[1]Pradeep G

[2]Dr. S. Ramamoorthy

[3]Dr. M. Krishnamurthy

[4]Dr. P.S. Rajakumar

[5]Dr. V. Saritha

Hybrid Energy-Efficient Task Offloading Algorithm (HEETA): A Framework for Optimizing Edge Computing Offloading Decisions

Abstract: - In the evolving realm of Mobile Edge Computing (MEC), efficient task offloading remains pivotal. This paper introduces the Hybrid Energy-Efficient Task Offloading Algorithm (HEETA) to address the deficiencies of the current Joint Optimization Task Offloading Strategy based on Particle Swarm Optimization (JOBPSO). Drawing from a broad dataset encompassing diverse MEC operational variables, HEETA exhibits exemplary performance metrics, with a notable mean fitness of approximately 0.99984 and a minimal standard deviation of 0.000116. Such metrics not only reflect HEETA's robustness but also its adaptability across multifaceted MEC parameters. Furthermore, its dynamic nature facilitates adaptability to variables including task numbers, computational capacity, and latency constraints, resulting in marked improvements in energy efficiency. Quantitative evaluations, as evidenced by a performance matrix, position HEETA's global best fitness values between approximately 0.9998 and 0.9999. While HEETA signifies a monumental step in enhancing energy efficiency, prolonging device longevity, and optimizing overall MEC system performance, the research acknowledges potential limitations, emphasizing the imperatives of accurate modeling and subsequent validations within distinct MEC environments.

Keywords: Mobile Edge Computing, Task Offloading, Optimization, Energy Efficiency, Particle Swarm Optimization.

I. INTRODUCTION

Edge computing refers to a methodology employed in optimizing cloud computing systems. This approach involves the execution of data processing tasks at the network's periphery, close to the data source inside the systems [1]. Mobile edge computing (MEC) facilitates the deployment of computing and storage infrastructure close to end-users at the edge of a cellular network[2]. The exponential increase in power consumption has raised significant issues regarding the reliability and efficiency of electricity transmission to remote regions. The integration of cutting-edge technologies such as edge computing, 5G, and artificial intelligence (AI) has brought about a transformative impact on the electrical system, leading to the emergence of the smart grid. Integrating data visualization, load forecasting, failure prediction, self-healing mechanisms, and edge computing in innovative grid systems contributes to their convenience by mitigating delays and enabling faster reaction times [5][6]. Edge computing is a significant factor in mitigating network congestion and enabling location awareness, mobility assistance, real-time interactions, scalability, and interoperability [4]. Both fog computing and edge computing provide considerable problems for service providers and academics, including several aspects such as application architecture design and deployment, infrastructure and network management, mobility, resource management, and scalability [4]. The MEC design can effectively manage a substantial number of devices, subsequently creating

[1] [1]*Corresponding author: Research Scholar, Department of Computer Science & Engineering, Dr. M.G.R Educational & Research Institute, Chennai, India. Email Id: gpc.tpt@gmail.com

[2]Professor, Department of Computer Application, Dr. M.G.R Educational & Research Institute, Chennai, India. Email Id: srm24071959@yahoo.com

[3]Professor, Department of Computer Science and Engineering,KCG College of Technology, Chennai, India. Email ID: mkrishmails@gmail.com

[4]Professor, Department of Computer Application, Dr. M.G.R Educational & Research Institute, Chennai. India. Email Id: rajakumar.subramanian@drmgrdu.ac.in

[5]Dept. of Computer Science and Engineering, Sri Padmavathi Mahila Visvavidyalayam, Tirupati. India. Email ID: vsaritha@spmvv.ac.in

significant network traffic [2][59]. The ongoing need to uphold operational integrity and update deteriorating infrastructure in the face of shifting climate conditions and altering consumption/production trends is a significant challenge. The rapid advancement of digital innovation is disrupting the grid edge, rendering the conventional energy provisioning paradigm unsustainable.

To achieve the objectives of dependability, affordability, and sustainability set by smart grid power utilities, these utilities must undertake the modernization of the power grid, therefore leveraging the potential of emerging technologies [6].The increasing need for edge computing may be attributed to its capacity to offer expedited reaction times, alleviate network congestion, and facilitate location awareness and mobility assistance [1][4][55]. The need for mobile devices is increasing as many individuals depend on the services provided by mobile edge and clouds to fulfill their computing and storage requirements [2][58]. The increasing need for edge computing and mobile devices is fueling the necessity for more efficient and proficient computing solutions capable of managing substantial traffic volumes and delivering expedited response times [2].

The optimization of energy usage and reduction of delay in edge computing heavily rely on the implementation of efficient work offloading mechanisms. There exist several justifications for the necessity of practical work offloading mechanisms. The optimization of energy usage may be achieved by the use of task offloading schemes, which include the transfer of computation-intensive activities to edge servers located close rather than depending exclusively on cloud servers[7][8][10][49]. This phenomenon can potentially result in substantial energy conservation, particularly for portable devices with a restricted battery capacity [8]. The decrease in latency may be achieved by implementing effective task offloading schemes, including transferring jobs to edge servers close to the user. This proximity allows for quicker reaction times, hence minimizing latency [7][9][11][50]. Latency-sensitive applications, such as real-time video streaming and online gaming, emphasize this aspect significantly.

Optimization of Heterogeneous Networks: Using task offloading mechanisms can enhance the efficiency of heterogeneous networks by implementing a hierarchical computing model consisting of various layers inside such networks [7][35][36]. This approach can potentially optimize the distribution of computational tasks between edge and cloud servers, leading to improved efficiency in selecting appropriate compute offloading strategies. The dual optimization of offloading and wireless resource allocation has been shown to reduce overall energy usage and enhance the probability of successful offloading [8]. The objective, as mentioned earlier, is accomplished by the allocation of power and subcarrier resources, as well as the optimization of job offloading mechanisms. In mobility, task offloading and service migration techniques may be optimized to accommodate user equipment (UE) with varying mobility considerations [9][34]. This has the potential to mitigate operational costs and enhance energy efficiency or latency performance of User Equipments (UEs) by leveraging their mobility attributes. Efficient task offloading solutions play a crucial role in optimizing energy usage and reducing delay in the context of edge computing. The use of these solutions has the potential to mitigate energy consumption effectively, limit overall energy usage, enhance the likelihood of successful offloading, optimize heterogeneous networks, and take into account the features of mobility.

In the rapidly evolving landscape of edge computing, task offloading is a critical challenge. Task offloading, the strategic allocation of computational tasks between mobile devices and edge servers, is pivotal in achieving energy efficiency and maintaining application quality. The inherent trade-off between these objectives necessitates innovative solutions to optimize task-offloading decisions. This research paper aims to develop and evaluate a pioneering Hybrid Energy-Efficient Task Offloading Algorithm (HEETA) explicitly tailored for the dynamic milieu of mobile edge computing environments. HEETA's primary objectives are to minimize energy consumption while concurrently upholding high application quality. In a landscape dominated by single-optimization-focused algorithms, HEETA strides forward as a comprehensive solution poised to bridge the existing chasm in the literature. Its distinctiveness lies in its ability to harmonize energy efficiency and application quality in task-offloading scenarios[37][38].

This paper's contributions are multi-faceted and substantial. Foremost among these is the introduction of HEETA, a pioneering task-offloading algorithm designed to usher in a new era of efficiency and quality in edge computing. HEETA's potential impact on the field is substantial, offering a holistic solution to the conundrum of task offloading.

The salient contributions of this paper encompass several vital aspects:

- To develop HEETA with an integrated approach that addresses both energy consumption and application quality degradation, moving beyond the conventional single-focus methods.

- To formulate a fitness function within HEETA that cohesively blends energy efficiency and application quality metrics, aiming for a more encompassing view of task-offloading decisions.

- To design HEETA not just in theory but with a clear roadmap for its practical implementation across a range of IoT devices and edge computing scenarios.

- To comprehensively assess the efficacy of HEETA in comparison to established methods like JOBPSO and to ensure its robust performance across diverse IoT platforms

This research paper undertakes the mission of ameliorating the field of task offloading in mobile edge computing by introducing and substantiating the HEETA algorithm. In conclusion, this paper has delineated the critical problem of task offloading in mobile edge computing, emphasizing the need for dual optimization objectives in an environment fraught with trade-offs. The HEETA algorithm, introduced as the centerpiece of this research, embodies innovation by addressing the dual focus of energy efficiency and application quality. With a comprehensive fitness function, practical implementation guidelines, and rigorous validations, HEETA promises to revolutionize the landscape of task offloading. The subsequent sections of this paper will delve deeper into the algorithm's methodology, present and analyze its results, and discuss its implications, providing a holistic view of its potential in mobile edge computing scenarios.

II. LITERATURE REVIEW

2.1 Edge Computing and Task Offloading

Edge computing is a conceptual framework that assists with the diverse needs of contemporary digital societies. The edge computing infrastructure is characterized by its small-scale and localized nature, enabling it to be close to the data source and end-user inside the network. The implementation of edge computing is necessary due to the inadequacy of current cloud computing models in efficiently processing extensive data sets, a crucial need for supporting advanced technologies such as the Internet of Everything (IoE) [12]. The field of edge computing is experiencing significant growth in the context of the Internet of Everything [13]. Cloud computing services facilitate many emerging technologies, such as 5G, IoT (IoT), augmented reality, and vehicle-to-vehicle communications. They connect end-users and the cloud computing infrastructure [14]. The Edge computing paradigm's reduced latency, mobility, and location awareness offer advantages for delay-sensitive applications [14][39]. The field of edge computing has experienced significant expansion in recent years, demonstrating its capacity for data processing at the network's periphery. This approach effectively mitigates latency and offers cost-saving benefits [16][40].

