Journal of Biological Regulators and Homeostatic Agents 2026, 40(3), 116.
https://doi.org/10.65746/jbrha116

Article

Enhancing trust and efficiency in mobile edge computing with hybrid black widow-based updated jellyfish search optimization for task scheduling

Christina Ranjitham Manoharan*, Angela Jennifa Sujana Jesudoss

Department of Artificial Intelligence and Data Science, Mepco Schlenk Engineering College, Sivakasi, Tamil Nadu 626005, India
*** Corresponding author:** Christina Ranjitham Manoharan, christinaranjitham2@outlook.com

CITATION

Manoharan CR, Jesudoss AJS. Enhancing trust and efficiency in mobile edge computing with hybrid black widow-based updated jellyfish search optimization for task scheduling. 2026; 40(3): 116. https://doi.org/10.65746/jbrha116

ARTICLE INFO

Received: 18 April 2026
Revised: 23 June 2026
Accepted: 26 June 2026
Available online: 24 July 2026

Abstract: Task scheduling in Mobile Edge Computing (MEC) is a challenging multi-objective optimization problem, where conflicting goals such as minimizing latency, reducing energy consumption, and controlling execution cost must be achieved under uncertain and unreliable resource conditions. This study proposes a Federated Learning (FL)-based trust evaluation framework combined with a novel hybrid metaheuristic, the Black Widow-Updated Jellyfish Search (BW-UJS) algorithm. By integrating the exploration-exploitation mechanisms of Black Widow Optimization with the adaptive search behavior of Updated Jellyfish Search, the proposed method enhances decision-making in complex scheduling environments. The problem is formulated as a multi-objective optimization model that incorporates trust constraints to ensure reliable resource provider selection. Computational experiments conducted on synthetic MEC offloading scenarios demonstrate that BW-UJS consistently outperforms benchmark algorithms (Original, Offload, MUCAO, FLO), achieving up to 2.1 % improvement in energy efficiency, 3 % reduction in execution cost, and 0.01 % decrease in delay. The findings highlight the effectiveness of BW-UJS as a robust optimization approach for task scheduling in distributed systems. Future work will focus on extending the method with real-time adaptive learning mechanisms to address dynamic and large-scale network conditions.

Keywords: mobile edge computing (MEC); task scheduling, offloading strategy; black widow optimization (BW); updated jellyfish search (UJS); resource allocation; energy efficiency; federated learning (FL); trust evaluation

1. Introduction

By shifting computing workloads from centralized cloud servers to edge nodes, MEC improves network efficiency by relieving pressure on core networks and boosting system performance. Mission-critical applications like real-time analytics, smart grid management, and remote healthcare benefit greatly from this decentralized architecture's reduced latency and quicker reaction times. Within MEC, federated learning (FL) is a novel technique that preserves data privacy while enabling decentralized machine learning among dispersed edge devices. To reduce transmission costs and improve data security, edge devices work together to build a global model rather than sending raw data to a central server. Additionally, FL-based trust in MEC improves security, privacy, and dependability in edge networks by combining machine learning with trust management. Edge computing has a great deal of promise to improve MEC's energy and computational resource efficiency by utilizing job offloading technologies [1]. FL is being utilized more and more in ubiquitous computing, such as activity recognition, mobile applications, and remote healthcare, since it efficiently utilizes sensitive and decentralized data sources [2]. Through the

integration of Federated Expectation-Maximization with Deep Q Networks, FEDQ-Trust successfully addresses the statistical heterogeneity concerns [3]. The performance of the MEC system and scheduling costs can be efficiently balanced by the DORS algorithm. The DORS algorithm's ability to lower energy consumption is demonstrated by the comparison trials [4]. The DDQN-based approach can drastically cut energy consumption by 20 %, 35 %, and 53 % on average, and performs very closely to the exhaustion method [5]. Boost the edge network's transaction success rate by over two times when compared to a network without a trust assessment mechanism. It also outperforms the conventional subject logic (TSL) trust evaluation mechanism [6].

Contribution of this work

- Proposes a FL based trust evaluation scheme to enhance the reliability of MEC collaborative systems.
- Proposes a novel hybrid BW-UJS method to optimize job allocation in edge computing scenarios.
- Incorporates BW-UJS into the MEC framework to improve resource provider selection, reducing latency and energy consumption while maintaining service trustworthiness.
- Demonstrates BW-UJS's effectiveness through simulations, outperforming existing scheduling methods in terms of delay, cost, and energy efficiency.

The paper's establishment is described as follows: The relevant works are described in Section 2. The proposed approaches are explained in Section 3, the experiment's findings are shown in Section 4, and a conclusion and future work are provided in Section 5.

2. Related works

Cao et al. [7] have proposed, as society has grown more intelligent and people's wants have been continuously met, intelligence has permeated many sectors of the economy and people's day-to-day existence. Every facet of society is now impacted by edge devices, including cameras, intelligent production robots in intelligent manufacturing, smart homes, and driverless cars in the transportation sector. As an outcome, the overall number of Internet-connected gadgets has substantially expanded.

Kong et al. [8] have suggested, the way humans live, work, and learn has also been profoundly altered by edge computing, which offers robust support in application scenarios including digital advertising, remote working, and new physical retail industries. The definition, paradigm, and features of edge computing are presented, along with a number of important edge computing problems and innovative solutions backed by new technologies in the Internet of Everything era. Furthermore, they investigate possible and promising developments in the context of technological integration.

Abbas et al. [9] have recommended, an MEC facilitates the smooth integration of many vendors and application service providers for mobile users, businesses, and other vertical markets. It is a crucial component of the 5G architecture, supporting a wide range of novel applications and services that demand ultralow latency. This study aims

to provide a complete assessment of significant research and technology achievements in the field of MEC. A t last will discuss security and privacy concerns, as well as existing remedies.

Liang et al. [10] have discussed, a measuring MEC throughput, minimizing the migration cost, and optimizing the total offloading rate. A decision-optimization problem is used to develop the policy design, taking into consideration wireless multi-access, virtualization, and I/O interference between virtual machines (VMs). To handle the complicated combinatorial issue, we propose an effective relaxation-and-rounding-based solution strategy. The method is based on a new integer-recovery architecture and an effective iterative algorithm for solving the integer-relaxed issue.

