Enhanced Whale Optimization Algorithm for Dependent Tasks Offloading Problem in Multi-edge Cloud Computing

Atluri Charitha Sri, Bandapalli Venkata Durga Sahithi, K Triveni
B.Tech, Computer Science and Engineering,
National Institute of Technology, Calicut.

***Abstract*—We are implementing the Enhanced Whale Optimization Algorithm (EWA) to optimize dependent task offloading in a multi-edge cloud computing environment. The algorithm aims to identify the most suitable offloading environment for dependent tasks, focusing on minimizing total processing latency, energy consumption, and associated costs. We are implementing this algorithm in a system having many decentralized Mobile Edge Computing servers (MECs) and a centralized cloud server. The two improvement operations Frame Shifting (FS) and Load Redistribution Strategy (LRS), are introduced to enhance the performance of the whale algorithm. Through simulation, our results present us with the superior performance of EWA over WOA. When compared to the Whale Optimization Algorithm (WOA), EWA achieves a reduction in latency by 22.84%, a decrease in energy consumption by 78.28%, and a reduction in cost usage by 61.47%. These outcomes tell us the efficiency and the significance of the proposed EWA in overcoming the challenges posed by dependent task offloading in the multi-edge cloud computing environment.**

***Index Terms*—Computation offloading, Task dependency, Mobile edge computing, Multi-edge cloud computing, Multi-objective optimization, Enhanced Whale Optimization, Dynamic allocation**

I. Introduction

Nowadays, people are using the services of advanced wireless applications in various fields like Computer Vision, Natural Language Processing and many more which are demanding the need of computing resources [1]. Many automotive tasks require many computational resources and have tight deadlines for the completion, for example, autonomous driving. Vehicles have onboard computers (OBCs), but they face the problem of high hardware cost of these devices and insufficient resources [2]. Smartphones and IoT devices are lightweight devices so they have low processing capabilities and also they work temporary battery for power [3].

In these cases, Cloud Computing helps vehicles and IoT gadgets to offload their processing tasks to remote cloud servers and addresses, solves their limitations. As the latency resulting from long-distance task transfers is insufficient to fulfill the low latency needs of real-time applications, we deploy Mobile Edge Computing (MEC) at the edge of the network as the solution [4]. With the MEC in the network, the lightweight IoT devices can offload their tasks to be completed as quickly as possible. Depending just on MEC servers might not be able to meet the processing demands of intensive tasks, though it reduces latency. So we also have Cloud Server in our project system. Now in this system, Optimization is also an important parameter. Some of the key optimization objectives are energy conservation (E), cost minimization for MEC and cloud usage (UC), and reduction of latency (L) [5].

In order to perform effective task execution on a cloud network aided by MEC servers, we have to make decisions about task offloading. For this reason, Numerous strategies, including Lyapunov optimization, heuristics, metaheuristics, game-theoretic, convex optimization, mathematical, machine learning, supervised deep learning (SDL), Deep Reinforcement Learning (DRL), and various other algorithms, have been proposed to offer a stable solution to this problem. However, these solutions serve a singular purpose, have challenges in application, are expensive, focus solely on individual user considerations, are either cloud or MEC-based, or overlook the interdependence among application tasks.

In this project, we are implementing an enhanced version of the Whale Optimization Algorithm (WOA) [6] to address the dependent tasks offloading problem within the multi-edge cloud computing environ ment, targeting three main objectives (L, UC, E). The cooperative hunting behavior of humpback whales inspired the WOA. The standard WOA is improved by adding two main operations. The first operation is a FrameShifting (FS) operation, which is inspired by the replication behavior of the COVID-19 virus [7]. The second enhancement operation is the Load Redistribution Strategy (LRS), which helps us to maintain a fair load balancing among the included processing locations. In this project, we have a system composed of M MECs and a cloud server for processing the offloaded tasks, taking the tasks' interdependency into account.