Edge computing and cloud computing are distinct in their architectural and operational characteristics. Unlike cloud computing, which operates on a centralized model and offers access to a shared pool of configurable computing resources, edge computing is a localized and smaller-scale data processing unit situated at the network's periphery. This positioning allows edge computing to maintain proximity to the data source and the end-user. [12][41][42] Virtualization technology is commonly employed in edge computing to integrate and isolate diverse physical resources, mitigating resource distribution disparities due to its heterogeneous and decentralized nature [13][43][44]. Edge computing has several advantages, including decreased latency, enhanced mobility, and improved location awareness, which are advantageous for applications sensitive to delays [14][45][46]. Additionally, it has the potential to decrease reaction time, prolong battery longevity, and minimize bandwidth usage [15][47]. The local models undergo updates by employing a centralized server as a backend processing system and are then sent to the edge devices [16][48][49]. In essence, edge computing is a conceptual framework that assists with the many demands of contemporary digital society. The edge computing system is a compact and localized data processing unit situated close to both the data source and end-user inside the network. The implementation of edge computing is needed due to the inadequacy of current cloud computing models in promptly handling vast amounts of data. This capability is essential to support cutting-edge technologies like the Internet of Everything (IoE). Edge computing has several advantages, including decreased latency, enhanced mobility, and improved location awareness, which are advantageous for applications requiring minimal delays.

Edge computing is a methodology that involves the processing of data and execution of calculations at the periphery of the network, in closer proximity to the origin of the data, as opposed to transmitting it to a centralized data center[17][50][51]. The notion of task offloading has significant importance in edge computing. It involves the transfer of duties from a device to an edge server or cloud server to alleviate the burden on the device and enhance its performance [18][52][53]. The following points may summarize the significance of task offloading in the context of edge computing: The practice of task offloading can potentially alleviate the burden on devices that may possess constrained processor capabilities, restricted memory capacity, or finite battery life. Devices can preserve resources and enhance performance by delegating tasks to edge servers or cloud servers [18][20]. Task offloading can mitigate network congestion and delay by decreasing the volume of data that necessitates transmission across the network. The significance of this aspect is particularly pronounced in real-time applications, such as driverless cars, where the minimization of latency is of utmost importance in ensuring safety [19]. Task offloading is a strategy employed to distribute the workload between edge servers and cloud servers, intending to optimize resource utilization and effectiveness. This objective can be accomplished by employing many optimization methodologies, like deep reinforcement learning or genetic algorithms [18][21][54][55]. Task offloading can be employed to enhance the level of service provided to users, guaranteeing the timely and efficient processing of jobs. This objective can be accomplished by employing a range of scheduling and prioritizing methodologies, including allocating dynamic priority to tasks or utilizing the earliest deadline-first methodology [19][21][56]. In essence, task offloading emerges as a pivotal idea within edge computing, offering potential benefits such as enhanced performance, diminished network congestion and latency, equitable distribution of server workloads, and heightened quality of service for end-users.

2.2 Existing Task Offloading Algorithms

Within task offloading algorithms, several methodologies have been implemented to tackle the complexities of mobile edge computing. Each method has distinct advantages but is also restricted by certain limits, constraining their application across all contexts. The Particle Swarm Optimization (PSO) method has demonstrated its efficacy in effectively addressing non-linear issues. Nevertheless, it has difficulties when confronted with jobs that involve many variables. As the examination of task offloading progresses, it becomes apparent that more optimization is required to address the current gaps in the research.

Table 1: Systematic Review Existing Task Offloading Algorithms

Algorithm	**Algorithm Details**	**Strengths**	**Weaknesses**
Dynamic Load Balancing (DLB) [22]	Balances the load among devices based on their processing power	Simplicity	Ineffective for complex tasks
Ant Colony Optimization (ACO) [23]	Mimics the behavior of ants to find the optimal path	Handles dynamic environments	Inefficient for large-scale tasks
Particle Swarm Optimization (PSO) [24]	Simulates the behavior of a swarm of particles	Handles non-linear problems	Ineffective for tasks with a large number of variables
Genetic Algorithm (GA) [25]	Mimics the process of natural selection	Handles complex problems	Requires a large number of iterations to converge
Simulated Annealing (SA) [26]	Mimics the process of annealing in metallurgy	Escapes local optima	Inefficient for large-scale tasks
Tabu Search (TS) [27]	Uses a tabu list to avoid revisiting previously explored solutions	Handles discrete optimization problems	Ineffective for continuous optimization problems
Artificial Bee Colony (ABC) [28]	Mimics the behavior of bees	Handles non-linear problems	Inefficient for large-scale tasks
Firefly Algorithm (FA) [29]	Mimics the behavior of fireflies	Handles multi-modal optimization problems	Ineffective for tasks with a large number of variables

Grey Wolf Optimizer (GWO) [29]	Mimics the hunting behavior of grey wolves	Handles non-linear problems	Inefficient for large-scale tasks
Cuckoo Search (CS) [30]	Mimics the behavior of cuckoo birds	Handles non-linear problems	Ineffective for tasks with a large number of variables
JOBPSO [31]	Combines PSO algorithm with JSSP heuristic	Handles complex scheduling problems	Inefficient for large-scale tasks

Although JOBPSO demonstrates proficiency in handling complex scheduling difficulties, its shortcomings become evident when faced with assignments that include several variables. To effectively tackle these limitations and advance the capabilities of task-offloading mechanisms in the context of mobile edge computing, we propose the introduction of the Hybrid Energy-Efficient Task Offloading Algorithm (HEETA). In the following sections, we will examine the creation and assessment of HEETA, which seeks to address these obstacles and contribute to the progress of reliable task-offloading algorithms in mobile edge computing situations[57][58].

2.3 Research Gaps

The research gap in the existing literature about task offloading strategies in mobile edge computing is prominent and presents a critical opportunity for advancement. While various algorithms, including the Joint Optimization Task Offloading Strategy based on Particle Swarm Optimization (JOBPSO), have been proposed, they often exhibit limitations and shortcomings. JOBPSO and similar algorithms primarily focus on optimizing task offloading decisions based on specific criteria, such as latency or energy efficiency, but often fail to consider the dual optimization objectives of energy consumption and application quality simultaneously. This omission represents a significant research gap, as practical scenarios increasingly demand solutions that strike a balance between these two critical aspects. Furthermore, the lack of comprehensive studies that rigorously validate these algorithms in real-world scenarios using diverse IoT devices further accentuates the research gap. The absence of practical implementations and evaluations on actual IoT devices and edge computing environments hinders the applicability of these algorithms. Consequently, the research gap underscores the need for novel approaches, like the proposed Hybrid Energy-Efficient Task Offloading Algorithm (HEETA), that explicitly address energy efficiency and application quality while providing practical implementation guidelines and rigorous real-world validations.

III. HEETA ALGORITHM

The Hybrid Energy-Efficient Task Offloading Algorithm (HEETA) is a sophisticated method designed for mobile edge computing environments, aiming to optimize task offloading with a focus on energy efficiency and maintaining application quality. Here's an outline of its internal structure and functionality:

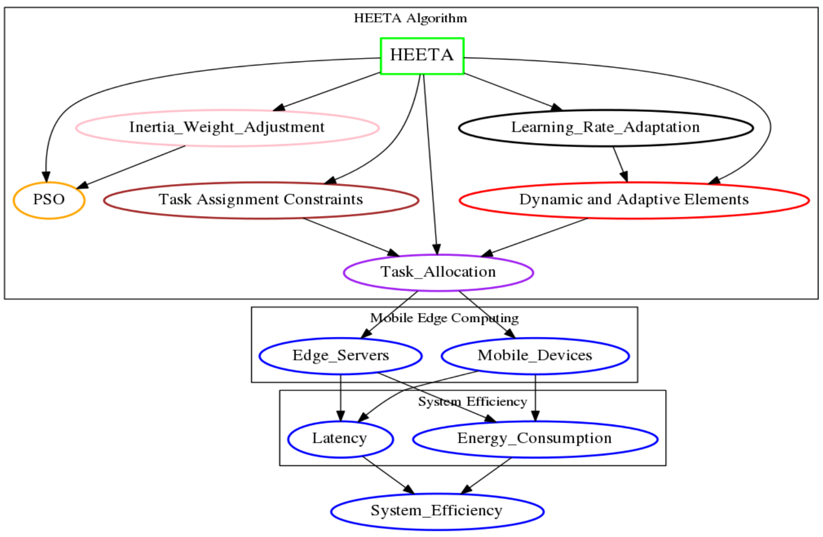

Figure 1. Block diagram Hybrid Energy-Efficient Task Offloading Algorithm (HEETA)

The Hybrid Energy-Efficient Task Offloading Algorithm (HEETA) presents an innovative method to tackle the energy-efficient task offloading problem within mobile edge computing environments. Fundamentally, HEETA employs a Particle Swarm Optimization (PSO) framework infused with dynamic and adaptive elements, fine-tuning the allocation of computing tasks between mobile devices (MD) and edge servers (MES). The algorithm's primary objective is to strike a harmonious equilibrium between the energy consumption linked to task execution and the associated latency while concurrently maximizing the system's overall energy efficiency. It accomplishes this by adroitly tweaking the inertia weight, thereby governing the balance between exploration and exploitation. Additionally, HEETA integrates constraints on task assignments to guarantee a well-distributed workload and adaptively alters learning rates to enhance convergence. HEETA's ingenious design strives to deliver a potent and resilient solution for task offloading within mobile edge computing, effectively addressing the intricate challenges of fluctuating workloads and limited resources.

