JIRBDAI

Journal of Institutional Research, Big Data Analytics and Innovation

An Enhanced Arithmetic Optimization Algorithm for Efficient Task Scheduling in Fog - Cloud Computing

[1*]Ayuba Liman, [2] Rafiu Mope Isiaka,[3] Nathaniel Akinbowale Babatunde, [4]Aminu Jafar, [5]Toyyib Olaitan Olanrewaju [6]Mustapha Abubakar Giro

[1,4,6] Department of Computer Science Abdullahi Fodio University of science and Technology, Aliero

[2,3,5] Department of Computer Science, Kwara State University, Malete

[*]Corresponding Author: limanjega01@gmail.com

ABSTRACT

The growth of Internet of Things (IoT) applications has increased the demand for efficient task processing in cloud–fog computing environments. Traditional cloud computing often suffers from high latency, network congestion, and inefficient resource use when managing large, heterogeneous workloads. Fog computing mitigates these issues by bringing computational resources closer to end-users, enabling faster task execution and improved Quality of Service (QoS). However, this research addresses the task scheduling problem in fog–cloud environments, focusing on optimal virtual machine (VM) allocation to minimize makespan and energy consumption. We propose an Enhanced Arithmetic Optimization Algorithm (EAOA) that integrates dynamic inertia weights, mutation coefficients, and a triangular mutation strategy to improve the exploration and exploitation capabilities of the standard Arithmetic Optimization Algorithm (AOA), additionally, a network model is introduced to assess and monitor various network performance metrics. Simulation results using NASA iPSC workload traces show that EAOA consistently outperforms existing algorithms, achieving lower makespan, reduced energy consumption, and balanced resource utilization. These findings demonstrate that EAOA is an effective solution for dynamic and heterogeneous IoT-based systems, ensuring efficient task scheduling and optimal use of fog–cloud resources.

KEYWORDS: Fog–Cloud Computing, Task Scheduling, Virtual Machine Allocation, Arithmetic Optimization Algorithm, Enhanced AOA, Makespan, Energy Consumption.

I. INTRODUCTION

In the modern era, the rapid expansion of telecommunication networks has greatly influenced the ongoing Internet of Things (IoT) revolution, which continues to gain widespread adoption[1]. Terminal devices play a crucial role in this ecosystem by sensing their environment and generating data for various applications. However, due to their limited computational and storage capacities, these devices often rely on cloud infrastructure for data storage, processing, analysis, and decision-making**,** leveraging the cloud's superior computational power and scalability[1] Cloud Computing (CC) has become widely adopted across numerous fields [2]. However, new challenges have emerged for traditional cloud infrastructures. In particular, cloud computing faces difficulties in efficiently managing the increasing demand for highly responsive and location-aware IoT services**,** largely due to its limited scalability and high operational and development costs**.** As the number of IoT devices continues to grow, an enormous volume of data must be processed and transmitted

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.21095167

to centralized data centers, further straining the cloud's capacity and responsiveness [3]. Using the traditional cloud computing (CC) model to process such vast amounts of data leads to high latency, network congestion, and reduced system efficiency. To overcome these limitations, Fog Computing (FC) was introduced by Cisco, offering a decentralized computing architecture that extends cloud services closer to end users. This approach enables faster data processing, reduced latency, and improved resource utilization by performing computation[4]. The collaboration between cloud and fog computing aims to effectively handle tasks of varying lengths and computational demands. With the rapid expansion of edge–cloud computing, the volume of user requests has significantly increased, requiring dynamic allocation between cloud and fog nodes based on several factors, such as input task size, task sensitivity, and performance metrics including latency, makespan, cost, and energy consumption. To ensure optimal service delivery, these diverse requests must be efficiently processed to meet user requirements and maintain high system performance[5].However, the distinct characteristics of fog and cloud environments, combined with random user requests and limited resource availability, make task scheduling optimization increasingly complex and essential for discussion. Task scheduling plays a crucial role in enhancing overall system performance by efficiently balancing workloads, reducing network overhead, maximizing resource utilization, and minimizing energy consumption. Therefore, developing effective scheduling strategies is vital for achieving efficient and reliable operations in fog–cloud computing environments[6]. it refers to the process of allocating computational tasks to available resources in an optimized manner to achieve minimal makespan (total task completion time), lower energy consumption, and reduced operational costs. The primary goal is to ensure efficient utilization of resources while maintaining high system performance and meeting application-specific requirements[7]. The core problem addressed in this paper is how to efficiently schedule tasks in a fog–cloud computing environment to achieve minimal makespan and reduced energy consumption, while effectively managing the resource limitations and dynamic behavior of fog servers. Traditional scheduling algorithms often struggle to maintain a balance between exploration and exploitation, resulting in suboptimal performance in complex and real-time computing scenarios. In applications such as augmented reality (AR), virtual reality (VR), and intelligent healthcare systems, inefficient task scheduling can lead to significant delays, high energy consumption, and degraded quality of service (QoS). These challenges ultimately make real-time responsiveness and reliability difficult to achieve for end-users, emphasizing the need for more adaptive and intelligent scheduling approaches[4]. In this aspect, traditional scheduling algorithms often struggle to cope with the dynamic, heterogeneous, and resource-constrained nature of fog–cloud environments. Their deterministic behavior and limited adaptability make them unsuitable for handling the rapidly changing workloads and diverse computational requirements typical of such systems[8]. Consequently, advanced optimization techniques, particularly metaheuristic algorithms, have gained significant prominence for their ability to adaptively explore complex search spaces, balance computational loads, and enhance overall system performance in fog–cloud computing environments especially Arithmetic Optimization Algorithm(AOA)[9]. A new meta-heuristic optimization Algorithm which leverages the distributional behavior of the four fundamental arithmetic operators in mathematics Multiplication (M), Division (D), Subtraction (S), and Addition (A)[10]. The Arithmetic Optimization Algorithm (AOA) exhibits several notable strengths that make it highly effective for addressing complex optimization problems. It maintains a strong balance between exploration and exploitation features, a simple and powerful mathematical structure, these features proved (AOA)

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.21095167

effective particularly in cloud–fog resource allocation. In such environments, AOA efficiently optimizes task scheduling and resource allocation, thereby minimizing makespan, improving load balance, and reducing energy consumption[11]. However, despite its advantages, the standard Arithmetic Optimization Algorithm (AOA) exhibits certain limitations. It often suffers from slow convergence speed and a tendency to get trapped in local optima, particularly when dealing with high-dimensional or complex optimization problems. These shortcomings limit its global search capability and overall performance, especially in dynamic environments like fog–cloud computing, where rapid adaptation and efficient resource utilization are crucial[12]. Existing optimization methods such as the Whale Optimization Algorithm (WOA), Particle Swarm Optimization (PSO), Red Fox Optimization–Arithmetic Optimization Algorithm (RFOAOA), Marine Predators Algorithm (MPA), Artificial Ecosystem-based Optimization (AEO), and Salp Swarm Algorithm (SSA) have demonstrated effectiveness across various optimization domains. However, these algorithms often struggle to adapt swiftly to the dynamic and heterogeneous nature of fog–cloud computing environments. The frequent fluctuations in resource availability, task priorities, and network conditions can hinder their responsiveness, resulting in suboptimal task allocation, increased makespan, and higher energy consumption.