II. Related Work

The goal is to improve service quality by minimizing latency, reducing energy consumption, and costs. Many efforts in the literature have been made to offer solutions to the computational offloading issue, employing various system models and formulated problems. Some of the solutions include employing two MECs alternately for offloading Mobile tasks to reduce latency and battery usage, keeping a copy of the file after the first time processing, using game-theory methods, Stackelberg game, Stochastic game for addressing multi-user computational offloading challenges within the MEC system (aiming to optimize execution time and energy consumption during offloading decisions), Heuristic and Metaheuristic algorithms, the multi-object evolutionary algorithm to minimize energy consumption and latency.

Many of them used the Whale Optimization Algorithm (WOA) to develop a task scheduling approach that effectively reduced both completion time and energy consumption. Many have employed SDL and DRL techniques, and gave solutions to address task offloading challenges. We are trying to replicate a fresh and original solution to the challenge of dependent task offloading within the MEC environment. Existing approaches may encounter one or more of the following challenges:

• Disregard a few of the three goals under consideration.

• Ignores the interdependence of tasks.

• For their offloading, rely either on the cloud server or the edge, but not both.

• Depends on a single MEC server.

• Ignore the variations in computing capabilities of the adopted devices.

We provide an approach to address the dependent task off loading problem in a multi-edge-cloud computing architecture by reducing three primary goals: use cost, energy consumption, and execution latency. We use an enhanced version of the WOA technique, which is used on this problem.

III. System Model and Problem Formulation

The system has M MEC servers and a single centralized cloud server. Each task can be executed locally on the smartphone, offloaded to one of the nearby MEC servers, or offloaded to the central cloud. IoT devices sense and preprocess data and forward it to a smartphone, which is the central point. It is divided into a set of T tasks:

$$T = \{T_0, T_1, \dots, T_N\},$$

where N denotes the total number of tasks.

Each task T_i is represented as a node in a directed acyclic graph (DAG), and edges represent dependencies among tasks. An edge E_{ij} indicates that task T_i must finish before task T_j begins.

Each task T_i is characterized by:

- CC_i: required CPU cycles (MIPS),
- Q_i^{in}: input data size (MB),
- Q_i^{out}: output data size (MB).

A. Local Computation

When a task T_i is processed on the local device, its execution time is

$$ET_i^{\text{local}} = \frac{CC_i}{F_{\text{local}}},$$

where F_{local} is the local CPU frequency.

Since no network transmission, the total local completion latency is

$$L_i^{\text{local}} = ET_i^{\text{local}} + WT_i^{\text{local}} + QT_{\text{local}},$$

where QT_{local} is the queuing time before scheduling, and WT_i^{local} is the waiting time due to dependencies.

The waiting time (due to predecessors) is computed as

$$WT_i^{\text{local}} = \max\Big\{ \max\left(L_j^{\text{local}},\ L_j^{\text{MEC}},\ L_j^{\text{cloud}}\right) \mid T_j \in P(T_i) \Big\} - QT_{\text{local}}.$$

If the computed value is negative or zero, then ignore.

Local energy consumption is:

$$E_i^{\text{local}} = C \cdot CC_i \cdot (F_{\text{local}})^2,$$

where C is the capacitance coefficient of the processor.

The cost of local execution is zero:

$$UC_i^{\text{local}} = 0.$$

B. Offloading to the MEC Server

If task T_i is offloaded to MEC server k (for $k = 1, \dots, M$), the execution time at MEC k is

$$ET_i^{\text{MEC}_k} = \frac{CC_i}{F_{\text{MEC}_k}},$$

where F_{MEC_k} is the CPU frequency of MEC k.

Input transmission and output reception times are:

$$T_{k,i}^{\text{trans}} = \frac{Q_i^{\text{in}}}{R_k}, \qquad T_{k,i}^{\text{rec}} = \frac{Q_i^{\text{out}}}{R_k},$$

where R_k denotes the achievable uplink/downlink rate between the smartphone and MEC k.

Using Shannon–Hartley, the rate R_k is

$$R_k = W \log_2\left(1 + \frac{WP \cdot G_k}{\sigma^2}\right),$$

with W the channel bandwidth, WP the transmit power, G_k the channel gain, and σ^2 the noise power. (Propagation delay is negligible for short links.)

