Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Joint Optimization of Computation Offloading and Task Scheduling using Multi-objective Arithmetic Optimization Algorithm in Cloud-Fog Computing

Asad Ali [a], Nazia Azim [b], Mohamed Tahar Ben Othman [c,*], (Senior Member IEEE), Ateeq Ur Rehman [d,*], (Senior Member IEEE), Masoud Alajmi [e], Mosleh Hmoud Al-Adhaileh [f], Faheem Ullah Khan [g], Mamyrbayev Orken [h], and Habib Hamam [I,j,k,l] , (Senior Member IEEE)

[a] Department of Computer Science, Mardan Institute of Science and Technology, Mardan 23200, Pakistan; asad.ali@uetpeshawar.edu.pk
[b] Department of Computer Science, Abdul Wali Khan University Mardan, Mardan 23200, Pakistan; n.azim@awkum.edu.pk
[c*] Department of Computer Science, College of Computer, Qassim University, Buraydah 51452, Saudi Arabia; maathaman@qu.edu.sa
[d*] School of Computing, Gachon University, Seongnam 13120, Republic of Korea; 202411144@gachon.ac.kr
[e] Department of Computer Engineering, College of Computers and IT, Taif University, Taif 21944, Saudi Arabia; ms.alajmi@tu.edu.sa
[f] Department of E learning and IT, King Faisal University, 31982 Al-Ahsa, Saudi Arabia; madaileh@kfu.edu.sa
[g] Department of Software Engineering, University of Science and Technology, Bannu, Pakistan; Cise42@gmail.com
[h] Department of Computer Science, Institute of Information and Computational Technologies, Almati, Kazakhstan; morkenj@mail.ru
[I] Faculty of Engineering, Uni de Moncton, Moncton, NB, E1A3E9, Canada; habib.hamam@umoncton.ca
[j] Department of Electrical and Electronic Engineering Science, School of Electrical Engineering, University of Johannesburg, Johannesburg 2006, South Africa
[k] Hodmas University College, Taleh Area, Mogadishu, Somalia
[l] Bridges for Academic Excellence, Tunis 1002, Centre Ville, Tunisia

Corresponding author: Mohamed Tahar Ben Othman (maathaman@qu.edu.sa) and Ateeq Ur Rehman (e-mail: 202411144@gachon.ac.kr).

ABSTRACT The exponential increase in the Internet of Things (IoT) has affected the cloud computing with increase transmission latency and network overhead for real-time applications. Cloud-fog computing paradigm tackle these limitations by moving computational services closer to the network edge i.e., fog nodes, enhancing the speed of real-time applications. This architecture, with its dynamic computing environment and diverse IoT devices and tasks, demands a reliable and energy-efficient communication network. Joint optimization of computation offloading and task scheduling is a primary challenge, as it involves offloading tasks to optimal computational resources and scheduling them in an efficient order for operational efficacy. While offloading tasks to fog nodes reduces delay but raises energy utilization, offloading them to cloud servers reduces energy usage but raises computational costs and latency. Additionally, inefficient order of task execution (executing lower priority jobs before higher priority tasks) can disrupt system stability and reliability. Therefore, an effective joint optimal computation offloading and task scheduling strategy is essential. To this end, we propose a Multi-objective Arithmetic Optimization-based joint computation offloading and task scheduling algorithm, aiming to minimize energy consumption and transmission latency. Extensive simulations in MATLAB demonstrate the efficacy of the proposed algorithm in terms of designated optimization objectives.

INDEX TERMS— Task offloading, computation offloading, optimizing algorithms, cloud-fog computing, Arithmetic Optimization

I. INTRODUCTION

The revolution of Internet of Things (IoT) has been greatly affected by the growth of telecommunication networks. Telecommunication devices are continuously generating large amounts of data which may require an instant response. Due to the limited processing capabilities of the end-user devices, data is forwarded to the cloud servers for processing, storing, and analyzing [1]

Cloud computing is a modern technology that offers scalable, flexible, reliable and dynamic computing infrastructure for various applications. It provides virtualized resources (e.g., computations, memory and storage) as a service to organizations and individuals via internet. Cloud computing architecture consist of front-end interface and back-end interface. The front-end interface contain users, mobile devices and application that requires services (e.g., processing, memory and storage) from the cloud servers. The client can access the cloud computing services using front-end interface. The back-end interface contain web servers, storage devices, network management policies, data security mechanism, load balancing strategies etc.[2],[3]

Although cloud computing can offers a plethora of resources, however, it cannot effectively meet the requirements of delay-sensitive and data-massive IoT applications. To address these limitations, cloud-fog computing has emerged as a cutting-edge hybrid computing paradigm that aims to process the diverse end user requests having different needs and requirements like bandwidth utilization, cost, delay, energy consumption and makespan etc [4]. This strategy integrates the cloud servers and fog nodes (located near to the network edge) for processing of heterogeneous tasks, optimizing offloading delay, energy and bandwidth utilization for improved operational efficacy.