Lin et al. [11] have reported, Fed-PEMC is an advanced federated deep learning method. This algorithm combines local differential privacy with model compression techniques. Fed-PEMC minimizes model size while retaining model correctness, resulting in improved communication efficiency. The experimental results demonstrate that Fed-PEMC outperforms the baseline solution DP-Fed by 2.27 and 2.02 percentage points in testing accuracy on the Mnist and Cifar10 datasets, respectively, and that Fed-PEMC is superior to baseline algorithms in terms of privacy, model accuracy, and communication efficiency.

Mazloomi et al. [12] have proposed, an innovative method of knowledge-sharing amongst FL servers that is built on trust, with the trust metric being the accuracy of shared global models on local data from clients. Implied methodology allows servers to use shared global models from trusted servers for their clients, boosting training accuracy and lowering loss. This improvement is especially noticeable during the initial training cycles compared to the original FL implementation.

Xiang et al. [13] have proposed, an adaptive collaborative federated learning (ACFL) technique is used to expedite convergence and increase model dependability by reducing communication-based parameter loss in a three-layer MEC architecture. First, a dynamic epoch adjustment approach is presented to reduce communication rounds by altering training epochs in end devices. Furthermore, proposed a multi-layer computing architecture-based edge server collaborative training technique to speed up the FL convergence. In this scheme, edge servers use their retained data to jointly train models with end devices. The results of comprehensive simulations demonstrate that ACFL can effectively increase model reliability and hasten the FL process's convergence in MEC.

Kuang et al. [14] have suggested, a partial offloading scheduling and power allocation (POSP) problem in single-user MEC systems is defined. The goal is to reduce the weighted total of execution delay and energy consumption while maintaining the transmission power constraint of the tasks. The execution delay of jobs executing on both MECs and mobile devices is examined. The energy consumption of both task computing and task data transfer is taken into consideration. Nonconvex mixed-integer optimization is the problem formulation. Numerical findings show that the suggested algorithms produce near-optimal delay performance with a significant reduction in energy usage.

Chen et al. [15] have described, to simultaneously find the best answers to these subproblems, an online technique called Task Offloading and Frequency Scaling for Energy Efficiency (TOFFEE) is put forward. This approach explores task allocation

and CPU-cycle frequency to achieve the lowest energy consumption while ensuring that the queue length is upper bounded. TOFFEE can achieve near-optimal energy utilization while limiting the application queue length. A performance evaluation is undertaken to confirm TOFFEE's effectiveness. The experiment findings show that TOFFEE can reduce energy usage by roughly 15 % compared to the RLE algorithm and 38 % compared to the RME method.

Mao et al. [16] have recommended, create an efficient compute offloading plan and integrate EH devices into a green MEC system. The execution cost, which includes both execution latency and task failure, is used as a performance metric. The Lyapunov optimization-based dynamic computation offloading technique is proposed as a low-complexity online algorithm that determines the offloading decision, CPU-cycle frequencies for mobile execution, and transmit power for computation offloading. This algorithm has the unique advantage of making judgments based solely on the status of the system at the moment, without requiring information about the distribution of the wireless channel, compute job request, and EH processes. Simulation results will be shown to confirm the theoretical analysis and validate its effectiveness.

Jiang et al. [17] have proposed, an online joint offloading and resource allocation (JORA) system under the long-term MEC energy limitation, with the goal of ensuring end-user quality of experience (QoE). Utilizing Lyapunov optimization, they take advantage of the long-term QoE maximization problem's optimality in order to do this. By creating an energy shortage queue to guide energy usage, a problem can be solved in real time. On this premise, they offer online JORA approaches in both centralized as well as scattered forms. The data demonstrate that this strategy outperforms others.

Wang et al. [18] have suggested, to solve the difficulties, they created a Trust-based Computation Offloading (TCO) architecture. They start by creating a mixed-integer nonlinear programming problem to minimize the difference between the cost of the CSC and its expected revenue (DCER). Second, they create a trust-based computation offloading method that rapidly discovers a good solution by breaking down the problem. Finally, in order to get precise trust values, a two-tiered trust evaluation method was suggested. TCO reduces the service timeout count in an interval by 34.37 % to 73.80 % with a performance loss of only 1.42 %–4.10 %.

Feng et al. [19] have reported, on average, a node selection and power control technique based on Benders' Decomposition (BD-NSPC) to get the global optimal solution effectively. Furthermore, simulation findings reveal that BD-NSPC consumes around 30 % less energy per EN on average and improves accuracy by 1–2 % when compared to existing SEL algorithms. Additionally, when compared to the federated learning (FL) architecture, BD-NSPC achieves comparable accuracy in the edge computing system while reducing energy consumption by roughly 25 % and latency by roughly 28 %.

Shiqiang et al. [20] have reported, use a Markov decision process (MDP) to formulate the service migration problem. In addition to providing a mathematical framework for designing optimal service migration plans, it also covers general cost models. The resulting MDP is exact for uniform 1-D user mobility, but it only delivers a close approximation for uniform 2-D mobility with a fixed additive error. Assessments derived from actual San Francisco taxi movement traces demonstrate the

suggested solution's better performance when compared to baseline solutions.

3. Proposed system

Figure 1 shows a framework for trust-aware task offloading in MEC that makes use of the FL-BE-UJS architecture. Three essential components make up the framework: Selection, Filtering, and Trust Evaluation. It begins with input data, which includes both identification and behavioral data. Data on behavior and identification move through several phases in the Trust Evaluation Module, including feature extraction, training, data collecting, and trust level prediction. This procedure uses the FL-BW-UJS model to evaluate MEC's performance and level of confidence. This module generates a behavior dataset, which is then processed by the Filtering Module. In order to maximize job allocation, this module refines the data via graph creation and nodes filtering. The filtered data is then assessed in the Selection Module, which examines important performance indicators such as energy usage, cost projection, and delay. Based on these evaluations, the framework chooses the best device or server to complete the task efficiently. This solution ensures a trust-based and performance-aware work offloading mechanism in the MEC contexts.