To overcome these limitations, this paper proposes an Enhanced Arithmetic Optimization Algorithm (EAOA) that integrates enhanced strategies to establish a more effective balance between exploration and exploitation throughout the optimization process. The proposed EAOA is specifically designed for task scheduling in fog–cloud computing environments, aiming to enhance resource utilization, reduce makespan, and minimize energy consumption. This improvement ensures faster convergence, better adaptability to dynamic workloads, and more efficient handling of resource constraints in heterogeneous fog–cloud infrastructures. Unlike closely related hybrid algorithms such as RFOAOA, which combine two metaheuristic strategies to enhance global search capability, the proposed Enhanced Arithmetic Optimization Algorithm (EAOA) introduces structural improvements directly within the Arithmetic Optimization Algorithm framework. While RFOAOA relies on external hybridization between Red Fox Optimization and AOA, EAOA enhances the internal search dynamics of AOA through adaptive mechanisms, including dynamic inertia weights, iteration-dependent mutation probability, and a triangular mutation strategy. These enhancements allow EAOA to achieve a more effective balance between exploration and exploitation throughout the optimization process, improve convergence speed, and reduce the likelihood of premature convergence. Moreover, EAOA maintains a simpler algorithmic structure and lower integration complexity compared to hybrid approaches, while delivering superior performance in terms of makespan and energy consumption. This distinguishes the proposed approach as a more efficient and scalable alternative for fog–cloud task scheduling. To assess the effectiveness of the proposed approach, extensive experiments were conducted under diverse workload scenarios of varying sizes**.** These experiments evaluated the performance of the Enhanced Arithmetic Optimization Algorithm (EAOA) in terms of key performance metrics such as makespan time and energy consumption**.** Additionally, a comparative analysis was performed to benchmark EAOA against several state-of-the-art task scheduling algorithms, demonstrating its superiority in optimizing fog–cloud resource allocation. The main contributions of this paper are summarized as follows:

(a). Modeling and formulation of task scheduling problem in fog – cloud computing
(b). Development of an Enhanced Arithmetic optimization algorithm used to improve the AOA algorithm local search strategy for efficient task scheduling

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.21095167

(c). Evaluation of the proposed EAOA algorithm using Energy consumption and makespan evaluation metrics and compared the results with the existing state of the art algorithms such as SSA, RFO and RFOAOA, to show significant improvements in terms of energy consumption and makespan.

The remaining sections of this paper are organized as follows: Section 2 provides a comprehensive review of related studies on task scheduling within fog–cloud computing environments. Section 3 describes the problem formulation**,** defining the mathematical model and key optimization objectives. Section 4 presents the proposed model and an Enhanced Arithmetic Optimization Algorithm (EAOA) in detail. Section 5 discusses the simulation setup, results, and performance analysis of the proposed method compared with existing algorithms. Finally, Section 6 concludes the paper and outlines potential future research directions aimed at enhancing scheduling efficiency and scalability in fog–cloud systems.

II. RELATED WORKS

The author in [4] developed a Hybrid Enhanced Optimization-Based Intelligent Task Scheduling framework for sustainable edge computing to address key challenges such as load instability, slow convergence rates, and underutilization of virtual machines, while optimizing for energy consumption and makespan time. The proposed hybrid method, termed RFOAOA, integrates the strengths of Red Fox Optimization (RFO) and the Arithmetic Optimization Algorithm (AOA) to enhance task allocation efficiency and system stability. Experimental results demonstrated that the RFOAOA approach outperforms several state-of-the-art methods in minimizing both energy consumption and makespan. According to [13], an Energy and Cost-Aware Real-Time Task Scheduling with Deadline Constraints (ECaTSD) approach was developed for the fog computing environment to minimize energy consumption and monetary costs while ensuring that tasks meet their deadline constraints. The proposed ECaTSD algorithm achieved remarkable efficiency, with a 99.58% task completion rate, demonstrating superior performance in terms of energy savings and cost-effectiveness compared to existing scheduling policies. However, the main limitation of this approach is that, although it enhances energy efficiency, it suffers from inefficient learning behavior and longer convergence times, which may hinder its adaptability in highly dynamic fog–cloud environments. However, its main limitation lies in the lack of consideration for task dependency and scalability, which restricts its applicability in more complex and large-scale fog–cloud environments. According to [14], a Deep Reinforcement Learning-based Intelligent Scheduling (DRLIS) approach was proposed to optimize system load and response time in edge and fog computing environments. The method adaptively and efficiently optimizes the response time of heterogeneous IoT applications while balancing the load among edge and fog servers. The experimental results show that the DRLIS approach significantly reduces the execution cost of IoT applications by 55%, 37%, and 50% in terms of load balancing, response time, and weighted cost, respectively. However, the major limitation of the DRLIS algorithm is its long convergence time, which makes it less practical for real-time IoT applications that require rapid decision-making and adaptability. The author [15] proposed Task Scheduling Optimization in Cloud-Fog MCS Environment Using Genetic Algorithm and Game Theory, focusing on minimizing makespan time, energy consumption, and degree of imbalance. The approach utilizes the Genetic Algorithm (GA) to enhance task allocation efficiency and resource utilization. Experimental results revealed that the proposed method achieved a 26% reduction in makespan, a 32.4% decrease in energy consumption, a 28% reduction in total system cost, and a 21.53% decrease in the degree of

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.21095167

imbalance compared to other scheduling approaches. However, the major limitation of the Genetic Algorithm (GA) is its tendency to prematurely converge to suboptimal solutions, often getting trapped in local optima instead of identifying the global optimum, especially in complex and dynamic scheduling environments. The author [1] developed a Multi-objective Grey Wolf Optimizer (MOGWO) Algorithm for Task Scheduling in Cloud-Fog Computing with the primary objective of reducing delay and minimizing energy consumption. The proposed MOGWO effectively manages the trade-off between delay and energy usage, ensuring a balanced and efficient scheduling process. Simulation results confirmed that the algorithm outperforms related approaches in terms of reducing delay and energy consumption, demonstrating its robustness and effectiveness. However, the main limitation of MOGWO is that its convergence rate to the optimal Pareto front can be slow, particularly when dealing with complex and high-dimensional multi-objective optimization problems, which may affect its performance in real-time dynamic environments. According to [16], an Efficient Offloading and Task Scheduling approach in Internet of Things (IoT) Cloud-Fog Environments was developed to ensure Quality of Service (QoS) by minimizing completion time, energy consumption, and cost using a Genetic Algorithm (GA) with a weighted sum objective function. The experimental results show that LCO performs optimally for latency-sensitive applications, EBO is more effective in energy reduction scenarios, and EO is suitable for applications seeking a balanced across all metrics. However, the trade-off limitation of this research lies in its restricted scalability, as the approach only considers a maximum of 100 tasks, making it unsuitable for competitive or large-scale environments due to limited solution diversity and reduced adaptability in handling complex workloads. The author [17] proposed a Multi-Swarm Particle Swarm Optimization (MS-PSO) Algorithm for Static Workflow Scheduling in Cloud-Fog Environments, aiming to minimize makespan, cost, energy consumption, and improve load balancing across both cloud and fog tiers. The experimental results indicate that MS-PSO consistently outperforms the canonical PSO across all scientific workflows and performance metrics, demonstrating superior scheduling efficiency and resource utilization. However, the limitation of this approach is its tendency to converge prematurely toward suboptimal solutions, as particles may get trapped in local optima rather than achieving the global optimal solution, especially in complex and high-dimensional search spaces. The author [18]developed a Real-Time Task Scheduling Algorithm for IoT-Based Applications in the Cloud–Fog Environment with the primary objective of achieving an effective trade-off between key performance metrics, such as makespan and total execution cost, using an Improved Genetic Algorithm based on Population Opposition-Based Learning (IGA-POP)**.** The experimental results demonstrate that the proposed approach enhances scheduling efficiency and achieves better optimization in real-time environments. However, the limitation of this study lies in its significant failure rate in certain scenarios, for instance, the algorithm recorded a failure rate of 45.3 ± 0.8 when executing 400 tasks under scenario 1, indicating instability and reduced robustness when handling large-scale or highly dynamic workloads.

