tasks executed in the edge is decreased by an average of 20 tasks. The reason is the increase in transmission delay by 0.63% even though the edge device has more power. Similarly, for the proposed integrated algorithm, the number of tasks processed at device is increased by 2%.

The data presented in Figure 7 illustrates a notable trend regarding task offloading behavior concerning task size variations. Specifically, when the task size, indicative of task intensity, is increased, the average number of tasks shifted

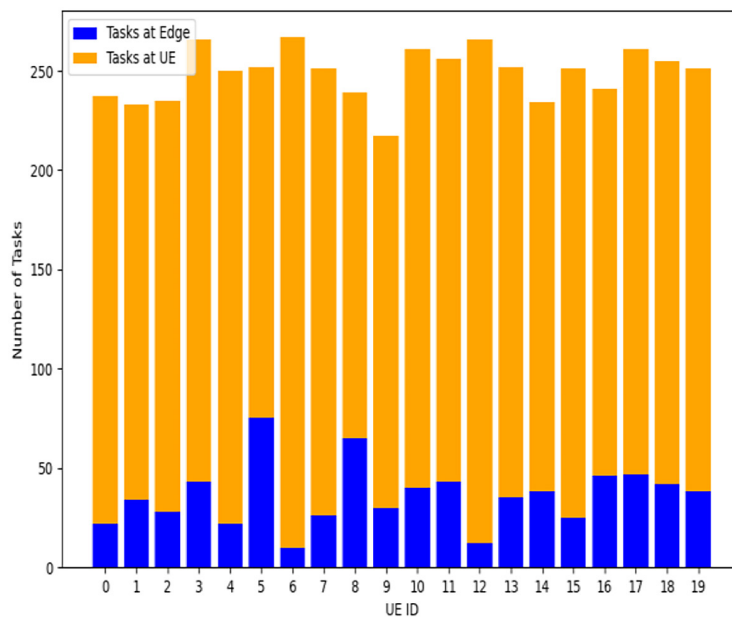


Figure 5. Distribution of tasks at edge and user equipment

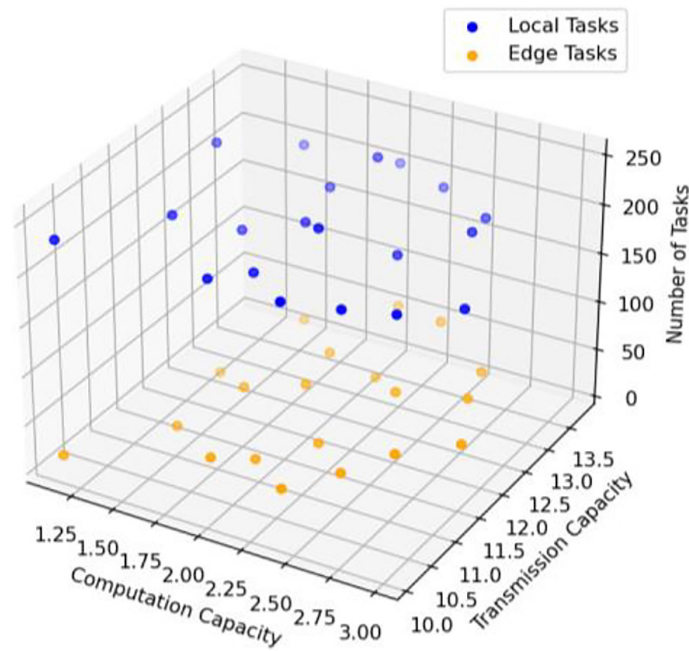


Figure 6. Task distribution on computation and transmission capacity

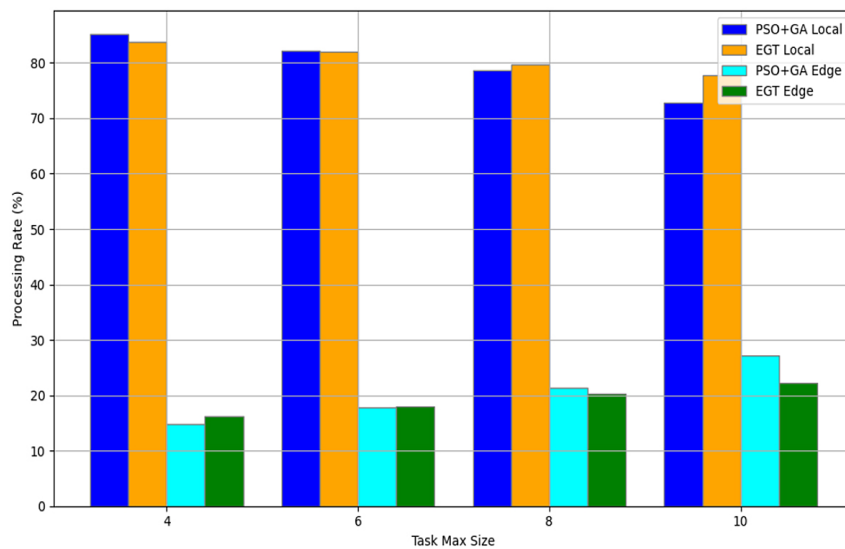


Figure 7. Effect of task size on processing rate

to the edge varies significantly between different algorithms. For instance, when employing the PSO+GA algorithm, the average number of tasks offloaded to the edge increases substantially, reaching an average of 25 tasks. Utility factor indicates the energy consumption at each User equipment, this energy is consumed due to two factors. The energy taken by the system to complete the task locally and the energy consumed to transmit the task to the edge server. If the Utility factor is less, it indicates that the energy consumed at that particular User equipment is less. A

trend in system performance can be found when comparing against the computation and system delay shown in Figure 8. In UE 12, due to the high Computation and Transmission delay, it can be noted that the system does not perform well.

System performance is calculated to find the overall performance of the algorithm which will consider delay and energy consumed by UE and edge devices, with a similar trend to what is observed in Figure 9, when the number of tasks increases, PSO+GA tends to have a better performance.