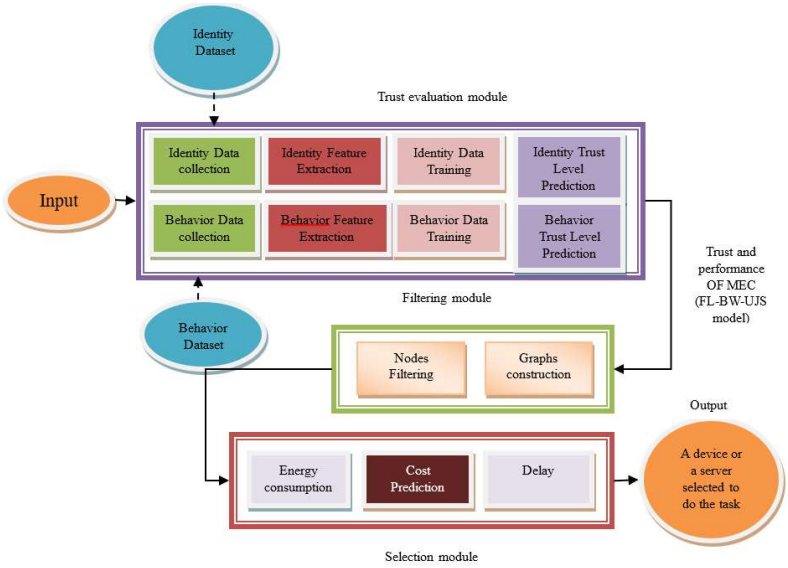

Figure 1. Trust-aware task offloading framework using FL-BE-UJS model in MEC.

3.1. FL based mobile edge computing

FL in MEC enables decentralized model training by using servers and edge devices to collaboratively train machine learning models without sharing raw data. This method simply exchanges model updates with a central aggregator, processing data locally at edge nodes to improve privacy and lower communication overhead [20]. FL optimizes the use of distributed computing resources in MEC contexts, reducing latency and improving efficiency in applications such as smart healthcare, autonomous vehicles, and IoT networks [21]. To guarantee reliable and effective learning, adaptive techniques like model compression, safe aggregation, and differential privacy are needed to overcome obstacles like device heterogeneity, communication limitations, and security risks like adversarial assaults.

3.2. Filtering

Filtering away unqualified nodes is the primary function of the filtering module. A supplier is deemed unqualified if the task sensitivity level exceeds their identity trust level [22]. A provider is considered unqualified if their behavior trust level is lower than the user's necessary behavior level. To simulate the social tie between the devices, we specifically establish the device identification trust graph G^{tru}, as illustrated in **Figure 2**.

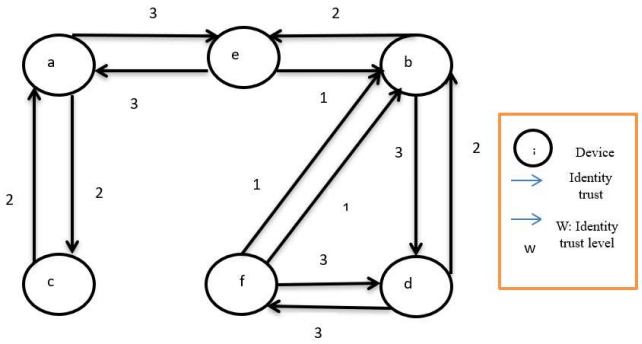

Figure 2. Device identity trust graph.

Figure 3 illustrates how devices' behavior trusts one another to create a D2D connection linked to the behavior trust graph. The weight of a directed edge between nodes i and j represents the level of behavior trust that i has in j.

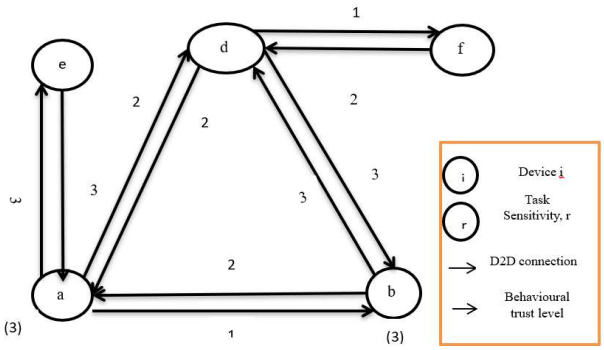

Figure 3. D2D relationship with the behavior trust graph.

3.3. BW-UJS model for optimal task scheduling in selection module

One is established as the lower bound (vmN), while virtual machine variant is set as the upper bound (vmN). **Figure 4** shows that the BW-UJS algorithm is based on the unusual mating habits of black widow spiders. The BW-UJS process begins with the initial population of spiders, each of which is waiting for a potential solution, much like earlier evolutionary approaches.

3.3.1. Initial population

The values of the problem variables must produce an appropriate design for the current problem solution in order to solve the optimization problem. The values of the problem variables are shown by each Black widow spider. It is best to think of this study's design as an array that circumvents the benchmark functions. A widow is a

$1 \times N_{var}$ array that is represented as $window = [x_1, x_2, \dots, x_{N_{var}}]_a$, a given solution to an optimization problem with N_{var} dimensions. Each $[x_1, x_2, \dots, x_{N_{var}}]$ represents a floating-point number. By assessing the fitness function f as in Equation (1) at a widow of any age, one can determine the widow's fitness of $[x_1, x_2, \dots, x_{N_{var}}]$.

$$fitness = f(x_1, x_2, \dots, x_{N_{var}}) \tag{1}$$

Create the candidate widow matrix using $N_{pop} \times N_{var}(size)$ and the spider's initial population prior to initializing the optimization model. Finally, parent couples are chosen at random to begin the reproductive process by mating, and the female Black widow consumes the male during or immediately following mating.

3.3.2. Proposed procreate

Due of their independence from one another, the parent pairs begin mating in order to create a new, parallel generation. Now, the following Equation (2) is used in this operation. Alpha is an array that must also be created for reproduction, along with a widow array that includes random values, where x_1, x_2 stand for parents and y_1, y_2 for offspring.

$$\begin{cases} y_1 = \alpha \times x_1 + (1-\alpha) \times x_2 \\ y_2 = \alpha \times x_2 + (1-\alpha) \times x_1 \end{cases} \tag{2}$$

The proposed logic states that the traditional procreate process equation and the JSO position update equation Equation (4) are combined to complete the procreate process update equation in Equation (3). The distribution coefficient is represented by β, the mean position of the jellyfish by μ, and the optimal jellyfish location at the moment is represented by x^*.