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.21095167

Table 1: Summary of Related Works on Task Scheduling in Cloud – Fog Computing

Author (Year)	Problem Addressed	Algorithm	Evaluation Metrics	Strengths	Limitation
[4]	Minimize energy consumption and makespan time on fog – cloud computing	RFOAOA	Energy consumption and makespane	Consider load instability and underutilization of VMs	Not consider VM availability
[13]	Minimize Energy consumption and monetary costs	ECaTSD	Energy consumption and cost	shows high efficiency in meeting deadlines (99.58% completion rate)	it lead to inefficient learning and longer convergence times
[14]	Minimize response time and load balancing	DRLIS	Response time and load balancing	Reduces the execution cost of IoT applications by up to 55% and 37%, in terms of load balancing and response time	It takes long time to converge to an optimal solution, which is not efficient for real-time IoT applications
[15]	Minimize completion time, energy consumption, and cost.	Genetic Algorithm	Completion time energy	reduced the makespan, energy consumption and cost by (26%, 32.4% and 28%)	can converge prematurely to suboptimal solutions, getting stuck in local optima rather than finding the global optimum.
[1]	Minimize delay and energy consumption	MGWO	Delay and energy consumption	Balance tradeoff between delay and Energy consumption	No complexity analysis
[16]	Minimize completion time, energy	Genetic Algorithm	Completion time, energy consumption, and cost.	The simulation result is optimal for latency-sensitive applications	not suitable when dealing with competitive

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.21095167

	consumption, and cost.				environment with large number of task
[17]	Minimize Makespan, cost, energy and load balancing	MS-PSO	Makespan, cost, energy and load balancing	MS-PSO outperforms the canonical PSO	Lacks of adaptability to dynamic conditions
[18]	Minimize makespan and total execution cost	IGA-POP	makespan and total execution cost	Balance the tradeoff between makespan time and execution cost	Lack task complexity discussion
[19]	Minimize Makespan time and throughput	CHMPAD	Makespan time and throughput	shows makespan time improvements of 1.12–43.20% (for synthetic workloads), 1.00–43.43% (for NASA iPSC workloads), and 2.75–42.53% (for HPC2N workloads	Does not consider task Defendancy for all the workload trace
[20]	Minimize Makespan time, energy consumption and cost	EEOA	Makespan time, energy consumption and cost	The approach improves efficiency by 89%, energy consumption by 94%, and total cost by 87% over existing algorithms	Cannot predict upcoming workloads

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.21095167

III. METHODOLOGY

Figure 1, presents the proposed Fog–Cloud task scheduling model based on the Enhanced Arithmetic Optimization Algorithm (EAOA), augmented with a network model for assessing and monitoring network performance metrics. The architecture is structured into three main layers: the IoT devices layer, the fog–cloud layer, and the task scheduling and optimization layer. At the IoT layer, heterogeneous devices such as sensors, smartphones, cameras, and smart terminals generate computation-intensive tasks with diverse Quality of Service (QoS) requirements. Each task is divided into several smaller tasks (Task 1, Task2, …, Taskn) and transmitted to the scheduler through wireless communication links. A network model is incorporated to continuously evaluate key network performance metrics, including latency, bandwidth utilization, transmission delay, and link reliability, which directly influence task offloading and scheduling decisions. The task scheduling process begins with the computation of the Expected Time to Compute (ETC) matrix, which estimates task execution times across available VMs while considering network-induced delays. The EAOA optimization module then utilizes this information to determine an optimal task-to-VM mapping that balances multiple objectives, including makespan reduction and energy consumption. Once the optimal solution is identified, tasks are sent to fog broker and assigned to the selected VMs for execution in either fog server, cloud server or both. The QoS Monitoring module checks if each task is correctly matched with its VM. When a match is confirmed, the task will be processes, after processing, results are returned to the IoT devices through the Fog Broker. If the QoS monitor detects a mismatch, it triggers the feedback loop to the scheduler, allowing the EAOA to re-evaluate and reassign the tasks. This continuous feedback process improves system performance, reduces delay, and enhances the overall Quality of Service (QoS) within the fog–cloud computing environment.

A. Problem Description

This Section, present the mathematical formulation of task scheduling problem for fog-cloud computing environment. Let assume there are a large number of physical servers available in the fog-cloud computing environment with variable specifications (i.e., storage capacity, RAM size, CPU cores, and network bandwidth). These servers can be scaled up or down to achieve the Quality of Service (QoS) requirements[19]. Let's assume $VM = \{VM_1, VM_2 \dots \dots VM_n\}$ represents a collection of virtual machines in the fog - cloud environment. The processing capabilities of each VM_y are measured in MIPS (millions of instructions per second). Let $t = \{t_1, t_2 \dots, t_n\}$ denotes a set of tasks provided by fog - cloud users to be completed on the data center's collection of VMs. The task length TL_x of each task t_x is measured in millions of instructions (MI). The Expected Time to Compute (ETCom) is used to keep track of the time it takes to conduct a specific task on different virtual machines [21]. $ETCom_{xy}$ refers to the expected time to compute of the task x on VM_y computed as follows:

$$ETCom_{xy} = \frac{TL_x}{VMP_y}, \quad 1 \le i \le n, 1 \le j \le m \qquad (1)$$

where TL_x is the length of task x and VMP_y refers to the processing power of VM_y. The objective of our study is to enhance the QoS in terms of Makespan and energy efficiency. Makespan refers to the total time required to complete all com- putational tasks. Therefore, proper job mapping is required to necessitate shorter Makespan. Makespan (MKS) is calculated as follows

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.21095167

$$MKS = Max_{i\epsilon(1,2.....,m)} \sum_{x=1}^{n} ETCom_{xy} \quad (2)$$

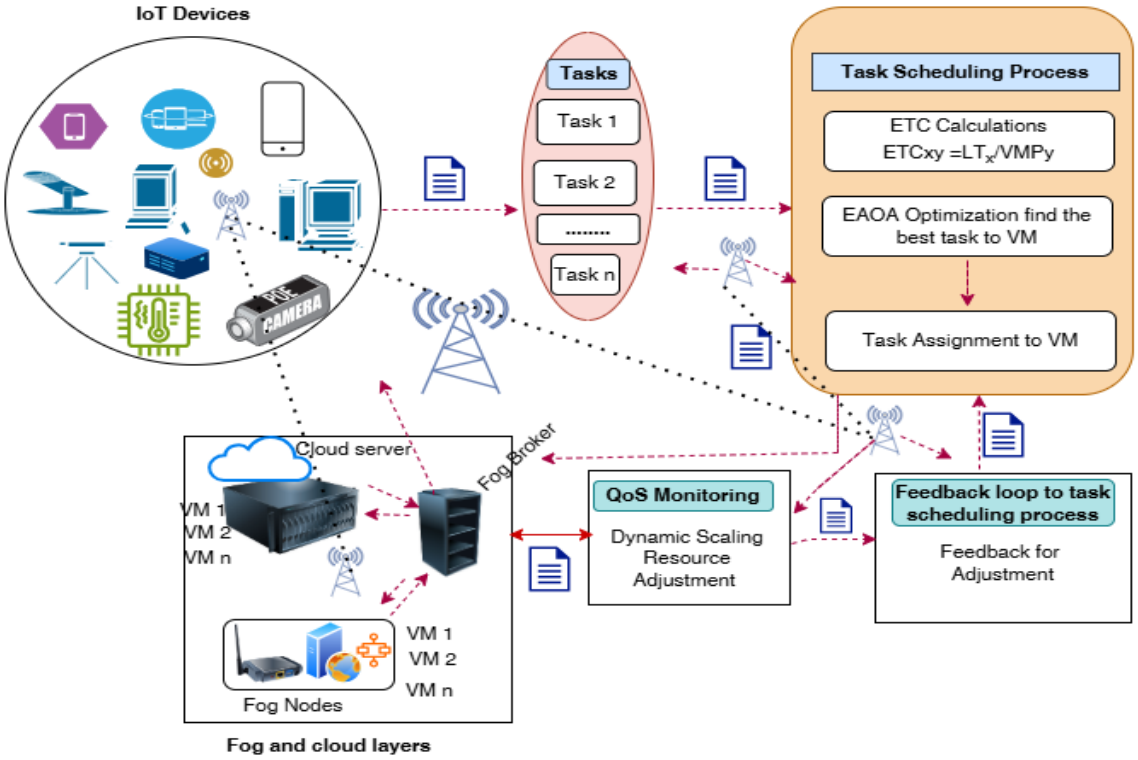

Figure 1: Task scheduling Model on Fog - Cloud Computing

The amount of total energy utilized by computing resources to complete the total tasks is called energy consumption and the objective is to kept it minimum to improve system performance and to deliver the required QoS. However, the total energy consumed by VM_y. equals the amount of energy used in its active and idle states[22]. However, it is predicted that a VM roughly consumes 60% of total energy in idle state only. Thus, energy consumption Eng(VM_y) for a particular VM_y can be calculated as follows:

$$Eng(VM_y) = \langle TE_y \times \beta_y + \langle MKS - TE_y \rangle \alpha_y \rangle \quad (3)$$

$$\beta_y = 10^{-8} \times VMP_y^2 \quad (4)$$

$$\alpha_y = 0.6 \times \beta_y \quad (5)$$

Where TE_y refers the total execution time of VM_y, α_y and β_y denote the consumed energy by VM_y in the idle state and in active state, respectively. Thus, the total energy consumption (TEng) of a fog - cloud system can be calculated as:

$$TEng = \sum_{y=1}^{m} Eng(VM_y) \quad (6)$$

Where m represents the total number of virtual machines. As energy consumption and Makespan have more impact on a fog - cloud system overall performance. Therefore, our primary goal is to