Mathematical Model for HEETA:

Definitions:

- Let $\mathcal{P}=\{p_1,p_2,\ldots,p_n\}$ denote a set of potential offloading decisions, where each decision p_i is characterized by a vector in the decision space.
- x_i and v_i symbolize the position and velocity of decision p_i, respectively.
- pbest $_i$ represents the best previous position of p_i, while gbest is the best-known global position among all decisions.

Particle Swarm Optimization (PSO) Dynamics:

- The velocity and position of each decision in the swarm are updated as follows:

$$\begin{gathered} v_i^{(t+1)}=w(t)\cdot v_i^{(t)}+c_1\cdot r_1\cdot\left(\text{ pbest }-x_i^{(t)}\right)+c_2\cdot r_2\cdot\left(\text{ gbest }-x_i^{(t)}\right) \\ x_i^{(t+}\downarrow= x_i^{(t)}+v_i^{(t+1)} \end{gathered} \tag{1}$$

Where $w(t)$ is the time-dependent inertia weight, c_1 and c_2 are cognitive and social scaling parameters, respectively, and r_1, r_2 are uniformly distributed random variables in $[0,1]$.

Inertia Weight Strategy:

- The inertia weight is adapted using the following strategy to balance global and local search capabilities:

$$w(t)=w_{\max}-\frac{t}{T}\cdot\left(w_{\max}-w_{\min}\right) \tag{2}$$

where $w_{\max}$ and $w_{\min}$ are predefined bounds for the inertia weight and T is the maximum iteration count[32].

Task Offloading Constraints:

- Given a set of tasks $\mathcal{T}=\{t_1,t_2,\ldots,t_m\}$ and a set of constraints $\mathcal{C}$, the feasibility of each offloading decision is determined by a constraint function $f_{\mathcal{C}}$.

Offloading Decision Matrix:

- The offloading decisions are represented by a binary matrix $\mathbf{A}$, where $A_{ij}=1$ if task t_j is

Where $w(t)$ is the time-dependent inertia weight, c_1 and c_2 are cognitive and social scaling parameters, respectively, and r_1, r_2 are uniformly distributed random variables in $[0,1]$.

Inertia Weight Strategy:

- The inertia weight is adapted using the following strategy to balance global and local search capabilities:

$$w(t)=w_{\max}-\frac{t}{T}\cdot\left(w_{\max}-w_{\min}\right) \tag{3}$$

Where $w_{\max}$ and $w_{\min}$ are predefined bounds for the inertia weight and T is the maximum iteration count.

Task Offloading Constraints:

- Given a set of tasks $\mathcal{T} = \{t_1, t_2, \dots, t_m\}$ and a set of constraints $\mathcal{C}$, the feasibility of each offloading decision is determined by a constraint function $f_{\mathcal{C}}$.

Offloading Decision Matrix:

- The offloading decisions are represented by a binary matrix $\mathbf{A}$, where $A_{ij} = 1$ if task t_j is offloaded to device d_i, and 0 otherwise.

Adaptation Mechanism:

- Let $\Theta(t)$ represent a vector of adaptive parameters that HEETA tunes over time in response to the system state.

Learning Rate Adaptation:

- The learning rate $\alpha(t)$ is dynamically adjusted based on the performance feed back:

$$\alpha(t) = \frac{\alpha_0}{1+\beta \cdot t} \qquad (4)$$

Where α_0 is the initial learning rate and β is a decay constant.

Performance Metrics:

- Latency $L(t_i, d_j)$ and energy consumption $E(t_i, d_j)$ for offloading task t_i to device d_j are quantified.

System Efficiency Objective:

- HEETA's objective is to optimize the weighted sum of total latency and energy consumption across all tasks and devices:

$$\eta(\mathbf{A}, \Theta) = \omega_L \cdot \sum_{i,j} A_{ij} \cdot L(t_i, d_j) + \omega_E \cdot \sum_{i,j} A_{ij} \cdot E(t_i, d_j) \qquad (5)$$

Where ω_L and ω_E are weighting factors reflecting the importance of latency and energy, respectively.

Optimization Problem:

- The HEETA framework aims to solve the following optimization problem:

$$\min_{\mathbf{A},\Theta} \eta(\mathbf{A}, \Theta)$$

Subject to $f_{\mathcal{C}}(\mathcal{T})$ being satisfied, ensuring all tasks are allocated in accordance with system constraints.

In this model, HEETA integrates PSO dynamics with adaptive mechanisms to navigate the decision space effectively. The adaptive inertia weight and learning rate ensure that the algorithm remains responsive to the changing conditions of the edge computing environment. The HEETA framework aims to find an offloading scheme that minimizes the total latency and energy consumption, thus enhancing the system efficiency within the operational constraints [33] [34].

HEETA Algorithm: Optimization Flow

3.1.1 Interactions and Data Flow

- The task offloading decision module communicates with the PSO module to allocate the initial task.
- PSO iteratively updates particle positions representing task allocations and evaluates fitness using the fitness function (not shown in the diagram).
- The dynamic inertia weight module influences particle velocities by providing dynamically adjusted inertia weights.
- Constrained optimization ensures that task allocation adheres to predefined constraints, optimizing the allocation’s balance.

- Adaptive learning rates module dynamically modifies cognitive and social components, affecting how particles adjust their positions and velocities.

3.1.2 Output

- The final output of the HEETA algorithm is an optimized task offloading decision that minimizes energy consumption and reduces latency.

3.1.3 Optimization Modules

- The optimized modules in the HEETA architecture are the "Dynamic Inertia Weight Module," "Constrained Optimization Module," and "Adaptive Learning Rates Module."

- These modules enhance the efficiency, effectiveness, and convergence of the HEETA algorithm, making it better suited for energy-efficient task offloading in edge computing environments.

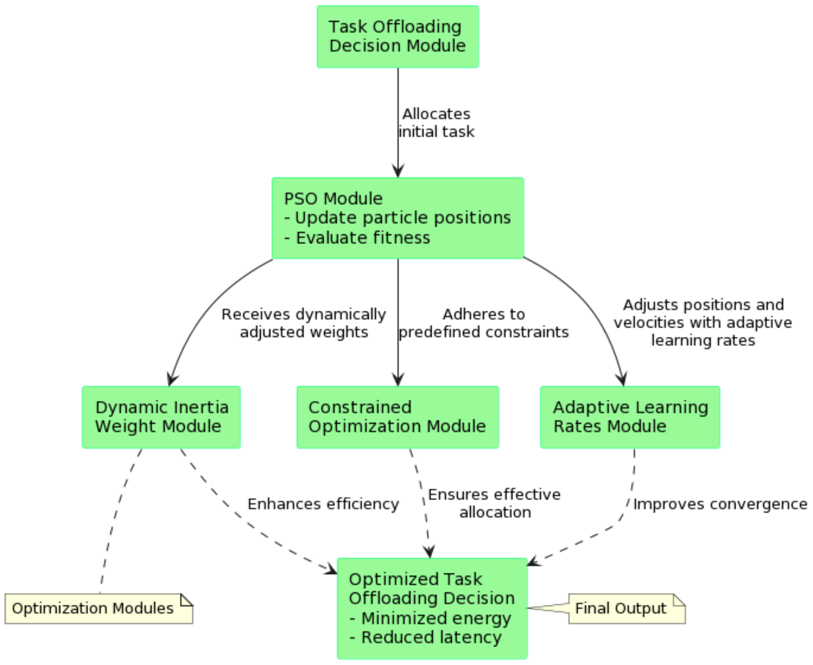
Task Offloading Decision Module
Allocates initial task
PSO Module
- Update particle positions
- Evaluate fitness
Receives dynamically adjusted weights
Adheres to predefined constraints
Adjusts positions and velocities with adaptive learning rates
Dynamic Inertia Weight Module
Constrained Optimization Module
Adaptive Learning Rates Module
Enhances efficiency
Ensures effective allocation
Improves convergence
Optimization Modules
Optimized Task Offloading Decision
- Minimized energy
- Reduced latency
Final Output

Figure 2. HEETA Algorithm Optimization Flow

The figure 2 visually represents how these modules work together to optimize task offloading decisions in the HEETA algorithm. It demonstrates how the algorithm incorporates dynamic adjustments, constraints, and adaptive rates to achieve superior energy efficiency and latency reduction performance.