According to the suggested reasoning, the procreate process update equation is completed in Equation (3) by combining the traditional procreate process equation with the JSO position update equation Equation (4). β is the distribution coefficient, μ is the jellyfish mean position, and x^* is the best jellyfish location at the moment.

$$x_i(t+1) = x_i(t) + rand(0,1) \times (x^* - \beta \times rand(0,1) \times \mu) \tag{3}$$

$$\begin{cases} y_1 = \dfrac{\alpha x_1 + (1-\alpha)x_2 + x + rand(x^* - \beta \times rand \times \mu)}{2} \\ y_2 = \dfrac{\alpha x_2 + (1-\alpha)x_1 + x + rand(x^* - \beta \times rand \times \mu)}{2} \end{cases} \tag{4}$$

Duplication of the randomly selected integers should be avoided by repeating this procedure $N_{var}/2$ times. These are the procedures that each pair must follow: the father and children are grouped together and ranked according to their degree of fitness, which is now established by the cannibalism rating. A select handful of the fittest people are then included to the newly formed population. The Gaussian mutation in Equation (5) improves the proposed method and leads to an improvement in local search capabilities such as search range and direction.

$$Gauss(x) = \frac{1}{\sqrt{2\pi\sigma^2}} exp^{-\left(\frac{(x-\alpha)^2}{2\sigma^2}\right)} \tag{5}$$

$$Here, \alpha = Mean\left(x(t)\right), \sigma = SD(x(t)) \tag{6}$$

3.3.3. Arithmetic crossover

According to the proposed theory, arithmetic crossover is also used to get the solutions in Equation (7), which merges the two parent chromosomes linearly, with δ representing the offspring, γ_1, γ_2 representing parent 1, and $H \in (0; 1)$ representing the random value.

$$\delta = H * \gamma_1 + (1 - H) * \gamma_2 \tag{7}$$

Figure 4 illustrates the steps of the BW-UJS algorithm. It begins with the initialization of a population, denoted as "pop", of potential solutions. The algorithm then enters a loop that continues until a specified termination condition is met. During each iteration, the reproduction count (nr) is calculated based on the processing rate, and the best "nr" solutions are selected and stored in a sub-population called *pop1*. A conditional check determines whether i = 1 to nr holds true. If not, the algorithm calculates the number of mutations based on the mutation rate. If true, two solutions are randomly chosen from another sub-population, *pop2*, to serve as parents for generating offspring. The count of mutation children (nm) is also calculated according to the mutation rate. Following this, a nested loop checks if j = 1 to nm. If not, the process proceeds to update the overall population by merging *pop1* and *pop2*. If the condition is met, solutions are selected from pop1 for mutation. Eventually, the best solution is identified and returned. Once the termination condition is satisfied, the algorithm concludes by returning the best solution and stopping execution. Overall, this flowchart outlines a genetic or evolutionary algorithm framework with specific mechanisms for selection, mutation, and population updating, aimed at solving an optimization problem effectively.

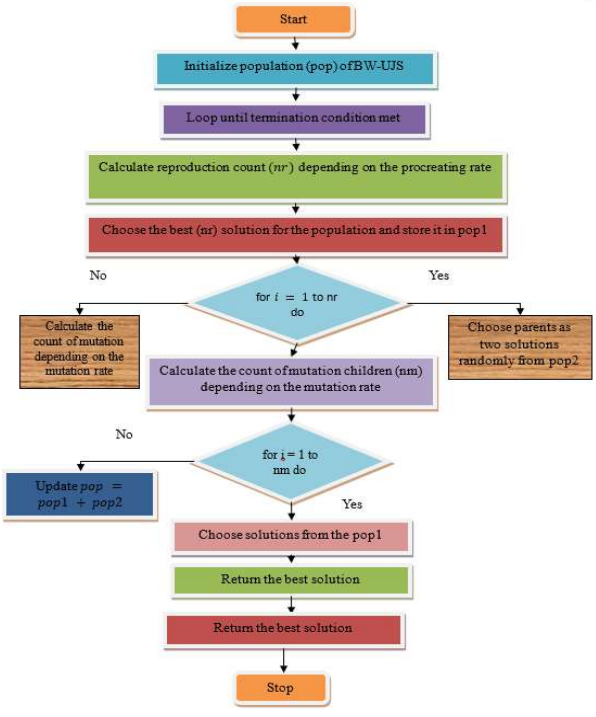

Figure 4. Flowchart for BW-UJS algorithm.

3.3.4. Cannibalism

Three types of cannibalism can be distinguished in this case. The first is sexual cannibalism, in which a black widow woman consumes her husband either during or after mating. Using this technique, men and women can be identified according to their fitness levels. When stronger spiderlings consume their weaker siblings, this is another form of cannibalism. A cannibalism rating is incorporated into this algorithm, which affects the number of survivors that are found. It has occasionally been observed that young spiders are engaging in the third type of cannibalism, where they consume their mother. The fitness rating evaluates spiderling strength and frailty. Algorithm 1 shows the pseudo code for the proposed BW-UJS algorithm.

Algorithm 1 Pseudo code of proposed BW-UJS algorithm using edge computing

Input: virtual machine (vm_N), procreating rate, cannibalism rate and mutation rate

Output: Optimal solution for the objective function

//initialization stage

The initial population of BWUJS

Every pop (population) is a d-dimensional chromosome array for the d-dimensional problem

//loop until termination condition met

Calculate reproduction count (*nr*) depending on the procreating rate

Choose the best (*nr*) solution for the population and store it in pop_1

for $i = 1$ **to** nr **do**

Choose parents as two solution randomly from pop_2

end for

//Mutation

Calculate the count of mutation children (nm) depending on the mutation rate

for i = 1 **to** *nm* **do**

Choose solutions from the *pop1*

$x = x \times (1 - Gauss(x))$

xnew = Arithmetic_crossover (x)

Store new solutions in pop_3

end for

Update $pop = pop_2 + pop_3$

Return the best solution

Return the best solution from the pop

3.4. Performance evaluation

When assessing our proposed approach and comparing it with other offloading methods, metrics such as energy usage, total execution cost, total network use, and delay should be considered.