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.21095167

improve the Makespan while reducing total energy consumption. Thus, our task scheduling problem becomes a bi-objective optimization problem whose fitness function is defined as follows:

$$F = (1 - \lambda) \times MKS + \lambda \times TEng. \quad (7)$$

Where λ represents the balancing parameter between the factors in the fitness function. Thus, our task scheduling objective is finding a schedule with the lowest F.

B. Network Model

The network connection model in the fog–cloud computing continuum is designed to optimize interaction and data flow among IoT devices, fog servers, and cloud servers. In this model, each IoT device, such as smartphones and sensors, connects to a fog server through a dedicated communication link. This design enables low-latency data processing, which is essential for real-time applications. Fog servers are further interconnected with one or more cloud servers, providing a pathway for tasks that require extensive computational resources or large-scale storage. This hierarchical architecture ensures seamless data and task flow between IoT devices and cloud servers, thereby balancing local processing efficiency with the computational power of the cloud. Moreover, the flow conservation constraint ensures that the amount of data entering a node is equal to the amount of data leaving it, thereby maintaining system stability. Interconnections among fog servers also allow IoT devices to offload tasks to one fog server and receive processed results from another. This network model plays a crucial role in optimizing resource utilization, reducing latency, and satisfying the Quality of Service (QoS) requirements of diverse applications, ultimately enhancing the overall performance and reliability of the fog–cloud computing system.

Let D = (ID_1 ID_2 ID_n) be the set of IoT Devices and let $F = (\ FS_1 \quad FS_2 \quad FS_n\ )$ be the set of Fog Servers and $C = (\ CS_1 \quad CS_2 \quad CS_n\ )$ be the set of cloud servers in the continuum. The network model is computed according to the following rules.

Each IoT devices must be connected to exactly one Fog server

$$\sum_{j \in F} Zij = 1, \ \forall ID_i \in D \quad (8)$$

Where Zij is the binary variable indicating if IoT Devices ID_i is connected to fog server FS_j (1 if true, 0 otherwise). Each fog server can be connected to multiple cloud servers

$$\sum_{j \in C} wij \geq 0, \ \forall FS_i \in F \quad (9)$$

Where wij is the binary variable indicating if fog server FS_i is connected to cloud server CS_j (1 if true, 0 otherwise). The flow of data must be conserved from the source to the destination.

$$\sum_{k:(k,i) \in L} fki \ - \sum_{k:(k,i) \in L} fij \ = b_i 0, \ \forall_i \in D \cup F \cup C \quad (10)$$

Where b_i is the flow demand at node i, fij is the flow of data from node i to node j, L is the set of network links and k is an index for nodes from which data is flowing in to node i.

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.21095167

Table 2: Key Notations

Symbols	Meaning
TL_x	Length of task x
VM	Set of virtual machines in data centre
VM_y	Virtual machines y
λ	Balancing Parameter
m	Total Number of virtual machines
TE_y	Total execution time of virtual machine VM_y
α_y	Consumed energy by VM_y in the idle state
β_y	Consumed energy by β_y in the active state
F	Fitness function
$ETCom_{xy}$	ETCom of the task x on virtual machine VM_y
$\alpha[0, d_i, b]$	Scaling hyper- parameter
n	Number of tasks
N	Number of solution
t_{max}	Number of iteration
$TEng$	Total energy consumed by a cloud system

C. Arithmetic Optimization Algorithm (AOA)

The Arithmetic Optimization Algorithm (AOA) [23]. is a recently developed population-based metaheuristic algorithm introduced by Abualigah et al. It draws inspiration from the use of arithmetic operators' addition, subtraction, multiplication, and division in solving mathematical problems. The algorithm can effectively address optimization problems without requiring derivative computations, making it highly applicable and valuable across various disciplines. For instance, Khatir et al. [24] applied the AOA to detect, locate, and quantify damage in functionally graded material (FGM) plate structures. Similarly, Deepa et al. [25] employed the algorithm in their research to address complex optimization challenges within their respective domain. Ahmadi et al.[10]utilized the AOA to determine the optimal placement of various types of distributed generations (DGs) and energy storage systems (ESSs), aiming to achieve an efficient and balanced energy distribution layout. Bhat et al. [26]applied the AOA to optimize the deployment of wireless sensor networks (WSNs). However, like many other metaheuristic algorithms, AOA faces certain limitations, including slow convergence and a tendency to become trapped in local optima. Consequently, several researchers have proposed various enhancements and applications to improve the algorithm's performance and overcome these challenges. For instance, Kaveh et al. [40] refined the original AOA formulation to improve its exploration and exploitation capabilities, applying the enhanced version to the optimization of skeletal structures with discrete design variables. Similarly, Agushaka et al. [27] incorporated natural logarithm and exponential operators to strengthen the exploration ability of AOA and successfully applied it to the optimization of welded beam design. Although AOA has been widely applied in various fields, one of its inherent limitations is the tendency to fall into local optima and exhibit slow convergence during the search

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.21095167

process. This issue arises because, in AOA, individual updates are primarily conducted around a single global best position, which restricts population diversity and limits the algorithm's exploration capability. As noted by Jamil et al. [25], this approach makes the search strategy overly selective, as individuals that depend on a single, centrally guided position update may fail to converge toward the true global optimum. Therefore, in this research work, dynamic inertia weights are incorporated to improve the convergence speed of the AOA, while dynamic probability coefficients and triangular mutation strategies are introduced to enhance its capability to escape from local optima.

D. Standard AOA

The AOA is a recently developed metaheuristic algorithm introduced in 2021[9]. In the standard AOA Arithmetic operators such as (multiplication, division, subtraction, and addition) serve as the primary inspiration behind the design of the AOA algorithm. The AOA employs multiplication and division operators to facilitate global exploration, while addition and subtraction operators are used to perform local exploitation.

(a). Math Optimizer Accelerated (MOA) Function

The AOA determines the search phase whether to perform global exploration or local exploitation based on the Math Optimizer Accelerated (MOA) parameter. A random number r_1 (ranging between 0 and 1) is generated; if r_1>MOA(t), the algorithm proceeds with global exploration; otherwise, it performs local exploitation. The mathematical representation of MOA is expressed as follows in Equation (8):

$$MOA(t) = Min + t \times \left(\frac{\text{Max} - Min}{T}\right) \quad (11)$$

Here, t represents the current iteration number, T denotes the maximum number of iterations, and Max and Min correspond to the maximum and minimum values of the Mathematical Optimizer Acceleration (MOA) function, respectively.