However, due to the constrained wireless resources, dynamic and intricate tasks setups, and enormous heterogeneous end-user requests present significant challenges in computation offloading and task scheduling within the cloud-fog computing, necessitating the developments of effective strategies to manage these complexities for operational efficacy[5-7]. Computation offloading entails directing jobs to the appropriate computing resource based on task requirements while task scheduling ensures that the jobs are executed in a manner to optimize different performance metrics like delay, energy, workload, bandwidth, cost and makespan. Joint optimization of computation offloading and task scheduling is crucial to escalate resource utilization, significant saving of energy consumption, prevents resource bottlenecks, reduce communication and offloading latency, and improve overall system stability and reliability.

Computation offloading and task scheduling entails an effective strategy due to trade-offs involved in offloading end-user jobs to computational nodes (cloud servers and fog nodes). While offloading tasks to fog nodes reduces delay but might raises energy utilization, offloading them to cloud servers reduces energy usage but raises computational cost and latency. Additionally, in the realm of task scheduling, executing tasks in a less efficient sequence (e.g., executing low priority jobs before higher priority tasks) can disrupt overall system stability and reliability. For instance, in a smart city infrastructure, the traffic management system depends on timely data from IoT devices to take prompt action regarding traffic lights and congestion. If the management system is overloaded with non-traffic jobs, and unable to handle real-time traffic related tasks, this could cause congestion and increase the ratio of accidents. Thereby, effecting the stability and efficiency of the smart city. Therefore an effective joint optimal computation offloading and task scheduling strategy is required to find an efficient trade-off between energy efficiency and delay reduction. Research has demonstrated that achieving a balanced workload allocation between the computational nodes (fog node and cloud server) can greatly escalate the network performance.

Joint optimization of computation offloading and task scheduling is recognized as multi-objective NP-complete optimization problem[8]. There exist various techniques for identifying optimal strategy within a cloud-fog computing network like brute force algorithms and dynamic programming. Nevertheless, these algorithms are not always efficient in obtaining optimal offloading and scheduling strategy. For instance, the dynamic programming approach is not efficient when dealing with enormous tasks and resources due to exponential increase in the sub problems, making it computational intensive and impractical for delay sensitive IoT applications.

In recent years, many nature-inspired algorithms are utilized to address the complexities of computation offloading and task scheduling in cloud-fog computing like Genetic Algorithm (GA)[9], Ant Colony Optimization (ACO)[10], Particle Swarm Optimization (PSO)[11], Gray Wolf Optimization (GWO)[12] and Harris Hawks Optimization (HHO)[13]. Their objectives is to find an optimal job schedule to optimize designated performance metrics (e.g., delay, energy, throughput, makespan and cost etc.). Nevertheless, these algorithms suffer from various challenges such as inefficient use of randomized operators, computational intensive operations, and inefficient trade-off between intensification and diversification phases, resulting in local stagnation and reducing system efficiency.

Given that task offloading and scheduling is a multi-objective optimization problem, we utilize Multi-objective Arithmetic Optimization Algorithm (MoAOA) as a mathematical model to optimize computation offloading and task scheduling with an objective to reduce energy consumption and offloading latency. AOA has proven its effectiveness in tackling various optimization problems [14]. Its rich repertoire of randomize operators and their efficient utilization enable the algorithm to equalize different search strategies during both the exploration and exploitation phase. Furthermore, the strategic design of control variable allow candidate solution to engage in exploration not just at the beginning of the iteration but also towards the end of the iterations. This capability help the algorithm in preventing local optima solution.