3.4.1. Energy consumption

Equation (8) determines the energy usage for each edge server and cloud after the input modules have been serviced.

$$E = E_c + (T_n - T_{lu}) * P_h \tag{8}$$

Determine how much energy the edge server uses by dividing the total power of all hosts during a specific execution period. In this case, P_h represents the host's power in the most recent utilization, T_n the current time, T_{lu} the update time at the most recent use, and E_c the energy consumption in the current state. The energy used by the cloud and all edge servers must be added up to determine the overall energy usage.

3.4.2. Total execution cost

To calculate the execution cost, divide the total MIPS of hosts by the time frame, which is obtained by subtracting the latest usage time from the simulation's current time.

$$Cost = \sum_{i=1}^{N} [C + (SC - T_{lu}) * R_M * U_l \sum_{k=1}^{\omega} MIPS_k] \tag{9}$$

In Equation (9), N represents the number of edge servers, C represents the execution cost, SC represents the system clock or current simulation time, T_{lu} represents the time since the last usage, R_M represents the rate per MIPS that changes for each inter-module edge, and TM represents the host's total $MIPS$. The most recent usage (U_l) is calculated using Equation (10). The allotted MIPS and MIPS of the *k-th* processor in the host are represented by $MIPS_k^A$ and $MIPS_K$, respectively, where the number of all processors and allocated processors in a host is m1 and m2.

$$U_l = Min(1, \frac{\sum_{k=2}^{m2} MIPS_k^A}{\sum_{k=1}^{m1} MIPS_k}) \tag{10}$$

3.4.3. Total network usage

Because tuples specify the connections between modules, the amount of the transferred tuples at a given moment determines how much resource is used. Network use as a whole is based on Equation (11).

$$TNU = \frac{\sum_{i=1}^{N'} (L_i * S_i)}{T_{max}} \tag{11}$$

Where, L_i and S_i are the overall latency and size of the i-th tuple, T_{max} is the maximum simulation period, and N' is the total number of tuples.

3.4.4. Application delay

The system clock and a tuple's end time are used to determine the application execution delay.

$$T_{TN} = \begin{cases} SC - T_{st}, & if\ T_a = 0, \\ \dfrac{T_{st} * N_{ET} + (SC - T_{st})}{N_{ET} + 1} & if\ T_a \geq 0 \end{cases} \tag{12}$$

Equation (12) is used to determine the tuple's end time. Where *SC* stands for system clock, T_{st} for tuple start time, N_{ET} for the number of executed tuple types, and $(SC - T_{st})$ for execution time. The average CPU time according to the tuple type is

denoted by T_a. A tuple's emission time is indicated by *ET*, and the system clock is represented by *CC*. The time it takes for two modules to transfer is T_s.

$$T_{TN} = \frac{T_{st} * N_{RT} + (SC - T_s)}{N_{ET} + 1} \tag{13}$$

Equation (13) is the basis for the tuple receipt time. N_{RT} denotes the number of different receipt tuple kinds. The time difference between a module's tuple finish time and another module's tuple reception time is used to compute application delay.

4. Experimental result and discussion

Edge CloudSim was extended and adjusted to gather the necessary data for the behavior trust evaluation's data collection phase. Three specialists who have researched mobile edge computing were asked to classify the training data in order to guarantee its quality. A total of 1920 recordings were labeled, with 1728 serving as training data and the remaining 192 as test data. Behavior trust was divided into three degrees. The elements of the behavior trust feature vector $v(i;j)$ are listed in **Table 1**.

Table 1. Elements of behavior trust feature vector.

Name	Interpretation
$ANum\ (j)$	The number of times that provides services
$CNum\ (j)$	The number of times that j successfully completed these tasks
$ACRatio\ (j)$	The ratio of $CNum\ (j)$ and $ANum\ (j)$
$Reputation(j)$	Average score of j given by objects interacting with j
$TD\_rate\ (j)$	Time division rate of j's services
$Num(j)$	The number of times that j provides services for i
$Score(i,j)$	The average score of j given by i

Table 2 and **Figure 5** show the results of the three previously indicated measures for each level of behavior trust evaluation. They evaluate the trained trust classifiers' performance and usefulness. After training, some data was added, and experiments were carried out with different test data sizes to verify the classifiers' accuracy.

The comparison result of the behavior trust evaluation is shown in **Figure 6**. BTC is an identity trust classifier. Based on their experience, the experts analyzed the global trust value and distinguished between [0, 1] into three intervals: [0, 0.35], [0.35, 0.6], and [0.6, 1] in order to compare behavior classifier with MSTrust. Behavioral trust levels 1, 2, and 3 are represented by these three intervals, accordingly. The image shows how a classifier can achieve more accuracy. The MSTrust technique only considers three trust factors: General reputation, direct feedback, and success rate [23].

Table 2. Measurement of behavior trust classifier.

	P	R	F
Level 1 (poor)	0.9393	0.9117	0.9252
Level 2 (acceptable)	0.94	0.94	0.8836
Level 3 (Strongly recommended)	0.8461	0.9166	0.8799

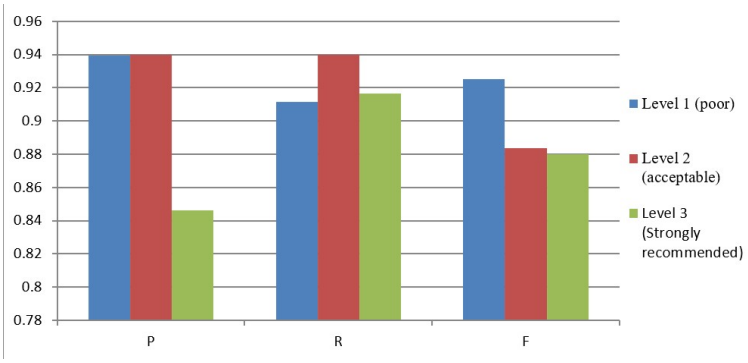

Figure 5. A flowchart of the behavior trust classifier.