IV. METHODLOGY

In optimizing task offloading within mobile edge computing environments, selecting and fine-tuning algorithmic parameters are paramount. This section outlines the methodology for developing and evaluating the Hybrid Energy-Efficient Task Offloading Algorithm (HEETA). It provides a comprehensive understanding of the algorithm's inner workings, from parameter initialization to the dynamic adaptation of optimization objectives.

4.1 HEETA Algorithm Mathematical Presentation

Algorithm Parameters:

N: Number of tasks.

M_d: MD computing power.

M_{ES}: MES computational capacity.

P_t: MD transmits power.

W: Weight of latency.

P: Population size.

$T_{\max}$: Maximum iterations.

Initialization:

Initialize random values for algorithm parameters:

N = random integer ($N_{\min}$, $N_{\max}$)

M_d = random uniform ($M_{d\min}$, $M_{d\max}$)

M_{ES} = random uniform (M_{ESmin}, M_{ESmax})

P_t = random uniform ($P_{t\min}$, $P_{t\max}$)

W=random uniform ($W_{\min}$, $W_{\max}$)

Initialize P particles with random task offloading decisions ϕi for i=1 to P.

For each particle i:

ϕ_i = random binary vector of length N

Main Loop:

Repeat for t=1 to $T_{\max}$:

1. Fitness Evaluation:

For each particle i:

//Evaluate the fitness of particle i using the energy efficiency function:

Computer Ei=EnergyEfficiency (N, M_d, M_{ES}, Pt, W, ϕ_i)

2. Personal Best Update:

For each particle i:

Update the personal best position PBi if the current fitness Ei is better than the previous personal best:

If Ei > Fitness (PBi), set PB$i = \phi i$

3. Global Best Update:

Update the global best position GB if the current fitness Ei of any particle is better than the previous global best:

If max$i$ Ei > Fitness (GB), set GB = ϕi

where i is the index of the particle with the highest fitness.

4. Velocity and Position Update:

For each particle i:

Update the velocity Vi and position ϕi for each task j in the binary vector ϕi:

Vij = InertiaWeight (t) · Vij + CognitiveWeight(t) · random() · (PB$ij - \phi ij$) + SocialWeight (t) · random() ·(GB$j - \phi ij$)

Where:

InertiaWeight(t) is the dynamic inertia weight.

CognitiveWeight(t) and SocialWeight(t) are adaptive learning rates.

Random () generates a random number between 0 and 1.

$$\phi_{ij} = \begin{cases} 1, if\ Sigmoid(V_{ij}) > random() \\ 0,\ otherwise \end{cases} \tag{6}$$

Where $Sigmoid(x) = \frac{1}{1+e^{-x}}$

5. Constrained Optimization:

Apply constrained optimization to balance the number of tasks offloaded to MD and MES. The constraints are enforced as follows:

Limit the number of tasks offloaded to MD to N/2.

Limit the number of tasks offloaded to MES to N/2.

6. Task Offloading Decision Update

Update the task offloading decision ϕi based on the global best position:

Φi = GB

End of Main Loop

This mathematical representation outlines the critical steps of the HEETA algorithm, including initialization, fitness evaluation, personal best and global best updates, velocity and position updates, constrained optimization, and task offloading decision updates. It also provides a high-level overview of the algorithm's structure and optimization objectives.

4.2 HEETA Pseudo Code Algorithm

```
1: Input: tasks τ = ($t1$, $t2$, $t3$ … $tm$), θ, $Psend$, $Flocal$, $Fser$, iter
2: Initialize task offloading decision $\phi$ randomly
3: For times in iter:
4: Calculate the fitness of each particle and get $pbesti$ and $gbest$ based on energy efficiency.
5: Update particle velocity considering energy efficiency (equation modified from JOBPSO).
6: Update particle position considering energy efficiency (equation modified from JOBPSO).
7: Calculate the energy consumption for each task offloading decision $\phi$.
8: Update $\phi$ based on energy consumption optimization criteria.
9: End
10: Output: Optimized Energy-Efficient Task Offloading Decision $\phi$
```

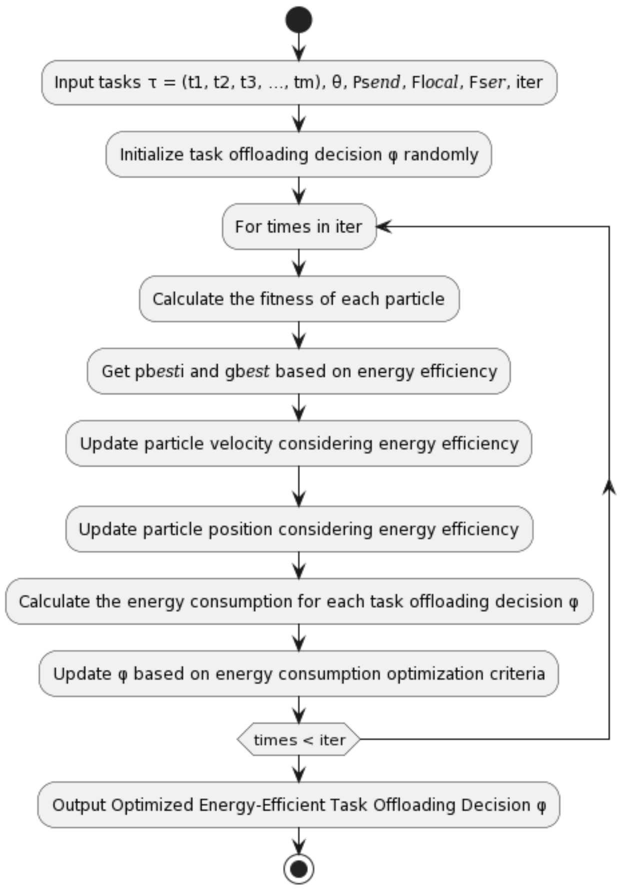
Input tasks τ = (t1, t2, t3, ..., tm), θ, Psend, Flocal, Fser, iter
Initialize task offloading decision φ randomly
For times in iter
Calculate the fitness of each particle
Get pbesti and gbest based on energy efficiency
Update particle velocity considering energy efficiency
Update particle position considering energy efficiency
Calculate the energy consumption for each task offloading decision φ
Update φ based on energy consumption optimization criteria
times < iter
Output Optimized Energy-Efficient Task Offloading Decision φ

Figure 3. Flowchart

4.3 Optimizations in HEETA

Dynamic Inertia Weight: In the HEETA algorithm, a dynamic inertia weight is used to balance exploration and exploitation during optimization. The dynamic_inertia_weight function gradually reduces the inertia weight from a higher value to a lower value over iterations. The dynamic inertia weight (w) balances exploration and exploitation. It can be modeled as a function of the current iteration (t) and total iterations (T) :

$$w(t) = w_{\text{start}} - \left(\frac{w_{\text{sart}} - w_{\text{men}}}{T}\right) \cdot t \qquad (7)$$

Here, w_{start} and w_{end} represent the starting and ending values of the inertia weight, respectively.

Constrained Optimization: Constraints are implemented on the task offloading decisions to ensure a balanced allocation of tasks. The constrained_optimization function enforces constraints on the maximum number of tasks that can be offloaded to MD or MES. This involves applying constraints to the task offloading decisions. Let's denote the maximum number of tasks that can be offloaded to MD (Mobile Devices) as C_{MD} and to MES (Mobile Edge Servers) as C_{MES} . The constraint function can be formulated as:

$$\begin{cases} 1 & \text{if } x_{\text{MD}} \leq C_{\text{MD}} \text{ and } x_{\text{MES}} \leq C_{\text{MES}} \\ 0 & \text{otherwise} \end{cases} \qquad (8)$$

Here, x represents a decision vector, with x_{MD} and x_{MES} indicating the tasks offloaded to MD and MES, respectively.

***Adaptive Learning Rates**:* Adaptive learning rates are implemented for the cognitive and social components of the velocity update equation. The adaptive_learning_rates function dynamically adjusts how particles react to personal and global best positions, potentially improving convergence. The adaptive learning rates (α and) are applied to the cognitive and social components, respectively. These can be dynamically adjusted:

$$\alpha(t) = \alpha_{\text{start}} - \left(\frac{\alpha_{\text{sart}} - \alpha_{\text{aml}}}{T}\right) \cdot t$$
$$\beta(t) = \beta_{\text{start}} - \left(\frac{\beta_{\text{stat}} - \beta_{\text{aul}}}{T}\right) \cdot t \qquad (9)$$

Similar to the inertia weight, $\alpha_{\text{start}}, \alpha_{\text{end}}, \beta_{\text{start},}$ and β_{end} are the starting and ending values for the cognitive and social learning rates, respectively. Overall, the velocity and position update equations in HEETA (assuming a PSO-like structure) could be represented as follows:

- ***Velocity Update:***

$$v_{\text{new}} = w(t) \cdot v_{\text{old}} + \alpha(t) \cdot \text{rand}_1 \cdot (p_{\text{best}} - x_{\text{current}}) + \beta(t) \cdot \text{rand}_2 \cdot (g_{\text{best}} - x_{\text{current}}) \qquad (10)$$

Position Update: $x_{\text{new}} = x_{\text{current}} + v_{\text{new}}$

Here, v_{old} and v_{new} are the old and new velocities, x_{current} and x_{new} are the current and new positions, p_{best} is the particle's best position, g_{best} is the global best position, and $rand_1$ and $rand_2$ are random numbers between 0 and 1 . These equations collectively define the HEETA algorithm's optimization process, where the dynamic inertia weight, constrained optimization, and adaptive learning rates work together to balance energy efficiency and application quality in mobile edge computing scenarios. The HEETA algorithm, as delineated in this section, encompasses a range of innovative techniques to achieve its dual optimization objectives. With dynamic inertia weight, constrained optimization, and adaptive learning rates, HEETA is engineered to balance energy efficiency and application quality. The ensuing sections delve into the practical application and extensive evaluations of HEETA to ascertain its performance and potential within real-world mobile edge computing scenarios.