Our approach stands out due to its utilization of MoAOA to address the multi-objective problem of task offloading and scheduling in this domain. To the best of our knowledge, no existing approach tackle this optimization problem using MoAOA. The search process, based on AOA's basic operators, offers simplicity and computational efficiency which are particularly important in cloud-fog environment where real-time decision-making is necessary. The proposed algorithm efficiently handle the heterogonous IoT tasks by offloading computational intensive tasks to servers and delay-sensitive tasks to nearby fog devices. Furthermore, it prioritize tasks ensuring higher priority tasks are executed before lower priority ones to maintain system stability, reliability and responsiveness.

In the proposed approach, the search solutions are initialized in the problem space along with the initialization of prerequisites (e.g., IoT devices, fog nodes, MOA function and cloud servers). Each search solution represent a unique strategy of task offloading and scheduling which is evaluated using the optimization function. This phase identifies the best search solution with minimum energy consumption and offloading delay. The subsequent solutions, converge towards the best solution by repositioning search strategy using random variables and MOA function value. Upon reaching the stopping criteria, the optimal search solution obtained so for contains the optimal strategy of task offloading and scheduling.

The experiments are performed in MATLAB and the results are analyzed against the similar comparative algorithms like MoGWO [12], Cloud-fog cooperation algorithm [15], Ant Colony Optimization (ACO) [16], and Genetic Algorithm (GA) [9] . The results demonstrate the efficacy of the proposed approach in terms minimum energy consumption and offloading latency. Moreover, the rich repertoire of stochastic variables and strategic design of control parameter allow the proposed algorithm to effectively distribute the workload across different computational nodes. This ability improves network stability and reliability. Thus the computing network is able to handle massive range of heterogeneous tasks from IoT devices.

The novelty and contributions of the paper are given below:

- We integrate the multi-objective optimization capabilities into AOA, unlocking the potential for simultaneous optimization of offloading latency and energy consumption.
- We design a multi-objective fitness function that simultaneously optimizes designated objectives while considering the priority of tasks to enhance system stability and efficiency. The multi-objective function generate better Pareto-front, offering diverse set of optimal solutions.
- We designed a rigorous mathematical framework for MoAOA, providing a solid foundation for joint optimization of task offloading and scheduling.
- We pioneer an adaptive computation offloading and task scheduling capabilities that empower the algorithm to dynamically respond to heterogeneous tasks of different requirement, ensuring optimal resource utilization and performance.
- We introduced a collaborative task allocation scheme, that intelligently distribute tasks between fog nodes and cloud servers, maximizing system efficiency, minimizing offloading delay, energy consumption and enhancing overall system productivity.
- We strategically designed control variable to perform random search strategies at each iteration to prevent local optima solution.
- A comprehensive numerical analysis is performed with the similar competitors to demonstrate the efficacy of the proposed scheme.

The road map of the paper is given as follows: section 2 comprehensively analyze the literature review, section 3 contains the preliminaries i.e., introduced the basic architecture of cloud fog computing network, the difference between AOA and MoAOA and contain working operation of the proposed algorithm. Section 4 contain the proposed solution, section 5 discuss the simulation results and section 6 conclude the paper with future direction.

II. LITERATURE REVIEW

In the literature, several studies have addressed the problem of computation offloading and task scheduling in a cloud-fog computing environment. Computation offloading is a primary concern of the research, particularly, in optimizing various Quality of Service (QoS) parameters such as energy consumption and communication delays. Numerous optimization algorithms are designed for efficient task offloading and resource allocation in cloud-fog computing[17],[18].Mainly the algorithms are categorized into four groups i.e., Mathematical Programming, Machine Learning, Heuristic, and Meta-heuristic. The taxonomy of the task offloading algorithm is given in Fig. 1.