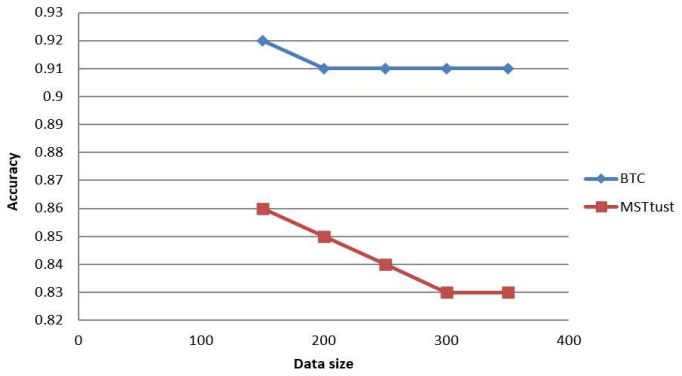

Figure 6. Variations in behavior trust accuracy.

4.1. Energy consumption based on the number of users

According to the examination of **Figure 7**, MUCAO uses less energy than the original and offload approaches.

A greater number of users can use less energy thanks to MUCAO. This figure indicates that the maximum energy usage for the initial method was $1.635 \times 107\ MJ$ multiplied by 7 users. The BW-UJS technique requires a minimum value of $1.58 \times 107\ MJ$ for every ten users. This outcome demonstrates that the distributed structure FL-based approach uses less energy than the others. Additionally, employing multiple EDs and incorporating context-awareness data into FLO result in a more effective energy efficiency technique than BW-UJS.

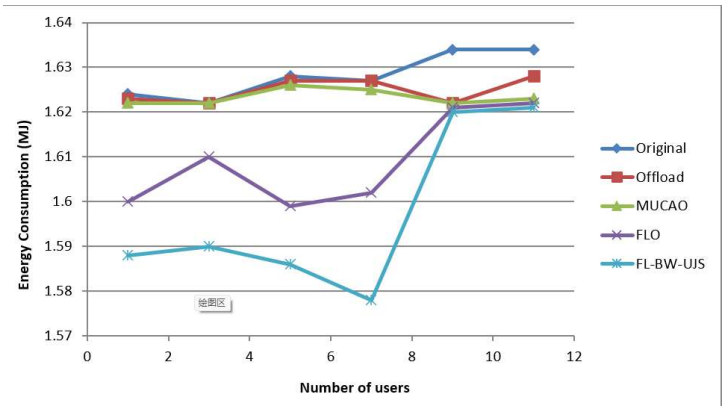

Figure 7. The quantity of users determines the energy consumption.

4.2. Total execution cost based on the number of users

The cost is an important parameter in this study. **Figure 8** illustrates how raising the number of users from three to seven or ten results in higher costs for both original and offloading approaches. In comparison to these approaches, MUCAO offers a balanced cost for many consumers, ranging from $4.12 \times 106\,\$\ to\ 4.15 \times 106\,\$$. Additionally, because BW-UJS uses less energy than FLO and MUCAO, it has the lowest overall execution costs. This result demonstrates how the BW-UJS has effectively managed costs and brought about economic savings despite the environment's increased user base and distribution. Its distributed algorithm in some EDs, which chooses the best device with the lowest latency and the best performance, is the main cause of this gain.

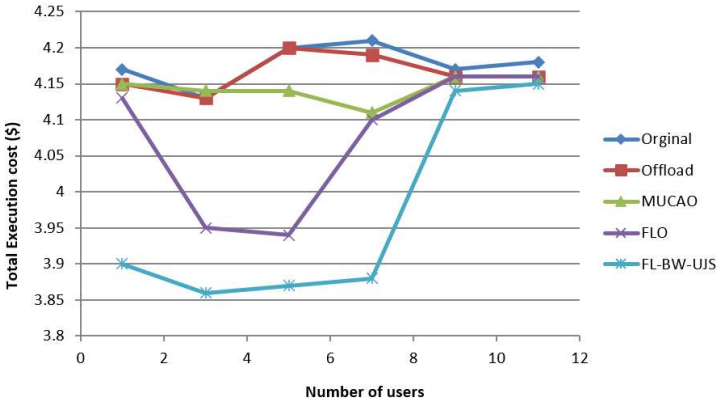

Figure 8. Total cost of execution according to user count.

4.3. Delay based on the number of users

Except for users 3, the MUCAO application loop delay has somewhat decreased. Compared to the original and offload approaches, these findings are superior. **Figure 9** illustrates that the maximum delay is seven times the number of users compared to the original method. The number of users is ten times greater than the smallest figure, which is correlated with BW-UJS. This demonstrated how the usage of distributed algorithms and context information speeds up the offloading and request execution processes. This implies that MDs can save more time by using BW-UJS to swiftly identify the ideal location to offload their modules.

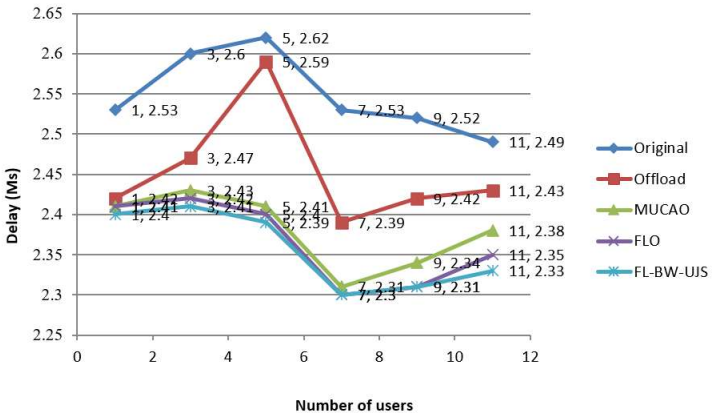

Figure 9. The number of users determines the delay.

4.4. Energy consumption based on mobile types

In every state (AB, AC, BC, and ABC), BW-UJS outperforms the others according to mobile type analysis. **Figure 10** results demonstrate that the suggested strategy can reduce energy usage due to the variety of mobile devices.