The total latency for MEC execution is

$$L_i^{\text{MEC}_k} = ET_i^{\text{MEC}_k} + T_{k,i}^{\text{trans}} + T_{k,i}^{\text{rec}} + WT_i^{\text{MEC}_k},$$

where $WT_i^{\text{MEC}_k}$ is the waiting time on MEC k computed similarly to the local waiting time but using MEC/cloud completion latencies of predecessors.

Energy consumed by the smartphone during transmission and reception is

$$TE_i^{\text{MEC}_k} = WP \cdot T_{k,i}^{\text{trans}}, \qquad RE_i^{\text{MEC}_k} = WP \cdot T_{k,i}^{\text{rec}}.$$

The total smartphone energy for MEC offloading is

$$E_i^{\text{MEC}_k} = TE_i^{\text{MEC}_k} + RE_i^{\text{MEC}_k}.$$

The execution cost for using MEC k is

$$UC_i^{\text{MEC}_k} = \beta \cdot E_i^{\text{MEC}_k},$$

where β is the cost-per-energy coefficient.

C. Offloading to the Cloud Server

If task T_i is offloaded to the centralized cloud, the cloud execution time is

$$ET_i^{\text{cloud}} = \frac{CC_i}{F_{\text{cloud}}},$$

where F_{cloud} denotes the CPU frequency allocated in the cloud.

Transmission and reception delays to/from the cloud include propagation:

$$T_i^{\text{trans,cloud}} = \frac{Q_i^{\text{in}}}{R_k} + T_{\text{prop}}, \qquad T_i^{\text{rec,cloud}} = \frac{Q_i^{\text{out}}}{R_k} + T_{\text{prop}},$$

where T_{prop} is the propagation delay along the backhaul, typically modeled as

$$T_{\text{prop}} = \frac{L}{S},$$

with L the backhaul distance and S the signal propagation speed.

Total cloud latency is

$$L_i^{\text{cloud}} = ET_i^{\text{cloud}} + T_i^{\text{trans,cloud}} + T_i^{\text{rec,cloud}} + WT_i^{\text{cloud}},$$

where WT_i^{cloud} is the cloud waiting time for the task.

Energy for the smartphone for cloud offloading is similar to MEC:

$$E_i^{\text{cloud}} = TE_i^{\text{cloud}} + RE_i^{\text{cloud}},$$

and the cost is

$$UC_i^{\text{cloud}} = \beta \cdot E_i^{\text{cloud}}.$$

D. Objective Function

For each task T_i, define its latency, energy, and usage cost contributions as

$$L(T_i) = L_i^{\text{local}} + L_i^{\text{MEC}_k} + L_i^{\text{cloud}},$$
$$E(T_i) = E_i^{\text{local}} + E_i^{\text{MEC}_k} + E_i^{\text{cloud}},$$
$$UC(T_i) = UC_i^{\text{MEC}_k} + UC_i^{\text{cloud}}.$$

Note that each task is executed at exactly one location; e.g., if executed locally then $L_i^{\text{MEC}_k} = L_i^{\text{cloud}} = 0$, $E_i^{\text{MEC}_k} = E_i^{\text{cloud}} = 0$, and $UC_i^{\text{MEC}_k} = UC_i^{\text{cloud}} = 0$.

The metrics for the entire application are:

$$L = \max_i \{ L(T_i) \}, \qquad E = \sum_i E(T_i), \qquad UC = \sum_i UC(T_i).$$

The fitness function Z

$$\text{Minimize} \quad Z = a_1 L + a_2 E + a_3 UC,$$

subject to

$$a_1 + a_2 + a_3 = 1, \qquad 0 \le a_1, a_2, a_3 \le 1.$$

IV. Research Methodology

This study evaluates the performance of an Enhanced Whale Optimization Algorithm (EWA) for dependent-task offloading in a heterogeneous computing environment that includes a local device, multiple MEC servers, and a cloud server. The methodology consists of two parts: the design of the optimization algorithms and the experimental procedure used for evaluation.