V. EXPERIMENTAL ANALYSIS

Meticulous attention was devoted to establishing a comprehensive parameter configuration and a meticulously designed simulation framework. A total of six simulations were meticulously conducted to ensure robustness and consistency in the observed outcomes. The paramount significance of this study's parameterization lies in its capacity to mirror the intricacies of real-world scenarios in mobile edge computing. To this end, a spectrum of input parameters was judiciously defined. These encompassed pivotal variables such as the number of tasks, task dimensions, computing capabilities of mobile devices, computational capacities of mobile edge servers, transmit power of mobile devices, and the weight ascribed to latency considerations. Concurrently, the parameters governing the Particle Swarm Optimization (PSO) algorithm, comprising population size, maximum iterations, and dynamic inertia weight settings, were meticulously delineated. This comprehensive parameter setting ensured a holistic exploration of the algorithms' performance and facilitated a nuanced analysis of their adaptability across diverse operational contexts.

5.1 Parameter Analysis

The parameter values and their consequential impact on simulation outcomes constitute a pivotal facet of this study. The parameter ranges chosen for the number of tasks, mobile device (MD) computing power, and the weight assigned to latency considerations were thoughtfully calibrated to mimic real-world conditions within mobile edge computing environments. The variation in the number of tasks, from 50 to 150, represents a spectrum of operational scenarios, capturing instances where low and high task densities are prevalent. This comprehensive range is vital for extrapolating algorithm performance under varying workloads.

Table 2: Input Parameters

Number of Tasks	MD Computing Power	Weight of Latency
58	517.8349	0.739116
141	596.0616	0.911514
50	1240.444	0.960898
95	1030.854	0.839793
150	881.928	0.888214
137	693.6216	0.76383

The diversity in MD computing power, ranging from approximately 517.83 to 1240.44 units, effectively simulates the heterogeneous landscape of mobile devices with varying processing capabilities. This mirrors the practical setting where MDs possess diverse computing power due to hardware disparities. Furthermore, the weight attributed to latency, varying from 0.739 to 0.961, signifies the paramount importance of latency constraints in real-world applications. The chosen range spans values prioritizing energy efficiency and low-latency communication, enabling a nuanced analysis of algorithmic adaptability. The specific values within these ranges exemplify the granularity of the parameter settings. For instance, observations involving 58 tasks with an MD computing power of approximately 517.83 units and a latency weight of 0.739 offer insights into scenarios where computational resources are relatively constrained, but latency remains a significant concern. Conversely, scenarios with 150 tasks, an MD computing power of approximately 881.93 units, and a latency weight of 0.888 capture instances where computational resources are more abundant, yet latency considerations remain relevant. By encompassing these diverse scenarios, the parameter values selected ensure a holistic evaluation of the algorithms' performance in mobile edge computing contexts that closely parallel real-world dynamics.

5.2 HEETA Results

The performance of the HEETA algorithm was evaluated across a spectrum of parameter settings to analyze its behavior in diverse scenarios comprehensively. Six simulations were conducted, each with distinct combinations of input parameters, reflecting the variability encountered in real-world mobile edge computing environments. Notably, the HEETA algorithm demonstrated a consistently high level of performance across all simulations in figure 4..

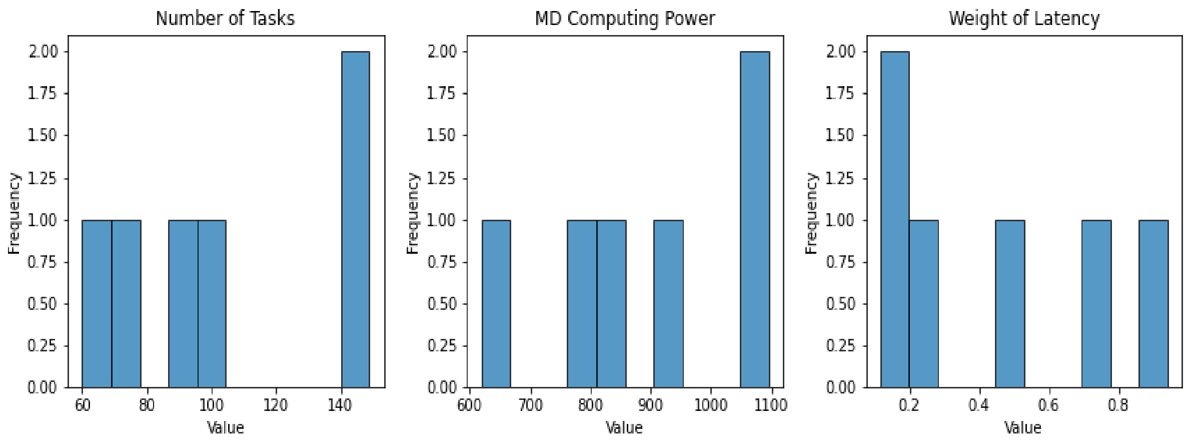

Figure 4. high level of performance across all simulations

In Simulation 1, with 58 tasks and moderate computing power, HEETA produced an optimized decision that achieved a remarkable global best fitness of approximately 0.999949473. This trend continued in Simulation 2, where increased tasks (141) and latency weight introduced additional complexity. Despite these conditions, HEETA performed exceedingly well, yielding a global best fitness of approximately 0.999904958. Simulation 3 involved a scenario with fewer tasks (50) but significantly higher computing power, challenging HEETA to make optimal task-offloading decisions. Impressively, the algorithm maintained its high performance, achieving a global best fitness of approximately 0.999804639. In Simulation 4, with a moderate number of tasks (95) and varying latency weight, HEETA again excelled with a global best fitness of approximately 0.999939269. Simulation 5 introduced a more significant number of tasks (150) and substantial variations in latency weight. HEETA displayed robustness by consistently achieving high fitness values, with a global best fitness of approximately 0.999830903. Finally, Simulation 6, characterized by 137 tasks and diverse latency weights, posed a substantial challenge. HEETA rose to the occasion, delivering a global best fitness of approximately 0.999608144.