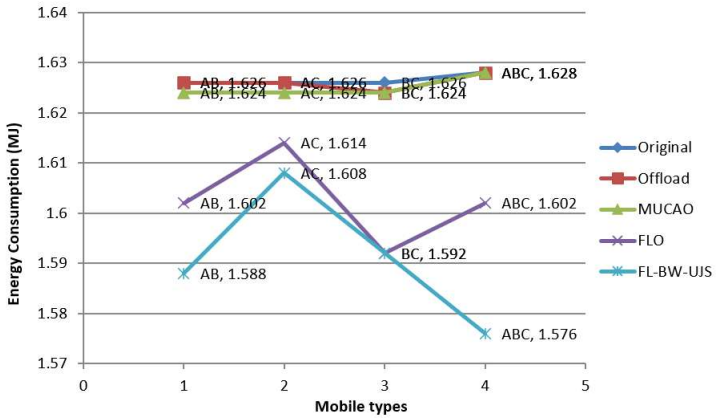

Figure 10. Energy usage according to the types of mobile.

4.5. Total execution cost based on mobile types

As seen in **Figure 11**, employing various mobile types lowers overall costs across the board. BW-UJS produces superior results than the others. More modules are executed locally by mobile devices with larger CPU, memory, and battery capacities. This method lowers the offloading cost. The ability to unload fairly across a variety of devices is another justification for enhancing the BW-UJS. As a result, lowering a device's price and providing equitable offloading to other d devices can result in cheaper costs.

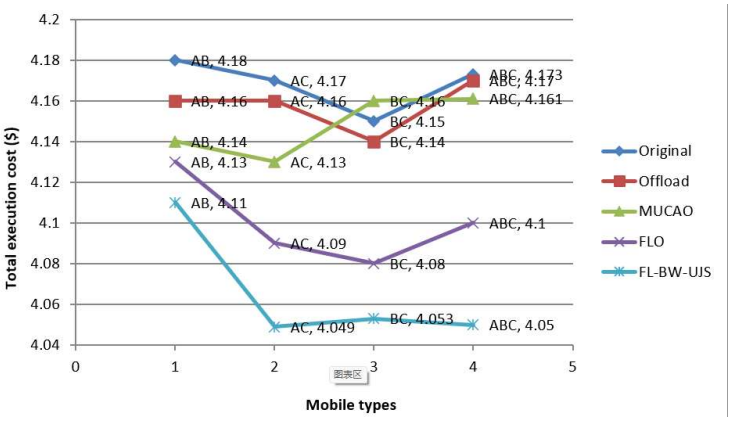

Figure 11. Total cost of execution according to module size.

4.6. Delay based on module types

Figure 12 illustrates the sensitive changes in delay in all mobile types, including AB, AC, and BC. On mobile type ABC, BW-UJS has a minimum latency of 224.5mS. The results show that delays are increased when MDs are more diversified. Distributed algorithms like FLO and BW-UJS are capable of outperforming others. BW-UJS has a shorter latency when the mobile type is AC. The fact that BW-UJS has greater context awareness than FLO may be the cause of this improvement.

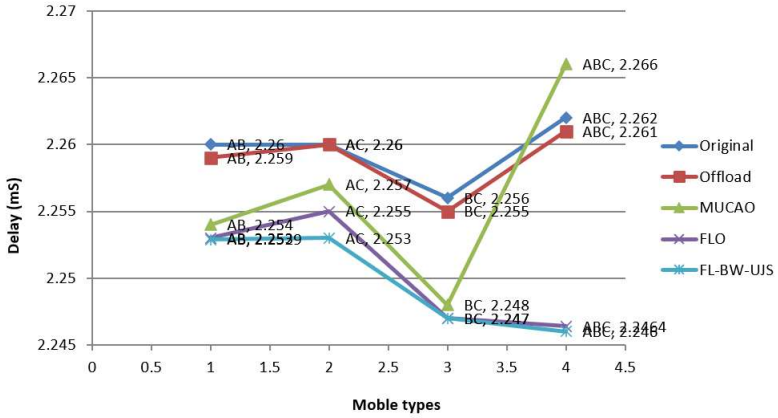

Figure 12. Delay according to module types.

The comparison of performance analysis between the proposed and current methods is shown in **Table 3**.

Table 3. A comparison of proposed algorithm Vs Existing algorithm.

Reference	Utilized technique	Performance metric	Evaluation tools
[24]	Particle Swarm optimization	Energy Consumption	Shanghai Telecom
[25]	GTGP algorithm	Delay, Cost	MATLAB
[26]	AR applications	Energy, execution Delay	CIFAR-10
Proposed	FL based BW-UJS algorithm	Trust, Execution cost, execution delay and energy	Edge CloudSim

5. Conclusion

This study introduces a trust evaluation scheme based on FL to address the issues of unreliable resource providers in MEC. The proposed hybrid BW-UJS optimization method improves job allocation by balancing important objectives such task completion time, cost, and resource use. By including BW-UJS into the MEC framework, the study enables effective resource provider selection, which reduces latency and energy consumption while ensuring service trustworthiness. BW-UJS outperforms previous scheduling algorithms, making it a reliable and efficient option for MEC-based task scheduling. The simulation results show that BW-UJS outperforms current scheduling approaches in terms of original, offload, MUCAO, and FLO methods in terms of energy consumption by 2 %, 2 %, 2.1 %, and 0.7 %, total execution cost by 3 %, 3 %, 2.34 %, and 1.08 %, and delay by 0.01 %, 0.01 %, 0.005 %, and 0.001 % and 0.002 %, respectively. Future research can focus on extending the BW-UJS method by incorporating real-time adaptive learning mechanisms to dynamically adjust scheduling decisions based on evolving network conditions and user demands in MEC environments.

Author contributions: Conceptualization, CRM and AJSJ; methodology, CRM; software, AJSJ; validation, CRM, and AJSJ; formal analysis, CRM; investigation, AJSJ; resources, CRM; data curation, AJSJ; writing—original draft preparation, AJSJ;

writing—review and editing, CRM; visualization, AJSJ; supervision, AJSJ; project administration, CRM; funding acquisition, AJSJ. All authors have read and agreed to the published version of the manuscript.

Funding: None.

Ethical approval: Not applicable.

Informed consent statement: Not applicable.

Acknowledgements: The author would like to express his heartfelt gratitude to the supervisor for his guidance and unwavering support during this research for his guidance and support.

Conflict of interest: The authors declare no conflict of interest.