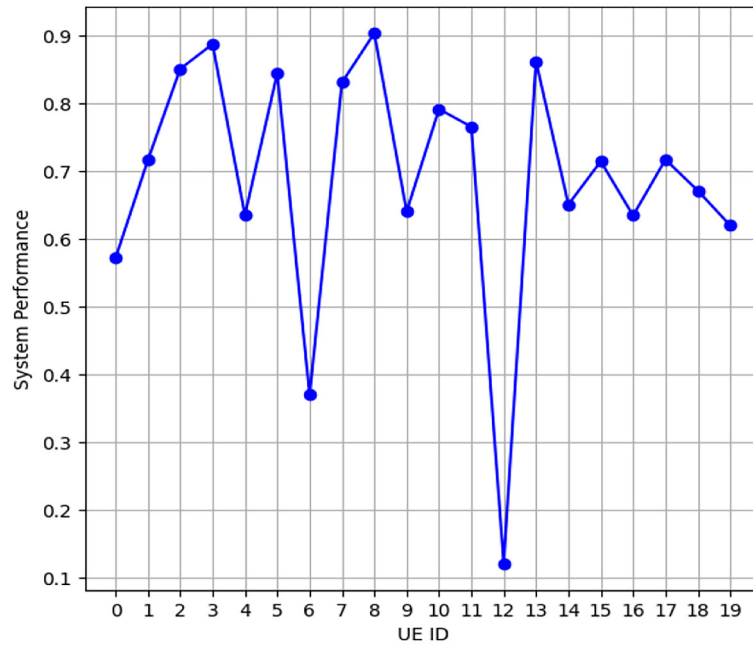


Figure 8. System performance employing PSO+GA

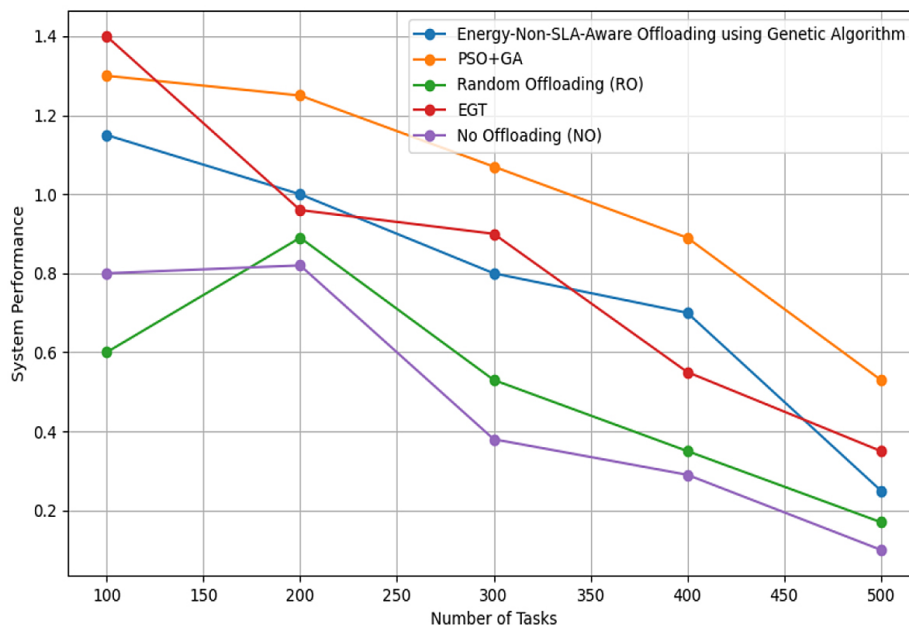


Figure 9. System performance of different algorithms for various number of tasks

CONCLUSIONS

In this paper, random strategy, energy aware offloading and the integration of genetic algorithm and particle swarm optimization, were examined, each demonstrating distinct strengths across varied scenarios. The GA+PSO displayed enhanced adaptability to varying task sizes, resulting in an average of 25 tasks offloaded to the edge, surpassing the performance of Random offloading.

Energy efficiency varied, with PSO+GA showing lower consumption overall. Notably, system performance correlated with computation and transmission delays, emphasizing their optimization’s importance. When compared to already existing algorithms, the proposed PSO+GA outperformed these algorithms in system performance and less energy consumption. Further analysis revealed trends in task offloading behavior related to work size variations and consumption of energy, as

well as utility factors and system performance aspects across different UEs and algorithms. These findings provide critical insights into future vehicular edge computing techniques and algorithmic optimizations.

REFERENCES

1. Liu L, Chen C, Pei Q, Maharjan S, Zhang Y. Vehicular edge computing and networking: A survey. *Mobile Network Application*. 2021; 26(3): 1145–68.
2. Shabariram CP, Shanthi N, Priya Ponnuswamy P. Computational offloading in vehicular edge computing using multiple agent-based deterministic policy gradient algorithm and generative adversarial networks. *International Journal of Ad Hoc and Ubiquitous Computing*. 2023; 44(4): 209–20.
3. Meneguette R, De Grande R, Ueyama J, Filho GPR, Madeira E. Vehicular edge computing: architecture, resource management, security, and challenges. *ACM computing survey*. 2023; 55(1): 1–46.
4. Geng L, Zhao H, Wang J, Kaushik A, Yuan S, Feng W. Deep-reinforcement-learning-based distributed computation offloading in vehicular edge computing networks. *IEEE Internet of Things Journal*. 2023; 10(14): 12416–33.