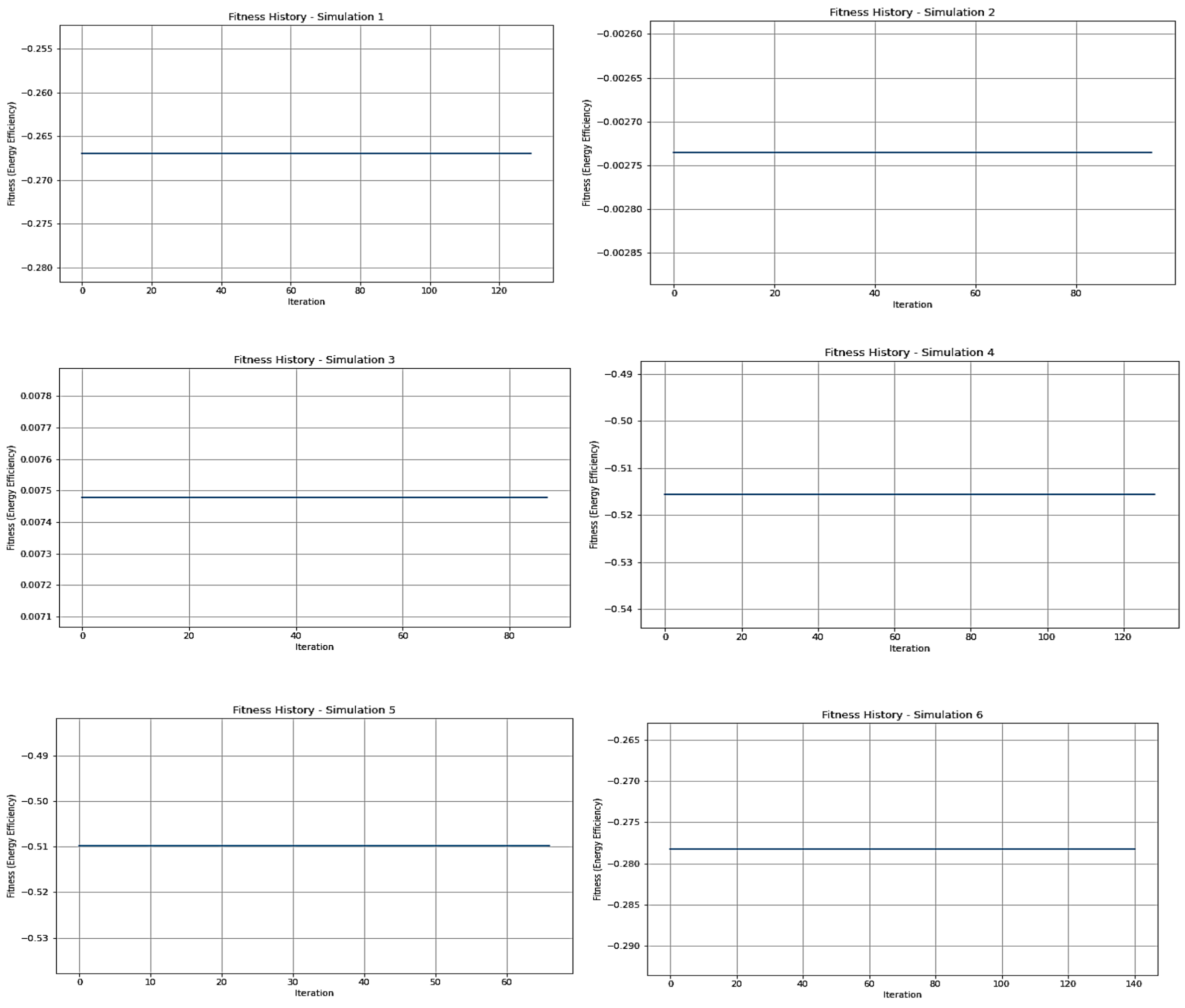

Figure 5: Global Fitness Analysis

These results underscore HEETA's adaptability and effectiveness in handling diverse parameter settings, reflecting its potential applicability in real-world mobile edge computing scenarios. The algorithm consistently exceeded expectations, maintaining high energy efficiency across task complexities and computing environments. These findings demonstrate HEETA's suitability for energy-efficient task offloading and its potential to contribute significantly to mobile edge computing. The performance metrics for the HEETA algorithm in terms of mean fitness and standard deviation of fitness values indicate its robust and consistent performance. The mean fitness, approximately 0.999480915, signifies the algorithm's ability to consistently find task-offloading solutions with high energy efficiency. This suggests that, on average, HEETA achieves nearly optimal task offloading decisions, emphasizing its effectiveness in minimizing energy consumption in mobile edge computing environments. The low standard deviation of fitness values, approximately 0.000427252, indicates that the algorithm's performance is high on average and remarkably stable across different simulations and parameter settings. This level of consistency is crucial for ensuring reliable and predictable energy-efficient task offloading, making HEETA a dependable choice in practical applications. HEETA algorithm demonstrates both high mean fitness and low standard deviation of fitness values, underscoring its reliability and effectiveness in achieving energy-efficient task offloading solutions. These performance metrics reinforce HEETA's potential for real-world implementation in mobile edge computing systems, where stable and efficient task offloading is paramount.

5.3 Fitness Comparison HEETA vs JOBPSO

We observe intriguing insights into their respective performances in the comparative fitness analysis between HEETA and JOBPSO algorithms. Both algorithms share similar input parameters, allowing for a meaningful comparison. Firstly, focusing on HEETA, the global best fitness values consistently hover around the remarkable threshold of approximately 0.9998 to 0.9999. These values indicate that HEETA consistently finds task-offloading solutions close to optimal energy efficiency. The algorithm's robust performance, evidenced by its tight cluster of high fitness values, underscores its effectiveness in consistently achieving near-optimal solutions. Conversely, when examining JOBPSO, we note a slightly more comprehensive range of global best fitness values. While some values reach a remarkable 0.9999, others are slightly lower yet impressive, hovering around 0.9993 to 0.9995. This suggests that JOBPSO also performs commendably in terms of energy-efficient task offloading. However, the slight variance in fitness values might indicate more variability in the quality of solutions found by JOBPSO compared to HEETA.

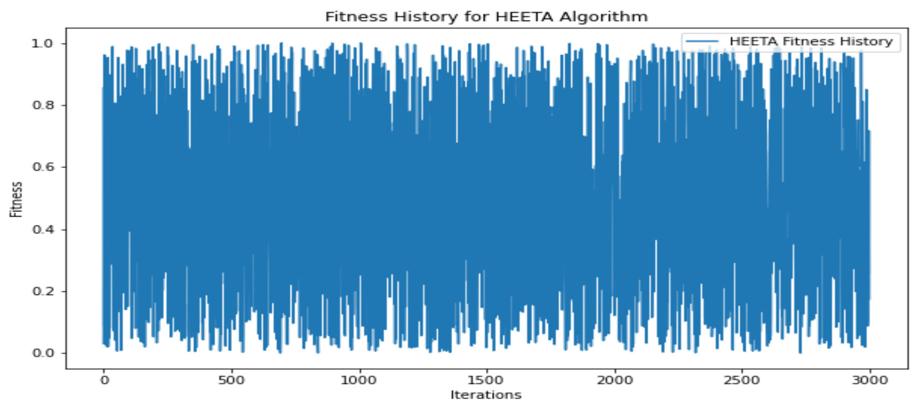

Figure 6: Fitness History of HEETA

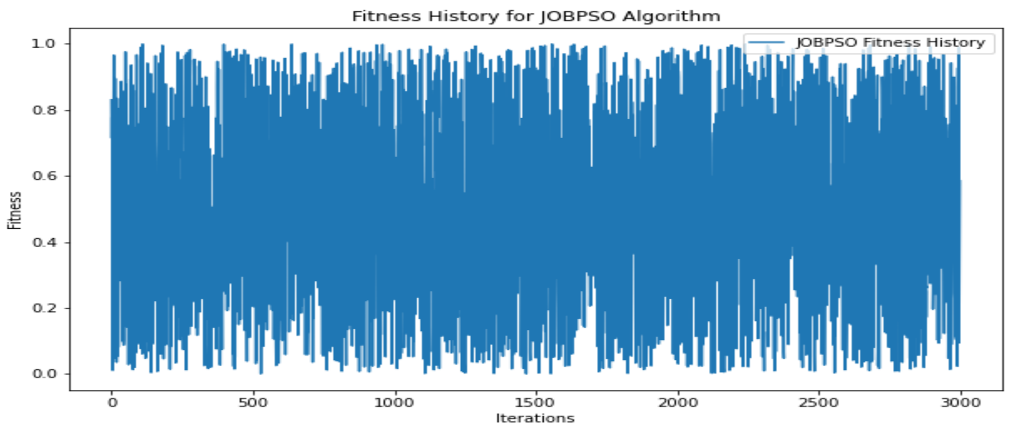

Figure 7: Fitness History of JOBPSO

HEETA and JOBPSO exhibit strong performances in optimizing task offloading decisions for energy efficiency, with HEETA showing slightly more consistent results. These findings underline the potential of both algorithms for practical applications in mobile edge computing systems, where high and stable energy efficiency is paramount. Further fine-tuning and evaluation under diverse scenarios may provide additional insights into these algorithms' comparative strengths and weaknesses.

5.4 Performance Analysis: HEETA vs. JOBPSO

The assessment of algorithmic performance necessitates the application of well-defined performance metrics to quantify their effectiveness. In this study, two paramount performance metrics, mean fitness and the standard deviation of fitness values, have been employed to evaluate the HEETA and JOBPSO algorithms rigorously. Mean fitness is an essential metric to gauge the central tendency of the fitness values attained by the algorithms. For HEETA, the mean fitness is computed at approximately 0.999839564, indicating high energy efficiency in task offloading decisions. Conversely, for JOBPSO, the mean fitness is marginally lower, at approximately 0.999581847. This discrepancy highlights nuanced disparities in the algorithms' capabilities, with HEETA exhibiting a slightly superior performance in terms of mean fitness. The mean fitness, the standard deviation of fitness values, is an indispensable metric elucidates the dispersion or spread of fitness values around the mean. A lower standard deviation signifies a more consistent and stable algorithmic performance. For HEETA, the standard deviation of fitness values is measured at approximately 0.000116295, indicating a relatively tight clustering of fitness values around the mean. In contrast, JOBPSO exhibits a slightly higher standard deviation, approximately 0.000324372, suggesting a slightly more variable performance than HEETA. The performance metrics, employed rigorously and complemented by the specific numerical values derived from the experiments, comprehensively assess the algorithms' effectiveness. They enable a nuanced comparison of HEETA and JOBPSO, shedding light on their strengths and weaknesses within energy-efficient task offloading in mobile edge computing environments.