(b). Global Exploration

At this stage, the AOA primarily employs two search strategies the division search strategy and the multiplication search strategy to explore better candidate solutions. r_2The mathematical formulation of this search process is presented in Equation (12):

$$X(t+1) = \begin{cases} best(x) \div (MOP(t) + \varepsilon) \times L, & r_2 < 0.5 \\ best(x) \times MOP(t) \times L, & otherwise \end{cases} \quad (12)$$

$$MOP(t) = 1 - t^{1/\alpha}/T^{1/\alpha} \quad (13)$$

$$L = (UB - LB) \times \mu + LB \quad (14)$$

Here, x(t + 1)) represents the position of an individual at iteration t + 1; best(x) indicates the position of the best individual among the current candidate solutions; ε is a small constant used to prevent division by zero; UB and LB denote the upper and lower bounds of the search space, respectively. Additionally, μ serves as a control parameter to regulate the search process, MOP(t) is the mathematical optimization rate coefficient, and α represents the sensitivity parameter that influences the precision of iterative development.

(c). Local Exploitation

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.21095167

At this stage, the AOA primarily utilizes two strategies for local exploitation the subtractive search strategy and the additive search strategy. A random number r_3 is generated within the range [0,1]; If $r_3 < 0.5$, the subtractive search strategy is applied, otherwise, the additive search strategy is executed. The mathematical representation of this search process is expressed in Equation (12)

$$X(t+1) = \begin{cases} best(x) - MOP(t) \times L, & r_3 < 0.5 \\ best(x) + MOP(t) \times L & otherwise \end{cases} \quad (15)$$

E. Enhanced EAOA

As discussed earlier, the AOA possesses a simple structure with relatively few parameters. However, its position update equations exhibit limited randomness among the search agents, resulting in restricted exploration ability and a higher tendency to become trapped in local minima. To address this limitation, this study introduces two key modifications to the original AOA framework, which are explained in detail in the following sections.

(a). Dynamic Inertia Weights

Inertia weights, first introduced by Shi and Eberhart, play a key role in balancing exploration and exploitation. Larger inertia weights promote global exploration by enabling broader searches of the solution space, while smaller inertia weights enhance local exploitation by refining solutions around promising regions[9]. Therefore, in this study, a nonlinear and exponentially decreasing inertia weight is introduced, along with dynamic coefficient-based inertia weights, to enhance the search efficiency of the AOA (Algorithm 1) and accelerate its convergence. The incorporation of dynamic coefficients increases the adaptability of the inertia weights, allowing the improved algorithm to introduce controlled perturbations that enhance the flexibility of the best individual. This adjustment helps mitigate the tendency of the algorithm to become trapped in local optima caused by position updates centered solely around the current best solution. The formulation of the dynamic inertia weights is expressed in Equation (16):

$$W(t) = C * w_{begin} \left(\frac{W_{begin}}{W_{end}}\right)^{1/(1+t/T)} \quad (16)$$

Where the maximum and minimum values of W_{begin} and W_{end} inertia weights, c are random values that vary dynamically around value 1. Where the sum of dynamic inertia weights is introduced to Equations (12) and (15), its updated formula becomes Equations (17) and (18):

Here, W_{begin} and W_{end} represent the maximum and minimum values of the inertia weights, respectively, while c denotes a random coefficient that varies dynamically around the value of 1. When the sum of these dynamic inertia weights is incorporated into Equations (12) and (15), the updated formulations are represented as Equations (17) and (18):

$$X(t+1) = \begin{cases} w(t) * best(x) \div (MOP(t) + \varepsilon) \times L, & r_2 < 0.5 \\ w(t) * best(x) \times MOP(t) \times L, & otherwise \end{cases} \quad (17)$$

$$X(t+1) = \begin{cases} w(t) * best(x) - MOP(t) \times L, & r_3 < 0.5 \\ w(t) * best(x) + MOP(t) \times L & otherwise \end{cases} \quad (18)$$

(b). Dynamic Coefficient of Mutation and Triangular Mutation Strategy

Drawing inspiration from the “mutation” operation in genetic algorithms, this study introduces a dynamic mutation probability coefficient that increases progressively with the number of iterations. This mechanism allows individuals to explore alternative search spaces, thereby broadening the overall search range and improving the algorithm’s ability to escape from local optima. The dynamic mutation probability coefficient is defined in Equation (12). Additionally,

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.21095167

the triangular mutation strategy[23] fully utilizes information from multiple individuals within the population, enabling mutual exchange of solution characteristics. This promotes population diversity and helps prevent premature convergence during the search process. One formulation of the triangular mutation strategy is expressed in Equation (13):

$$P = 0.2 + 0.5 * t/T \quad (19)$$

$$X(t) = (X_{r1} + X_{r2} + X_{r3})/3 + (t_{2-} t_1) * (X_{r1} - X_{r2}) + (t_3 - t_2) * (X_{r2} - X_{r3}) + (t_1 - t_3) * (X_{r3} - X_{r1}) \quad (20)$$

Here, p represents the mutation probability coefficient, which increases progressively with the number of iterations. In the later stages of the algorithm, this higher probability allows individuals in the population to explore alternative search spaces more frequently, thereby reducing the likelihood of the algorithm becoming trapped in local optima. Xr1, Xr2, and Xr3 denote three randomly selected individuals, while (t2 − t1), (t3 − t2), and (t1 − t3) represent the weights of the perturbed components. The triangular mutation strategy functions similarly to the crossover mutation process in genetic algorithms, facilitating the exchange and fusion of information among random individuals. This mechanism helps prevent individuals from updating solely around a single local best position, thereby enhancing the algorithm's capacity to escape local minima and improving overall search diversity. The pseudo-code and flowchart of EAOA is Shown below

Table 3: Algorithm 1: An Enhanced Arithmetic Optimization Algorithm

Algorithm 1: EAOA

1. Set population size N, the maximum number of iterations T.
2. Set up the initial parameters t = 0, α, μ.
3. Initialize the positions of the individuals xi (i = 1, 2, . . . , N).
4. While (t < T)
5. Update the w(t) using Equation (16)
6. Update the MOA using Equation (11) and the MOP using Equation (13)
7. Consider the network connection using Equation (10)
8. Calculate the fitness values using $F = (1 - \lambda) \times MKS + \lambda \times TEng$
9. For i = 1, 2, . . . , N do
10. For j = 1, 2, . . . , Dim
11. Generate the random values between [0, 1] (r_1, r_2, r_3).
12. If r1 > MOA
13. Update the position of x(t + 1) using Equation (17).
14. Else
15. Update the position of x(t + 1) using Equation (18)
16. End if
17. Calculate the p using Equation (19)
18. If p > rand
19. Update the position of x(t + 1) using Equation (20).
20. End if
21. End for
22. End for
23. t = t + 1.
24. End while
25. Return the best solution (x)

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.21095167

Table 4: Simulation Parameters

Cloud Entity	Parameters	Value
Client Data center Host	No. of client No. of data centers No. of host	100 – 200 1 2
Cloud Node		
	No. of Nodes CPU Power RAM Storage Bandwidth Capacity	10 3000 - 5000 MIPS 8GB 1TB 0.5Gb/s
Fog Nodes		
	No. of Nodes CPU Power RAM Storage Bandwidth Capacity	10 1000 – 2800 MIPS 2GB 0.5 TB 1Gb/s
Tasks		
	Task Length Data size Task Number	[1,000 – 5000] [400 - 600]MB [300 - 1500]

F. Algorithmic Complexity and Runtime Overhead

The proposed Enhanced Arithmetic Optimization Algorithm (EAOA) is a population-based metaheuristic whose computational cost is dominated by fitness evaluation and task-to-VM mapping. Let N denote the population size, T the number of iterations, K the number of tasks, and M the number of virtual machines. The overall time complexity of EAOA can be approximated as $O(T \times N \times K \times M)$, which is comparable to baseline algorithms such as SSA, RFO, and RFOAOA. The additional mechanisms introduced in EAOA, including dynamic inertia weights and mutation strategies, involve simple arithmetic operations and do not introduce significant runtime overhead. Although EAOA incurs slightly higher per-iteration computation, this is compensated by faster convergence and improved solution quality. Therefore, EAOA achieves superior optimization performance while maintaining computational efficiency suitable for fog–cloud scheduling environments.