5. Raza S, Wang S, Ahmed M, Anwar MR. A survey on vehicular edge computing: architecture, applications, technical issues, and future directions. *Wireless Communications and Mobile Computing*. 2019; 2019: 1–19.
6. Qi W, Xia X, Wang H, Xing Y. A task partitioning and offloading scheme in vehicular edge computing networks. In: 2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall). IEEE; 2021.
7. Cao S, Liu D, Dai C, Wang C, Yang Y, Zhang W, Zheng, D. Reinforcement learning based tasks offloading in vehicular edge computing networks. *Computer Networks*. 2023; 234(109894): 109894.
8. Zhang Z, Zeng F. Efficient task allocation for computation offloading in vehicular edge computing. *IEEE Internet of Things Journal*. 2023; 10(6): 5595–606.
9. Maleki H, Başaran M, Durak-Ata L. Handover-enabled dynamic computation offloading for vehicular edge computing networks. *IEEE Transaction on Vehicular Technology*. 2023; 72(7): 9394–405.
10. Ismail L, Materwala H, Hassanein HS. QoS-SLA-aware adaptive genetic algorithm for multi-request offloading in integrated edge-cloud computing in Internet of vehicles. *arXiv [cs.NI]*. 2022.
11. Dai X, Xiao Z, Jiang H, Chen H, Min G, Dustdar S. A learning-based approach for vehicle-to-vehicle computation offloading. *IEEE Internet of Things Journal*. 2023; 10(8): 7244–58.
12. Singh S, Kim DH. Joint optimization of computation offloading and resource allocation in C-RAN with mobile edge computing using evolutionary algorithms. *IEEE Access*. 2023; 11: 112693–705.
13. Dai Y, Xu D, Maharjan S, Zhang Y. Joint load balancing and offloading in vehicular edge computing and networks. *IEEE Internet of Things Journal*. 2019; 6(3): 4377–87.
14. Bute MS, Fan P, Zhang L, Abbas F. An efficient distributed task offloading scheme for vehicular edge computing networks. *IEEE Transaction on Vehicular Technology*. 2021; 70(12): 13149–61.
15. Materwala H, Ismail L, Shubair RM, Buyya R. Energy-SLA-aware genetic algorithm for edge–cloud integrated computation offloading in vehicular networks. *Future Generation Computing Systems*. 2022; 135: 205–22.