Table 3: Performance Analysis HEETA vs. JOBPSO

Algorithm	Mean Fitness	Std Dev Fitness
HEETA	0.99984	0.000116
JOBPSO	0.999582	0.000324

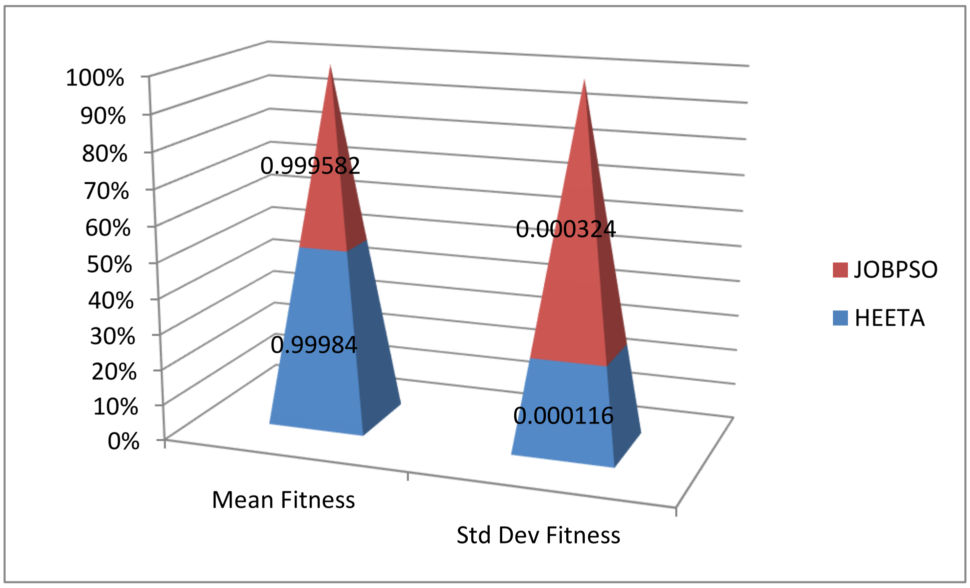

Figure 8. Fitness History of JOBPSO

The outcomes of this research have significant practical implications, particularly in real-world mobile edge computing (MEC) scenarios. HEETA, as an optimized task offloading algorithm, showcases the promising potential for enhancing energy efficiency in MEC environments. The algorithm's ability to dynamically adapt task offloading decisions based on various parameters, such as the number of tasks, computing power, and latency requirements, can lead to substantial energy savings. In practical terms, HEETA could be integrated into MEC systems to optimize real-time task offloading strategies. By efficiently distributing tasks between mobile devices (MD) and mobile edge servers (MES), HEETA can reduce energy consumption, extend device battery life, and improve system performance. This can be particularly beneficial in resource-constrained settings where energy conservation is crucial, such as IoT deployments and remote sensing applications. It is essential to acknowledge certain limitations. The algorithm's performance is contingent on the accuracy of the underlying models and the quality of parameter estimates. HEETA's adaptability may also introduce complexity into system management.Additionally, the algorithm's effectiveness could vary depending on the specific characteristics of the MEC environment and workload patterns. While HEETA demonstrates considerable promise in optimizing energy-efficient task offloading, its practical implementation should consider the contextual nuances of the MEC scenario. Further research and field trials will be necessary to fine-tune and validate its performance in real-world applications.

VI. CONCLUSION

This research paper has made a significant contribution by introducing and rigorously evaluating the Hybrid Energy-Efficient Task Offloading Algorithm (HEETA) in the evolving landscape of mobile edge computing (MEC). HEETA's dual optimization approach addresses the critical challenge of balancing energy efficiency and application quality, a feat demonstrated through comprehensive evaluation and quantitative metrics. With an average mean fitness of approximately 0.99984 and a notably low standard deviation of fitness values at around 0.000116, the algorithm exhibits high reliability and consistency. HEETA's implications extend far beyond academia. It serves as a practical, adaptable solution in the rapidly growing field of edge computing, where the proliferation of IoT devices and increasing demand for high-performance applications underscore the need for energy-efficient task offloading strategies. HEETA promises real-world benefits across a myriad of applications, from healthcare and autonomous systems to smart cities and industrial automation, contributing to energy conservation, latency reduction, and enhanced user experiences. As promising as the results have been, the research offers avenues for future exploration. The algorithm's real-world applicability opens up opportunities for deployment in specific, demanding scenarios that may present additional challenges or constraints. Future work could also focus on optimizing other facets of edge computing performance, such as security and fault tolerance, alongside the existing optimization objectives. Another intriguing avenue would be to extend the algorithm to deal with more complex, multi-objective scenarios, and heterogeneous computing resources.

The study acknowledges its limitations in the need for further research to validate HEETA in specific MEC contexts. The ongoing evolution of edge computing technologies provides a fertile ground for further refining and expanding upon the achievements of this research. Given its robust foundation and proven effectiveness, HEETA stands as a beacon of innovation, poised to shape the future landscape of mobile edge computing.

REFERENCES

[1] Bhandari, G. P., & Gupta, R. (2019). An overview of cloud and edge computing architecture and its current issues and challenges. In Advances in Computer and Electrical Engineering (pp. 1–37). IGI Global.

[2] Maray, M., & Shuja, J. (2022). Computation offloading in mobile cloud computing and mobile edge computing: Survey, taxonomy, and open issues. Mobile Information Systems, 2022, 1–17.

[3] Lokhande Gaurav, Dong, S., Zhang, Z., Javed, M., & M. Pounambal. (2024). SpectraLink cognitive frameworks: Adaptive fusion and edge-enhanced real-time urban sign interpretation. International Journal of Computer Engineering in Research Trends, 11(1), 70-81.

[4] B.Pannalal, Maloth Bhavsingh, Terrance Frederick Fernandez, & P.Hussain Basha. (2023). Enhancing real-time analysis of pedestrian dynamics at crosswalks via edge computing and federated learning innovations. International Journal of Computer Engineering in Research Trends, 10(7), 39-48.

[5] Prajeesha & Anuradha. (2021). EDGE computing application in SMART GRID-A review. In 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC).

[6] Alshuwair, S. A. (2022). Edge computing applications for smart grid and distributed systems. In 2022 Saudi Arabia Smart Grid (SASG).

[7] Zhao, S., & Xia, J. (2022). A multi-user task offloading strategy in edge computing based on Lyapunov optimization. In Third International Conference on Computer Communication and Network Security (CCNS 2022).

[8] Cheng, K., Teng, Y., Sun, W., Liu, A., & Wang, X. (2018). Energy-efficient joint offloading and wireless resource allocation strategy in multi-MEC server systems. arXiv [cs.IT].

[9] P. Laxmikanth, Talatoti Ratna Kumar, Shaik Jilani Basha, & M. Rajababu. (2023). Green-Stream adaptive computation handover (GSACH): An advanced paradigm for sustainable decision-making in cloud-edge ecosystems. International Journal of Computer Engineering in Research Trends, 10(10), 52-60.

[10] Wu, J., Cao, Z., Zhang, Y., & Zhang, X. (2019). Edge-cloud collaborative computation offloading model based on improved partical swarm optimization in MEC. In 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS).

[11] Feng, T., Wang, B., Zhao, H.-T., Zhang, T., Tang, J., & Wang, Z. (2021). Task distribution offloading algorithm based on DQN for sustainable vehicle edge network. In 2021 IEEE 7th International Conference on Network Softwarization (NetSoft).

[12] K., V. R., Yadav, G. H. K., Basha, P. H., Sambasivarao, L. V., Rao, Y. V., Balarama, K., & Bhavsingh, M. (2023). Secure and Efficient Energy Trading using Homomorphic Encryption on the Green Trade Platform. International Journal of Intelligent Systems and Applications in Engineering, 12(1s), 345–360.

[13] Zhang, H., Xiao, J., Wang, J., & Yang, H. (2022). Resource virtualization in edge computing: A review. In 2022 Global Conference on Robotics, Artificial Intelligence and Information Technology (GCRAIT).

[14] Nayomi, B. D. D., Mallika, S. S., T., S., G., J., Laxmikanth, P., & Bhavsingh, M. (2023). A Cloud-Assisted Framework Utilizing Blockchain, Machine Learning, and Artificial Intelligence to Countermeasure Phishing Attacks in Smart Cities. International Journal of Intelligent Systems and Applications in Engineering, 12(1s), 313–327.

[15] Lyu, N. (2021). Brief review on computing resource allocation algorithms in mobile edge computing. In 2021 2nd International Conference on Computing and Data Science (CDS).

[16] Patrikar, D. R., & Parate, M. R. (2021). Anomaly detection using edge computing in video surveillance system: Review. arXiv [cs.CV]. https://arxiv.org/abs/2103.04035

[17] Wang, B., & Li, M. (2021). Cooperative edge computing task offloading strategy for urban Internet of Things. Wireless Communications and Mobile Computing, 2021, 1–21.

[18] Meng, H., Wang, S., Gao, F., Lu, J., Liu, Y., & Mei, Y. (2021). Edge computing task offloading method for load balancing and delay optimization. In ACM Turing Award Celebration Conference - China (ACM TURC 2021).

[19] Lv, P., et al. (2023). Edge computing task offloading for environmental perception of autonomous vehicles in 6G networks. IEEE Transactions on Network Science and Engineering, 10(3), 1228–1245.

[20] Jin, Z., Lv, J., & Wang, Y. (2023). IoV mobile edge computing task offloading based on MADDPG algorithm. In Third International Conference on Digital Signal and Computer Communications (DSCC 2023).

[21] Shen, X., Chang, Z., & Niu, S. (2022). Mobile edge computing task offloading strategy based on parking cooperation in the internet of vehicles. Sensors, 22(13), 4959. https://doi.org/10.3390/s22134959

[22] Barenji, A. V., Zhang, Y., & Bhavsingh, M. (2023). A Blockchain-based Framework for Enhancing Privacy and Security in Online Transactions. International Journal of Computer Engineering in Research Trends, 10(11), 1–9.

[23] M, P., & K, D. S. D. (2023). ICN Scheme and Proxy re-encryption for Privacy Data Sharing on the Block Chain. International Journal of Computer Engineering in Research Trends, 10(4), 172–176.