IV. RESULTS AND DISCUSSION

In this section, a comprehensive assessment of the proposed scheme is presented in comparison with other state-of-the-art task scheduling approaches such as (SSA, RFO and RFOAOA). First, the simulation environment and parameter settings are described in detail to establish the experimental context. Next, the performance evaluation metrics used to assess the effectiveness and efficiency of the proposed scheme are discussed. Finally, the simulation results and corresponding discussions are provided to highlight the comparative performance and validate the proposed approach.

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.21095167

A. Experimental Setup

The experimentation was conducted using Python version 3.12, a well-recognized simulation platform [28], on a desktop system equipped with a 2.50 GHz Intel Core i7 processor, 8 GB of RAM, and running the Windows 10 operating system. In this experimental setup, the cloud–fog framework consists of fog nodes with limited processing capacities but with proximity to intelligent IoT devices, resulting in minimal transmission delay. Conversely, cloud nodes possess higher computational capabilities but experience longer communication delays due to their distance from IoT devices. Therefore, to optimize overall system performance, the proposed scheduling algorithm ensures an effective balance between the fog and cloud nodes. The simulated cloud–fog environment includes one data center, two host machines, and a total of ten cloud nodes and ten fog nodes, each configured with varying specifications to reflect realistic heterogeneous conditions.

To evaluate the effectiveness of the proposed EAOA algorithm, both real and synthetic workload traces were utilized. Specifically, real workload data were obtained from the NASA Ames iPSC/860 dataset available in the "Parallel Workload Archive" repository [4]. This dataset provides realistic task arrival patterns, execution times, and resource utilization details, enabling a comprehensive and practical assessment of the proposed algorithm's performance under real-world and simulated workload conditions. The dataset follows the Standard Workload Format (SWF), which is widely used within the scientific community for evaluating scheduling algorithms. The NASA Ames iPSC/860 log consists of 14,794 tasks, providing a realistic and diverse workload pattern. In addition to this real dataset, a synthetic workload comprising 1,500 tasks of varying lengths was also generated to further test the robustness and adaptability of the proposed algorithm under controlled and scalable conditions.

B. Evaluation Metrics

The main objective of this study is to minimize both the total energy consumption and the overall makespan time in fog–cloud computing environments. To validate the effectiveness of the proposed Enhanced Arithmetic Optimization Algorithm (EAOA), its performance is compared with other state-of-the-art task scheduling algorithms. The evaluation focuses specifically on two key performance metrics makespan time and total energy consumption to comprehensively assess the efficiency and optimization capability of the proposed method.

(a). Makespan: This metric represents the total time required to complete all submitted tasks, specifically the time taken by the last task to finish execution. A smaller makespan indicates an efficient and well-balanced task-to-resource mapping, ensuring that user tasks are appropriately distributed across the available computing nodes (CNs). The makespan is mathematically expressed as shown below

$$MKS = Max_{i\epsilon(1,2.....,m)} \sum_{x=1}^{n} ETCom_{xy} \qquad (21)$$

(b). Total Energy Consumption: This metric denotes the total amount of energy consumed by the entire infrastructure, including both fog and cloud nodes, during task execution. For an efficient fog–cloud system, minimizing the energy consumption of virtual machines (VMs) is crucial to achieving optimal resource utilization and sustainability. Therefore, the total energy consumption is computed below

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.21095167

$$TEng = \sum_{y=1}^{m} Eng(VM_y) \quad (22)$$

For comparative analysis, three optimization algorithms, including RFO, SSA, and RFOAOA, are employed to evaluate the performance of the proposed approach. Each benchmark problem is executed independently 30 times, and the average results are considered to ensure reliability and statistical significance. Furthermore, to maintain fairness in comparison, the population size for all algorithms is set to 50, with an inertia weight (ω) of 0.7. The behavior of the proposed EAOA under larger-scale fog–cloud scenarios is examined from a theoretical and workload-driven perspective. As the number of fog and cloud nodes increases to hundreds, the task scheduling problem grows proportionally in search space size and complexity. EAOA is well suited to such scenarios due to its population-based structure and independent fitness evaluations, which enable efficient scaling with increasing numbers of tasks and virtual machines. Although the experimental setup considers a limited number of fog and cloud nodes, the use of large-scale workload traces, including the NASA iPSC dataset with thousands of tasks, reflects realistic high-load conditions. The observed stability in convergence behavior and consistent reductions in makespan and energy consumption indicate that EAOA can effectively handle larger infrastructures by scaling the number of virtual machines and population size accordingly. Moreover, the enhanced exploration and exploitation mechanisms incorporated in EAOA, such as dynamic inertia weights and adaptive mutation strategies, help preserve population diversity and prevent premature convergence as the problem scale increases. These characteristics make the proposed approach suitable for deployment in large-scale fog–cloud environments involving hundreds of nodes. Future work will focus on validating EAOA through large-scale simulations and distributed implementations to further confirm its effectiveness in ultra-dense fog–cloud infrastructures. it is important to examine how the proposed EAOA behaves as the number of tasks and computing nodes increases. EAOA is a population-based metaheuristic whose computational complexity grows approximately linearly with the number of tasks and available virtual machines per iteration. As the fog–cloud infrastructure scales to hundreds of nodes, the parallel nature of task evaluation and fitness computation allows the algorithm to maintain stable performance.

Table 5: Results for Makespane using Synthetic Workload Trace

Algorithms	**Tasks**				
	300	**600**	**900**	**1200**	**1500**
SSA	100	220	320	430	580
RFO	95	209	311	400	525
RFOAOA	80	185	290	395	450
EAOA	74	175	255	374	425

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.21095167

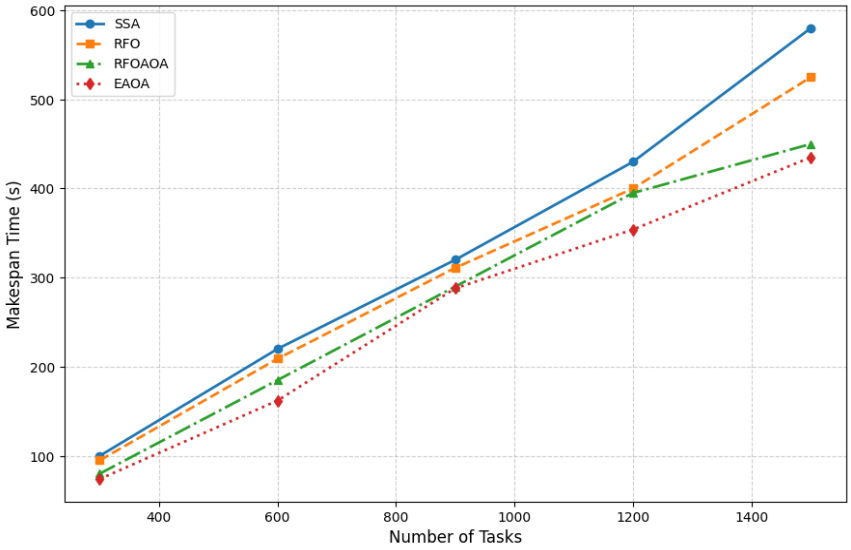

Figure 2: Average Makespan of all Algorithms using the Synthetic Workload

Table 6: Results for Makespane using NASA Ames Workload Trace

Algorithms	Tasks				
	1000	**2000**	**3000**	**4000**	**5000**
SSA	215	485	660	1000	1360
RFO	199	463	600	947	1200
RFOAOA	190	400	570	700	980
EAOA	177	318	552	620	935