A. Design of Optimization Algorithms

Two optimization strategies were implemented: the standard Whale Optimization Algorithm (WOA) and an enhanced variant (EWA). Both algorithms operate on continuous-valued vectors that are later mapped to discrete task–location assignments.

1) Standard WOA (Baseline): The baseline WOA follows the original encircling, bubble-net, and random-search mechanisms proposed in the classical whale optimization model. It serves as the reference for evaluating improvements introduced in the enhanced version.

2) Enhanced WOA (EWA): The EWA integrates two additional operators to increase solution diversity, improve exploration, and enhance convergence characteristics.

a) Frame-Shifting (FS): The FS operator introduces structured perturbations by:

- shifting the processing-location sequence left or right, and
- cyclically rotating successor tasks within the directed acyclic graph (DAG).

This mechanism prevents premature convergence and allows the algorithm to escape local optima.

b) Load Redistribution Strategy (LRS): The LRS mechanism detects overloaded computing nodes and redistributes selected tasks to underutilized nodes. This approach balances workload, reduces execution bottlenecks, and improves overall latency.

3) Normalization and Encoding: Because both algorithms generate continuous vectors, two conversion operations are used:

1) **Task List Restructuring Strategy (TLRS)** ensures task order respects DAG dependencies.
2) **Linear scaling of processing locations** maps continuous values to discrete computing nodes (local device, MEC servers, and cloud).

These steps ensure feasibility and validity of all generated solutions.

B. Experimental Procedure

The evaluation follows a structured five-step workflow.

1) Step 1: DAG Generation: A directed acyclic graph consisting of eleven dependent tasks is generated to represent the structure of an application with precedence constraints.

2) Step 2: Task Parameter Initialization: Each task is assigned the following randomly generated attributes:

- input size: 10–100 MB,
- output size: 1–10 MB,
- computational requirement: $100\text{–}500 \times 10^6$ CPU cycles.

3) Step 3: Algorithm Execution: Both algorithms are executed with identical parameters:

- population size of 12 solutions,
- 200 iterations,
- continuous-to-discrete conversion at each iteration.

The FS and LRS operators are applied exclusively in the EWA variant.

4) Step 4: Metric Computation: For each candidate solution, the following metrics are calculated:

- per-task latency, energy, and usage cost,
- total latency (L),
- total energy consumption (E),
- total usage cost (UC),
- overall objective function value (Z).

5) Step 5: Convergence Tracking: The minimum objective value at each iteration is recorded to generate convergence curves for performance comparison between WOA and EWA.

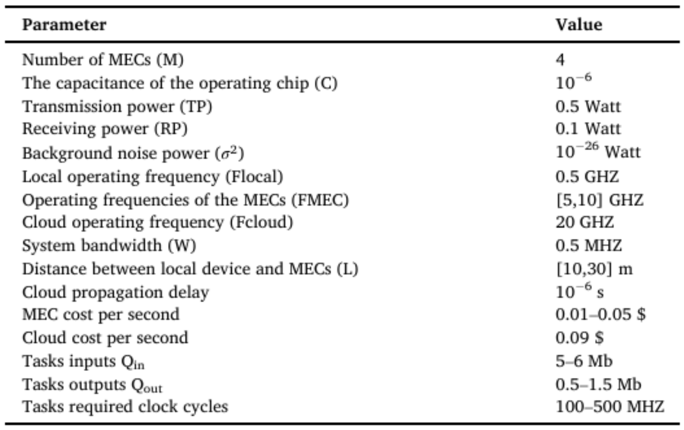
V. Numerical Experiments and Results

For our implementation, we assume that all tasks in each graph can be computationally offloaded, allowing us to measure the performance. We'll assume that M = 4 MECs. We also assume that the data is sent to the cloud through the nearest base station to the local device. To ensure a fair and thorough evaluation of the algorithms, we have certain parameters:

• Number of iterations: We set the number of iterations to 200. This decision allowed ample opportunities for the algorithms to converge and produce meaningful results.