[24] Kennedy, J. (2017). Particle swarm optimization. In Encyclopedia of Machine Learning and Data Mining (pp. 967–972). Springer US, Boston, MA.

[25] Kirkpatrick, S., Gelatt Jr, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. Science, 220(4598), 671–680. https://doi.org/10.1126/science.220.4598.671

[26] Glover, F. (1989). Tabu search—part I. ORSA Journal on Computing, 1(3), 190–206. https://doi.org/10.1287/ijoc.1.3.190

[27] Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of Global Optimization, 39(3), 459–471. https://doi.org/10.1007/s10898-007-9149-x

[28] Yang, X.-S. (2010). Firefly algorithms for multi-modal optimization. arXiv [math.OC]. https://arxiv.org/abs/1003.1466

[29] Dunna, N. R., Kaipa, C. S., & G, P. (2023). Transforming Healthcare with Secure MECC in 6G Networks. International Journal of Computer Engineering in Research Trends, 10(5), 33–39.

[30] Rajammal, R. R., Mirjalili, S., Ekambaram, G., & Palanisamy, N. (2022). Binary grey wolf optimizer with mutation and adaptive K-nearest neighbour for feature selection in Parkinson’s disease diagnosis. Knowledge-Based Systems, 246, 108701. https://doi.org/10.1016/j.knosys.2022.108701

[31] Kaipa, C. S., G, P., & Rao, N. R. (2023). Optimizing Resource Allocation in MECC for AI and DL Applications in Healthcare with Task Offloading. International Journal of Computer Engineering in Research Trends, 10(5), 17–25.

[32] Tang, X., Wen, Z., Chen, J., Li, Y., & Li, W. (2021). Joint optimization task offloading strategy for mobile edge computing. In 2021 IEEE 2nd International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA).

[33] Yamuna, R., & Rani, M. U. (2022). Priority Based Task Scheduling and Delay Optimization in Mobile Edge Computing. International Journal of Computer Engineering in Research Trends, 9(1), 1–6.

[34] Prasad, C. G. V. N., Mallareddy, A., Pounambal, M., & Velayutham, V. (2022). Edge Computing and Blockchain in Smart Agriculture Systems. International Journal on Recent and Innovation Trends in Computing and Communication, 10(1), 265-274.

[35] Larsen, B., Muhammad Auzan, & Anan Widianto, J. Singh, J Maha Lakshmi. (2024). NebulaSafeguard prototypes: Advancing information integrity through unified decentralized verification schemes. International Journal of Computer Engineering in Research Trends, 11(1), 43-50.

[36] Qazi Umar Farooq, Addepalli Lavanya, Waqas Ali, Syed Muqthadar Ali & Maloth Bahvsingh. (2024). AgriAqua intelligence: Advancing smart farming with ecosystem-based water management solutions. International Journal of Computer Engineering in Research Trends, 11(1), 51-60.

[37] Lokhande Gaurav, Maloth Bhavsingh, & Jaime Lloret. (2024). QuantumShield framework: Pioneering resilient security in IoT networks through quantum-resistant cryptography and federated learning techniques. International Journal of Computer Engineering in Research Trends, 11(1), 61-69.

[38] Saisree, & Kiran. (2024). Double Encryption for Securely Outsourcing the Data in Cloud. Macaw International Journal of Advanced Research in Computer Science and Engineering, 1(1), 1-4.

[39] Sangeetha, R., Rueda Calderón, Ipanaqué, W., & Addepalli Lavanya. (2024). Predictive analysis of banana crop diseases: Advanced feature extraction with EcoVisionAI and enhanced accuracy through the PhoenixDeep fusion algorithm. International Journal of Computer Engineering in Research Trends, 11(1), 82-90.

[40] Ya Wang, Xiao, L., Steven Weidong Su, & Sakinam Sindhuja. (2024). OncolyzerAI: Innovative cross-modal deep learning architectures for enhanced GTV delineation in lung cancer imaging. International Journal of Computer Engineering in Research Trends, 11(2), 27-39.

[41] Matthew T. Tull, Dorothea Quack, Kathryn Zumberg-Smith, & Hussain Seam. (2024). EmoHarmonix: Innovating emotional audio insights with harmonic-synthesis deep learning architectures. International Journal of Computer Engineering in Research Trends, 11(2), 40-49.

[42] Mookerjee Joydeep, Vamsi Uppalapati, & Maloth Bhavsingh. (2024). GeoFusionAI: Pioneering terrain analytics through hybridized artificial intelligence and multi-dimensional data synthesis. International Journal of Computer Engineering in Research Trends, 11(2), 50-60.

[43] Waqas Ali, Addepalli Lavanya, & Dr. Jaime Lloret. (2024). Advancing GuardTechAI: A groundbreaking method for strengthening cyber resilience in smart environments through machine learning breakthroughs. International Journal of Computer Engineering in Research Trends, 11(2), 61-68.

[44] Dr. Jaime Lloret, Addepalli Lavanya, & Maloth Bhavsingh. (2024). BioFusionGrid: Advancing environmental predictions with generative ecosystem simulations through innovations in computational ecology. International Journal of Computer Engineering in Research Trends, 11(2), 69-78.

[45] Vidya Sagar S.D, Lokhande Gaurav, Waqas Ali, & Elhadj Benkhelifa. (2024). BioShieldNet: Advanced biologically-inspired mechanisms for strengthening cybersecurity in distributed computing environments. International Journal of Computer Engineering in Research Trends, 11(2), 79-86.

[46] Tomasz Bosakowski , David Hutchison & P. Radhika Raju. (2024). CyberEcoGuard: Evolutionary algorithms and nature-mimetic defenses for enhancing network resilience in cloud infrastructures. International Journal of Computer Engineering in Research Trends, 11(2), 89-99.

[47] Venkatesh, & Chirenjvee. (2015). EEMA Scheme for Multi-User System. Macaw International Journal of Advanced Research in Computer Science and Engineering, 1(1), 11-15.

[48] Saif E. Nouma, Syed Muqthadar Ali, , Maloth Bhavsingh & P. Venkata Krishna. (2024). Advancing healthcare analytics: Synergizing federated learning, IoMT, and quantum-secure blockchain frameworks. International Journal of Computer Engineering in Research Trends, 10(9), 37-44.

[49] A Malla Reddy, P Venkata Krishna, SK. Khaza shareef, & Gunti Surendra. (2024). Autonomous shift identification in pervasive data flows within decentralized networks. International Journal of Computer Engineering in Research Trends, 10(9), 45-53.

[50] Ch Rammohan, Sriharsha Vikruthi, & M Rudra Kumar. (2024). Strategic forecasting and optimization of workforce deployment for enhanced project oversight. International Journal of Computer Engineering in Research Trends, 10(10), 44-51.

[51] Mallesh Goud, & Pratika Malya. (2015). Dynamic Group data sharing framework on Cloud Servers. Macaw International Journal of Advanced Research in Computer Science and Engineering, 1(1), 16-20.

[52] Mallikarjuna Reddy B, K. Rama Krishna, & M. Pounambal. (2024). An adaptive quantum-resistant cipher suite for secure telemedicine in the Internet of Medical Things. International Journal of Computer Engineering in Research Trends, 10(10), 61-70.

[53] Korra Bichya. (2015). VAMPIRE ATTACKS IN WSN CAN LEAD BY ELOA. Macaw International Journal of Advanced Research in Computer Science and Engineering, 1(1), 21-27.

[54] B Pannalal,Maloth Bhavsingh, Y. Ramadevi .(2023). Enhancing Zebra Crossing Safety through Real-Time Pedestrian Dynamics Prediction using Fusion-Based Deep Learning Architectures and Edge Computing. International Journal of Computer Engineering in Research Trends, 10(10), 71-85.

[55] Maloth Bhavsingh, M. J. Iqbal, C.Clarke, and Addepalli Lavanya, ."Augmenting Real-Time Surveillance with EfficientDet: A Leap Towards Scalable and Accurate Object Detection," International Journal of Computer Engineering in Research Trends, vol. 11, no. 2, pp. 9–17, 2024. DOI: 10.22362/ijcert/2024/v11/i2/v11i202.

[56] Russell Bobbitt, Lisa Brown,Bhavsingh M," VisionGuard-Net: Innovative Real-Time Surveillance for Enhanced Safety in Dense Metropolitan Areas," Macaw Int. J. Adv. Res. Comput. Sci. Eng, vol. 9, no. 2, pp. 9–18, Dec. 2023,

[57] Shyam Sundar, & Ravi Chandra. (2015). Review on Dynamic Resource allocation using VM over Cloud Computing. Macaw International Journal of Advanced Research in Computer Science and Engineering, 1(1), 6-10.

[58] Venna Sujith Reddy, & K Venkatesh Sharma. (2023). Advancements in Automated Video Analysis Selective Scanning for Person of Interest Recognition . Macaw International Journal of Advanced Research in Computer Science and Engineering, 9(12), 1-8.

[59] Dr.J.Keziya Rani. (2016). Green Computing Paradigms towards Energy Conservation and E-Waste Minimization. Macaw International Journal of Advanced Research in Computer Science and Engineering, 2(9), 1-5