References

1. Xie B, Cui H. Deep reinforcement learning-based dynamical task offloading for mobile edge computing: B. Xie, H. Cui. The Journal of Supercomputing. 2025; 81(1): 35. doi: 10.1007/s11227-024-06603-x
2. Djebrouni Y, Benarba N, Touat O, et al. Bias mitigation in federated learning for edge computing. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies. 2024; 7(4): 1–35. doi: 10.1145/3631455
3. Bai J, Dong H, Bouguettaya A. Fedq-trust: Efficient data-driven trust prediction for mobile edge-based iot systems. ArXiv Preprint ArXiv:2404.18356, 2024. doi: 10.48550/arXiv.2404.18356
4. Zhao F, Ying C, Zhang Y, et al. "Dynamic offloading and resource scheduling for mobile-edge computing with energy harvesting devices." IEEE Transactions on Network and Service Management. 2021; 18(2): 2154–2165. doi: 10.1109/TNSM.2021.3069993
5. Zhou H, Jiang K, Liu X, et al. Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing. IEEE Internet of Things Journal. 2021; 9(2): 1517–1530. doi: 10.1109/JIOT.2021.3091142
6. Deng X, Liu J, Wang L, et al. A trust evaluation system based on reputation data in mobile edge computing network. Peer-to-Peer Networking and Applications. 2020; 13(5): 1744–1755. doi: 10.1007/s12083-020-00889-3
7. Cao K, Liu Y, Meng G, et al. An overview on edge computing research. IEEE Access. 2020; 8: 85714–85728. doi: 10.1109/ACCESS.2020.2991734
8. Kong X, Wu Y, Wang H, et al. Edge computing for internet of everything: A survey. IEEE Internet of Things Journal. 2022; 9(23): 23472–23485.
9. Abbas N, Zhang Y, Taherkordi A, et al. Mobile edge computing: A survey. IEEE Internet of Things Journal. 2017; 5(1): 450–465. doi: 10.1109/JIOT.2017.2750180
10. Liang Z, Liu Y, Lok T M, et al. Multi-cell mobile edge computing: Joint service migration and resource allocation. IEEE Transactions on Wireless Communications. 2021; 20(9): 5898–5912. doi: 10.1109/TWC.2021.3070974
11. Lin Q, Jiang S, Zhen Z, et al. Fed-PEMC: A privacy-enhanced federated deep learning algorithm for consumer electronics in mobile edge computing. IEEE Transactions on Consumer Electronics. 2024; 70(1): 4073–4086. doi: 10.1109/TCE.2024.3351648
12. Mazloomi F, Shah Heydari S, El-Khatib K. Trust-based knowledge sharing among federated learning servers in vehicular edge computing. Proceedings of the Int'l ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications; 2023. pp. 9–15. doi: 10.1145/3616392.3624701
13. Xiang T, Bi Y, Chen X, et al. Federated learning with dynamic epoch adjustment and collaborative training in mobile edge computing. IEEE Transactions on Mobile Computing. 2023; 23(5): 4092–4106. doi: 10.1109/TMC.2023.3288392
14. Kuang Z, Li L, Gao J, et al. Partial offloading scheduling and power allocation for mobile edge computing systems. IEEE Internet of Things Journal. 2019; 6(4): 6774–6785. doi: 10.1109/JIOT.2019.2911455
15. Chen Y, Zhang N, Zhang Y, et al. TOFFEE: Task offloading and frequency scaling for energy efficiency of mobile devices in mobile edge computing. IEEE Transactions on Cloud Computing. 2019; 9(4): 1634–1644. doi: 10.1109/TCC.2019.2923692

16. Mao Y, Zhang J, Letaief K B. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. IEEE Journal on Selected Areas in Communications. 2016; 34(12): 3590–3605. doi: 10.1109/JSAC.2016.2611964
17. Jiang H, Dai X, Xiao Z, et al. Joint task offloading and resource allocation for energy-constrained mobile edge computing. IEEE Transactions on Mobile Computing. 2022; 22(7): 4000–4015. doi: 10.1109/TMC.2022.3150432
18. Wang J, Li Z, Liu H, et al. A trust-based computation offloading framework in mobile cloud-edge computing networks. IEEE Transactions on Mobile Computing. 2025; 24(6): 5370–5385. doi: 10.1109/TMC.2025.3530480
19. Feng L, Liao C, Shi Y, et al. Explainable and energy-efficient selective ensemble learning in mobile edge computing systems. IEEE Transactions on Network and Service Management. 2025. doi: 10.1109/TNSM.2025.3539830
20. Wang S, Urgaonkar R, Zafer M, et al. Dynamic service migration in mobile edge computing based on Markov decision process. IEEE/ACM Transactions on Networking. 2019; 27(3): 1272–1288. doi: 10.1109/TNET.2019.2916577
21. Ahmed E, Rehmani M H. Mobile edge computing: opportunities, solutions, and challenges. Future Generation Computer Systems. 2017; 70: 59–63. doi: 10.1016/j.future.2016.09.015
22. Ye Y, Li S, Liu F, et al. EdgeFed: Optimized federated learning based on edge computing. IEEe Access. 2020; 8: 209191–209198. doi: 10.1109/ACCESS.2020.3038287
23. Lakra A K, Suhag D. CrisisConnect: A blockchain-driven edge computing approach in mobile crowdsensing system. Data Science & Exploration in Artificial Intelligence. CRC Press; 2025: 297–304. doi: 10.1201/9781003589273-44
24. Li Y, Wang S. An energy-aware edge server placement algorithm in mobile edge computing. 2018 IEEE International conference on edge computing (EDGE). IEEE; 2018. pp. 66–73. doi: 10.1109/EDGE.2018.00016
25. Naouri A, Wu H, Nouri N A, et al. A novel framework for mobile-edge computing by optimizing task offloading. IEEE Internet of Things Journal. 2021; 8(16): 13065–13076. doi: 10.1109/JIOT.2021.3064225
26. Chen D, Xie L J, Kim B G, et al. Federated learning based mobile edge computing for augmented reality applications. 2020 international conference on computing, networking and communications (ICNC). IEEE; 2020. pp. 767–773. doi: 10.1109/ICNC47757.2020.9049708