• Population size: Each algorithm's population size was constant at 10.

• Multiple runs for evaluation: To assess the quality and consistency of each algorithm's performance, we executed each program 20 times.



Parameter	Value
Number of MECs (M)	4
The capacitance of the operating chip (C)	10^{-6}
Transmission power (TP)	0.5 Watt
Receiving power (RP)	0.1 Watt
Background noise power (σ^2)	10^{-26} Watt
Local operating frequency (Flocal)	0.5 GHZ
Operating frequencies of the MECs (FMEC)	[5,10] GHZ
Cloud operating frequency (Fcloud)	20 GHZ
System bandwidth (W)	0.5 MHZ
Distance between local device and MECs (L)	[10,30] m
Cloud propagation delay	10^{-6} s
MEC cost per second	0.01–0.05 $
Cloud cost per second	0.09 $
Tasks inputs Q_{in}	5–6 Mb
Tasks outputs Q_{out}	0.5–1.5 Mb
Tasks required clock cycles	100–500 MHZ

Fig. 1. Detailed parameters of the proposed system.

This project focuses on how the FS and LRS enhancement operations contributed to the WOA algorithm. We use four different metrics, finishing time, energy consumption, usage cost, and fitness. The results of four algorithm derivatives (WOA, EWA, WOA-FS, WOA-LRS) are shown in Fig. 4. The results show us the contribution of each operation. It can be seen from the figure that the LRS operation enhanced the standard WOA in all performance measures as follows: the execution latency is improved by 16.43%, the energy consumption is reduced by 78.28%, the cost is mini mized by 35.02%, and the fitness function is improved by 59.54%. The FS improvement operation also has an impact on the produced results. It improved the energy consumption by 78.37%, the cost by 29.49%, and the fitness function by 54.29%, here it fails to improve the execution latency of the application. In examining the results, it is important to note that the FS operation, simulating COVID-19 spreading behavior within the proposed algorithm, positively influenced energy consumption, cost, and fitness function. Combining these two operations has achieved an optimized EWA that enhanced the execution latency by 22.84%, the energy consump tion by 78.28%, the cost by 38.71%, and the fitness function by 61.47%. The minimum value of Z is, the more efficient the algorithm, and in our problem, EWA has the minimum Z value (objective function) than WOA.

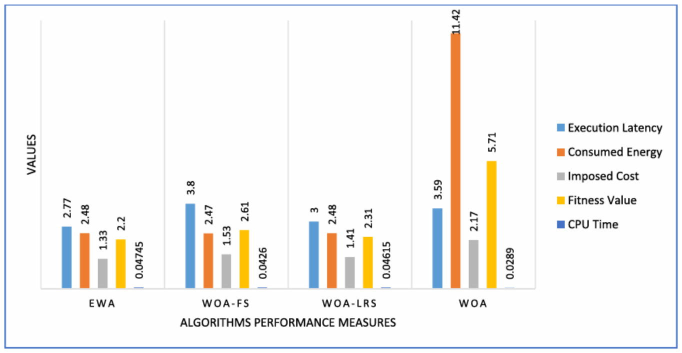


Fig. 2. Performance measures' values for running different derivatives of the WOA algorithm

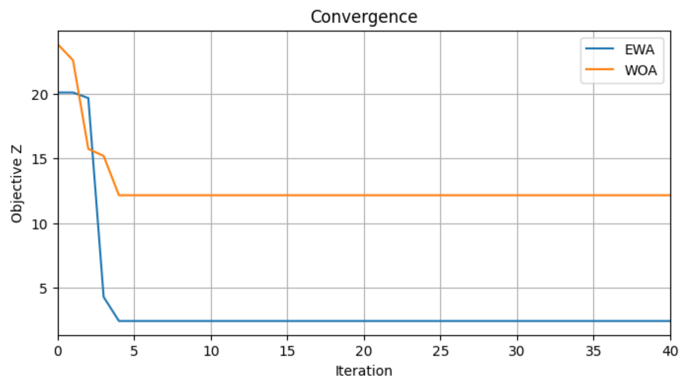


Fig. 3. The Convergence graph of Z for both EWA and WOA

This framework seeks to make it possible for developers of IoT and mobile applications to create effective applications that can most effectively benefit from the MEC and cloud servers. For this reason, our suggested algorithm is used in the design phase in most cases. Since the algorithm's running time in the design phase has no impact on the re sults, we do not consider it a performance indicator.