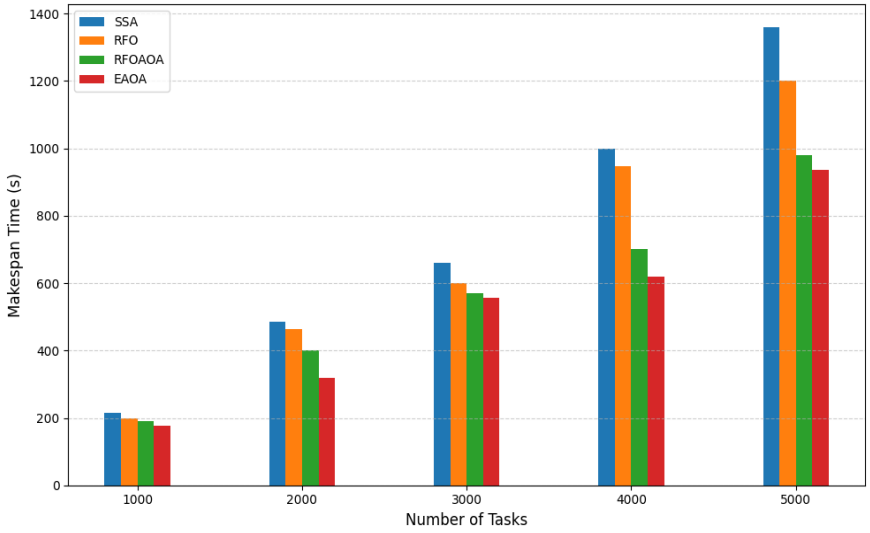

Figure 3: Average Makespan of all Algorithms using NASA iPSC Workload

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.21095167

Figures 2 and 3 illustrate a comparative analysis of the experimental results for both synthetic and real workload traces based on average makespan. As depicted in Figure 2, the proposed EAOA algorithm consistently achieves the lowest average makespan for synthetic workload traces when compared to the standard RFOAOA and other competing algorithms. Likewise, Figure 3 shows that EAOA outperforms all other optimization techniques in terms of average makespan for the real NASA iPSC workload traces.

Table 7: Results for Energy Consumption using Synthetic Workload Trace

Algorithms	Tasks				
	300	**600**	**900**	**1200**	**1500**
SSA	1.72×10^5	2.27×10^6	4.65×10^6	4.46×10^6	5.23×10^6
RFO	1.715×10^5	2.25×10^6	4.65×10^6	4.46×10^6	5.22×10^6
RFOAOA	1.69×10^5	2.19×10^6	4.58×10^6	4.30×10^6	5.10×10^6
EAOA	1.57×10^5	2.18×10^6	4.30×10^6	4.02×10^6	4.87×10^6

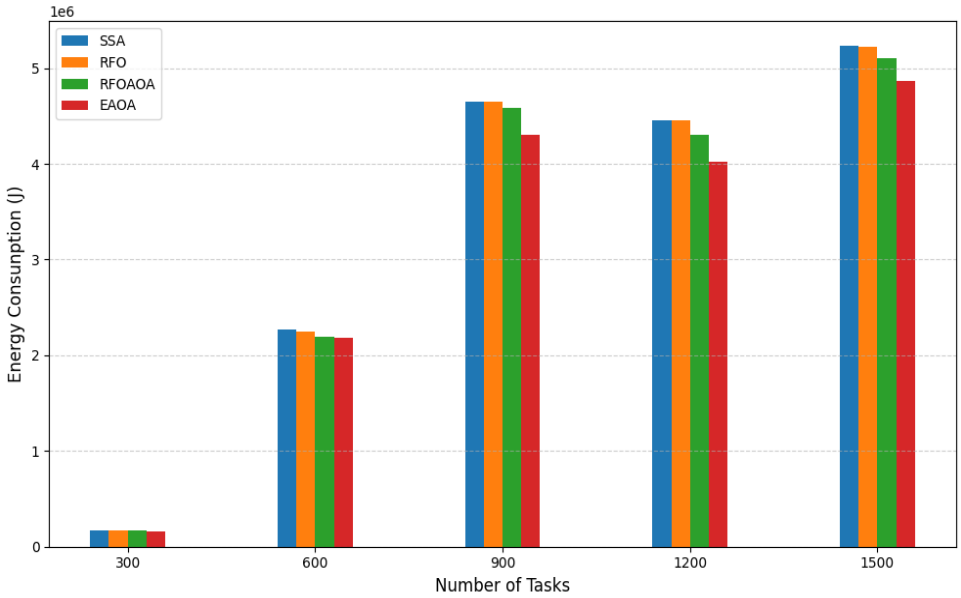

Figure 4: Total Energy Consumption Against Synthetic Workload Traces

Table 8: Results for Energy Consumption using NASA Ames Workload Trace

Algorithms	Tasks				
	1000	**2000**	**3000**	**4000**	**5000**
SSA	0.20×10^6	3.25×10^6	4.45×10^6	6.80×10^6	8.10×10^6
RFO	0.35×10^6	3.20×10^6	4.40×10^6	6.75×10^6	8.01×10^6
RFOAOA	0.30×10^6	3.05×10^6	4.15×10^6	6.05×10^6	7.95×10^6
EAOA	0.11×10^6	2.79×10^6	3.86×10^6	5.88×10^6	7.64×10^6

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.21095167

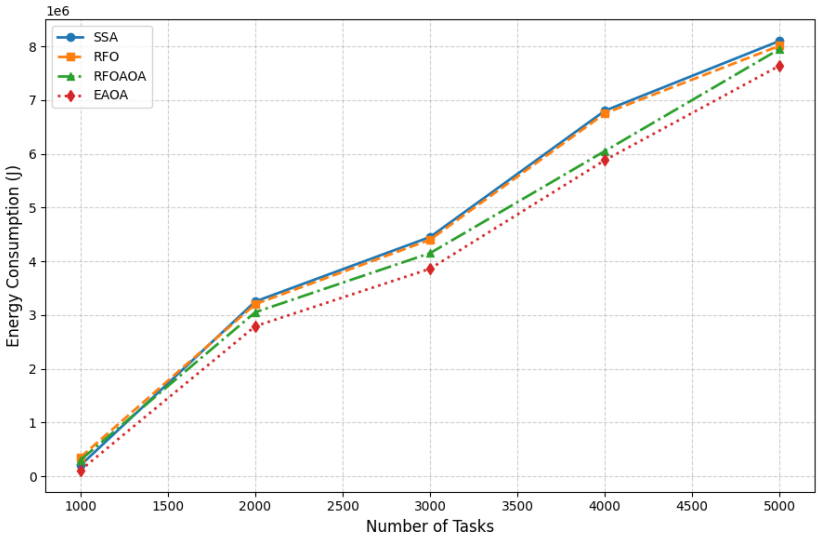

Figure 5: Total Energy Consumption Against Nasa IPSC Workload Traces

Figures 4 and 5 illustrate the total energy consumption of the proposed EAOA algorithm compared to other state-of-the-art optimization algorithms under both synthetic and real workload traces. As shown in Figure 4, EAOA achieves the lowest energy consumption for the synthetic dataset, outperforming the hybrid RFOAOA algorithm and other competing methods. Similarly, Figure 5 demonstrates that EAOA consistently surpasses existing approaches in minimizing total energy consumption for the NASA Ames iPSC workload traces. The simulation results demonstrate that the proposed EAOA algorithm consistently outperforms other methods in terms of Makespan and energy consumption across all workload traces. The integration of the enhanced EAOA improves search efficiency and provides superior solutions for large-scale workload scenarios. However, the algorithm does have some limitations, including increased time complexity and the challenge of selecting the optimal initial population. Overall, the comparative results indicate that EAOA delivers promising performance based on key metrics such as Makespan and energy consumption. Despite these improvements, further work is needed to optimize computational efficiency and refine initial population selection, a common challenge shared with other meta-heuristic techniques.

V. CONCLUSION

In this paper, we propose an enhanced optimization scheme, called Enhanced Arithmetic Optimization Algorithm (EAOA), for efficient task scheduling in fog–cloud computing environments. The EAOA algorithm integrates dynamic inertia weight, dynamic mutation coefficient, and a triangular mutation strategy to expand the search space of the standard Arithmetic Optimization Algorithm (AOA) and a network model is introduced to assess and monitor various network performance metrics. The efficiency of the proposed algorithm is evaluated using multiple performance metrics and compared with state-of-the-art methods, including RFOAOA, SSA, and RFO, under various workload scenarios. Results show that EAOA consistently outperforms these existing methods in terms of Makespan time and energy consumption, highlighting its superior scheduling performance and practical applicability. However, a notable limitation of the proposed scheme is scalability, as performance may be

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.21095167

constrained in large-scale fog–cloud systems with a high number of tasks and resources. For future work, we suggest extending EAOA to address additional optimization problems in cloud-edge environments, such as workflow scheduling and virtual machine placement, and exploring its potential applications in vehicle routing, production scheduling, assembly line balancing, and healthcare facility planning, among others.