VI. Conclusion

In this project, three objectives—the latency, energy consumption, and MEC and cloud servers usage fee—were optimized in a multi-edge cloud computing system with task interdependence. We implemented an enhanced whale optimization algorithm with two improvement algorithms to face these challenges. The first algorithm is the frameshifting method, which is inspired by the replication behavior of the COVID-19 virus. The second one is the load redistribution strategy which helps us to maintain a fair load balancing among processing locations. Each agent vector (solution) was divided into two parts: a task list part and a processing locations list part. First, we initialized some agent vectors also known as solutions. Since the resultant vectors produced by the WOA algorithm are continuous values, we transformed them into discrete and bounded vector values. The code we implemented compared EWA to the WOA. In conclusion, EWA has decreased latency by 22.84%, energy consumption by 78.28%, and cost usage by 61.47% compared to WOA.

Acknowledgment

We would like to acknowledge the valuable contributions Khalid M. Hosny, Ahmed I. Awad, Wael Said, Mahmoud Elmezain, Ehab R. Mohamed, Marwa M. Khashaba, Department of Information Technology, Faculty of Computers and Informatics, Zagazig University, Zagazig 44519, Egypt who have presented us with this paper. They have helped us to understand the concepts and methodologies presented in this report. Their significant research on Computation offloading, Task dependency, Mo-bile edge computing, Multi-edge cloud computing, Multi-objective optimization, Enhanced Whale Optimization, Dynamic allocation has greatly inspired us and guided this work.

References

[1] A. I. Awad, M. M. Fouda, M. M. Khashaba, E. R. Mohamed, and K. M. Hosny, “Utilization of mobile edge computing on the Internet of medical things: a survey,” *ICT Express*, vol. 9, no. 3, pp. 473–485, 2023. doi:10.1016/j.icte.2022.05.006.

[2] X. Dai, Z. Xiao, H. Jiang, J.C.S. Lui, ”UAV-assisted task offloading in vehicular edge computing networks,”, IEEE Trans. Mob. Comput. vol. PP (2023) 1–15, https://doi. org/10.1109/TMC.2023.3259394.

[3] Z. Xia, J.A. Abu Qahouq, ”State-of-charge balancing of lithium-ion batteries with state-of-health awareness capability,”, IEEE Trans. Ind. Appl. vol. 57 (1) (2021) 673–684, https://doi.org/10.1109/TIA.2020.3029755.

[4] A. Sarah, G. Nencioni, M.M.I. Khan, ”Resource allocation in multi-access edge computing for 5G-and-beyond networks,”, Comput. Netw. vol. 227 (308909) (2023) 109720, https://doi.org/10.1016/j.comnet.2023.109720.

[5] H. Jin, M.A. Gregory, S. Li, ”A review of intelligent computation offloading in multiaccess edge computing,”, IEEE Access vol. 10 (July) (2022) 71481–71495, https://doi.org/10.1109/ACCESS.2022.3187701.

[6] S. Mirjalili, A. Lewis, ”The whale optimization algorithm,”, Adv. Eng. Softw. vol. 95 (May 2016) 51–67, https://doi.org/10.1016/J.ADVENGSOFT.2016.01.008.

[7] A.M. Khalid, K.M. Hosny, S. Mirjalili, ”COVIDOA: a novel evolutionary optimization algorithm based on coronavirus disease replication lifecycle,”, Neural Comput. Appl. vol. 34 (24) (2022) 22465–22492, https://doi.org/10.1007/ s00521-022-07639-x.