REFERENCES

[1] F. A. Saif, R. Latip, Z. M. H. Member, and K. Shafinah, "Multi-objective Grey Wolf Optimizer Algorithm for Task Scheduling in Cloud-Fog Computing," *IEEE Access*, vol. PP, p. 1, 2023, doi: 10.1109/ACCESS.2023.3241240.

[2] A. R. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing : A literature survey," *Futur. Gener. Comput. Syst.*, vol. 91, pp. 407–415, 2019, doi: 10.1016/j.future.2018.09.014.

[3] E. Ahmed *et al.*, "AC PT," *Comput. Networks*, 2017, doi: 10.1016/j.comnet.2017.06.013.

[4] M. A. Elaziz, I. Attiya, L. Abualigah, M. Iqbal, and A. Ali, "Hybrid Enhanced Optimization-Based Intelligent Task Scheduling for Sustainable Edge Computing," *IEEE Trans. Consum. Electron.*, vol. 70, no. 1, pp. 889–898, 2024, doi: 10.1109/TCE.2023.3321783.

[5] F. Hoseiny, S. Azizi, and S. Dabiri, "Using the Power of Two Choices for Real-Time Task Scheduling in Fog-Cloud Computing," pp. 18–23, 2020.

[6] R. Tyagi and S. K. Gupta, "A Survey on Scheduling Algorithms for Parallel and Distributed Systems," pp. 51–64.

[7] J. Aminu, R. Latip, Z. M. Hanafi, S. Kamarudin, B. U. Kangiwa, and A. Liman, "An Enhanced Grey Wolf Optimization Algorithm for Efficient Task Scheduling in Mobile Edge Computing," *Int. J. Comput. Appl.*, vol. 186, no. 56, pp. 39–44, 2024, doi: 10.5120/ijca2024924287.

[8] L. Abualigah and A. Diabat, "A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments," *Cluster Comput.*, vol. 24, no. 1, pp. 205–223, 2021, doi: 10.1007/s10586-020-03075-5.

[9] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, and A. H. Gandomi, "The Arithmetic Optimization Algorithm," *Comput. Methods Appl. Mech. Eng.*, vol. 376, p. 113609, 2021, doi: 10.1016/j.cma.2020.113609.

[10] B. Ahmadi, S. Younesi, O. Ceylan, and A. Ozdemir, "The arithmetic optimization algorithm for optimal energy resource planning," *2021 56th Int. Univ. Power Eng. Conf. Powering Net Zero Emiss. UPEC 2021 - Proc.*, 2021, doi: 10.1109/UPEC50034.2021.9548204.

[11] R. Natarajan, G. Megharaj, A. Marchewka, P. B. Divakarachari, and M. R. Hans, "Energy and Distance Based Multi-Objective Red Fox Optimization Algorithm in Wireless Sensor Network," 2022.

[12] M. Premkumar *et al.*, "A New Arithmetic Optimization Algorithm for Solving Real-World Multiobjective CEC-2021 Constrained Optimization Problems: Diversity Analysis and Validations," *IEEE Access*, vol. 9, pp. 84263–84295, 2021, doi: 10.1109/ACCESS.2021.3085529.

[13] M. Trabelsi and S. Ben Ahmed, "Energy and Cost-Aware Real-Time Task Scheduling with Deadline-Constraints in Fog Computing Environments," no. Enase, pp. 434–441, 2024, doi: 10.5220/0012637600003687.

[14] Z. Wang, M. Goudarzi, M. Gong, and R. Buyya, "Deep Reinforcement Learning-based

scheduling for optimizing system load and response time in edge and fog computing environments," *Futur. Gener. Comput. Syst.*, vol. 152, no. August 2023, pp. 55–69, 2024, doi: 10.1016/j.future.2023.10.012.

[15] A. R. Kadhim and F. Rabee, "Task Scheduling Optimization in Cloud-Fog- MCS Environment Using Genetic Algorithm and Game Theory," vol. 13, no. 3, 2024, doi: 10.18178/ijeetc.13.3.200-213.

[16] M. Gamal, S. Awad, R. F. Abdel-kader, K. Abd, and E. Salam, "Efficient offloading and task scheduling in internet of things- cloud-fog environment," vol. 14, no. 4, pp. 4445–4455, 2024, doi: 10.11591/ijece.v14i4.pp4445-4455.

[17] D. Subramoney and C. N. Nyirenda, "Multi-Swarm PSO Algorithm for Static Workflow Scheduling in Cloud-Fog Environments," *IEEE Access*, vol. 10, no. October, pp. 117199–117214, 2022, doi: 10.1109/ACCESS.2022.3220239.

[18] A. S. A. Amir, E. Ghamry, and E. Hamouda, *Real - Time Task Scheduling Algorithm for IoT - Based Applications in the Cloud – Fog Environment*. Springer US, 2022. doi: 10.1007/s10922-022-09664-6.

[19] I. Attiya, L. Abualigah, D. Elsadek, S. A. Chelloug, and M. A. Elaziz, "An Intelligent Chimp Optimizer for Scheduling of IoT Application Tasks in Fog Computing," pp. 1–18, 2022.

[20] C. Framework, "EEOA : Cost and Energy Efficient Task Scheduling in a Cloud-Fog Framework," 2023.

[21] M. Abd, L. Abualigah, and I. Attiya, "Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments," *Futur. Gener. Comput. Syst.*, vol. 124, pp. 142–154, 2021, doi: 10.1016/j.future.2021.05.026.

[22] I. Attiya, L. Abualigah, S. Alshathri, D. Elsadek, and M. A. Elaziz, "Dynamic Jellyfish Search Algorithm Based on Simulated Annealing and Disruption Operators for Global Optimization with Applications to Cloud Task Scheduling," 2022.

[23] H. Y. Fan and J. Lampinen, "A trigonometric mutation operation to differential evolution," *J. Glob. Optim.*, vol. 27, no. 1, pp. 105–129, 2003, doi: 10.1023/A:1024653025686.

[24] S. Khatir, S. Tiachacht, C. Le Thanh, E. Ghandourah, S. Mirjalili, and M. Abdel Wahab, "An improved Artificial Neural Network using Arithmetic Optimization Algorithm for damage assessment in FGM composite plates," *Compos. Struct.*, vol. 273, no. June, p. 114287, 2021, doi: 10.1016/j.compstruct.2021.114287.

[25] M. Jamil and X. S. Yang, "A literature survey of benchmark functions for global optimisation problems," *Int. J. Math. Model. Numer. Optim.*, vol. 4, no. 2, pp. 150–194, 2013, doi: 10.1504/IJMMNO.2013.055204.

[26] S. J. Bhat and K. V. Santhosh, "A localization and deployment model for wireless sensor networks using arithmetic optimization algorithm," *Peer-to-Peer Netw. Appl.*, vol. 15, no. 3, pp. 1473–1485, 2022, doi: 10.1007/s12083-022-01302-x.

[27] J. O. Agushaka and A. E. Ezugwu, "Advanced arithmetic optimization algorithm for solving mechanical engineering design problems," *PLoS One*, vol. 16, no. 8 August, pp. 1–29, 2021, doi: 10.1371/journal.pone.0255703.

[28] A. Karimiafshar, M. R. Hashemi, M. R. Heidarpour, and A. N. Toosi, "Effective Utilization of Renewable Energy Sources in Fog Computing Environment via Frequency and Modulation Level Scaling," vol. 4662, no. c, pp. 1–9, 2020, doi: 10.1109/JIOT.2020.2993276.

https://doi.org/10.5281/zenodo.21095167