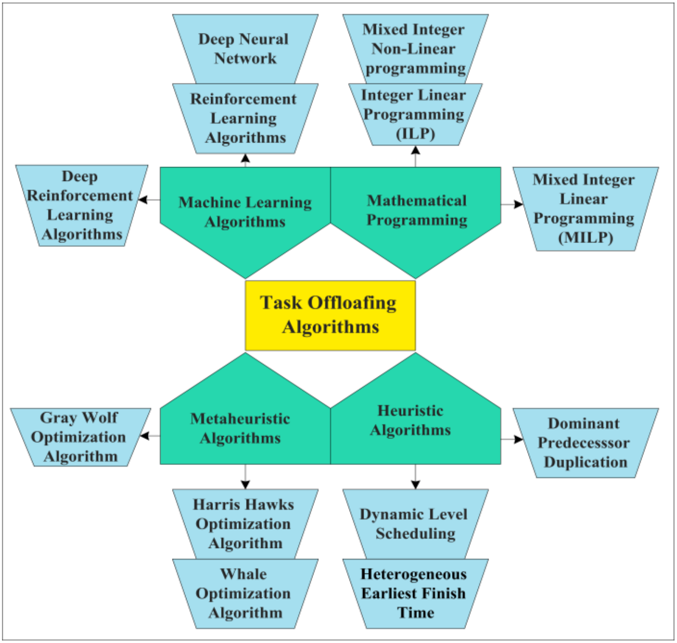

Deep Neural Network
Mixed Integer Non-Linear programming
Reinforcement Learning Algorithms
Integer Linear Programming (ILP)
Deep Reinforcement Learning Algorithms
Machine Learning Algorithms
Mathematical Programming
Mixed Integer Linear Programming (MILP)
Task Offloading Algorithms
Gray Wolf Optimization Algorithm
Metaheuristic Algorithms
Heuristic Algorithms
Dominant Predecesssor Duplication
Harris Hawks Optimization Algorithm
Dynamic Level Scheduling
Whale Optimization Algorithm
Heterogeneous Earliest Finish Time

Figure 1: Taxonomy of task offloading algorithm

A. MATHEMATICAL PROGRAMMING

Mathematical Programming (MP) is an optimization branch used for real-world complex optimization problems by formulating them as mathematical models with decision variables, fitness function, and constraints[19, 20]. Various MP techniques such as linear programming[21], mixed integer linear programming[22], mixed integer non-linear programming[23], and quadratic programming [24] etc. are applied, specifically to address resource allocation in cloud-fog computing. Nevertheless, exploring the entire solution space for optimal solutions may increase the computational complexity of the algorithm. Moreover, they lack in scalability when dealing with heterogeneous tasks and computational resources.

Optimal task offloading and resource allocation for fog computing is proposed in [25]. The mathematical model of the joint optimization problem is formulated using mixed integer non-linear programming (MINLP). They introduced a relaxing solution that converts the binary decision value into the real value to return the most optimal solution. The tasks are offloaded to any computational nodes i.e., for nodes or cloud servers to reduce energy consumption while meeting the requirements of delay-sensitive tasks.

Daneshfar et al. [26] proposed an ILP-based service allocation algorithm for fog computing with the objective to minimize the cost of service allocation. The tasks generated from heterogeneous IoT devices are multi-casted to multiple computational nodes to ensure the availability of computational resources. .

B. MACHINE LEARNING

Machine Learning (ML) techniques also have been widely used to tackle computational offloading, task scheduling, and resource management in fog computing. There are numerous techniques of ML such as reinforcement learning, fuzzy logic, neural networks, and Bayesian networks. Among others, reinforcement learning and its variants (e.g., deep reinforcement learning) are the most widely used techniques to apply to resource management in cloud-fog systems. This technique interacts with the unknown environment to learn the policy to obtain a trade-off between diversification and intensification [27, 28] .

Tang et al. [28] proposed a deep reinforcement learning-based task offloading in fog computing. The focus of the study is to tackle computation offloading and service placement in fog computing with the objective of minimizing latency, migration cost, and energy consumption. To this end, the optimization problem is formulated as a multi-dimensional Markov decision process.

Q-learning-based fragmented task offloading in fog computing is proposed by Razaq et al. [29]. The goal is to offload the tasks to the computational node while ensuring load balancing. Incoming IoT tasks are partitioned into segments based on their privacy level, completion time, and other real-time requirements, the segments are offloaded to multiple fog nodes.

A deep neural network (DNN) based distributed computation offloading in mobile fog computing is proposed in [30]. The algorithm generates multiple offloading decisions using parallel DNN. The performance of the DNN is improved using the back-propagation method where cross-entropy is used as a loss function.

C. HEURISTIC

Heuristics are another set of methods, widely used to solve computationally expensive problems in an attempt to get a feasible solution in a reasonable amount of time. However, they may not be able to provide feasible solutions in some cases as described in [19]. The shortcoming of most of the heuristic techniques is local stagnation, providing a locally optimal solution that results in a lack of efficiency. It uses pre-defined rules to allocate tasks to the computational node. Therefore, heuristic-based algorithms are more prone to dynamic network.

The most common heuristic-based task offloading algorithms are Dynamic Level Scheduling (DLS) [31], Heterogeneous Earliest Finish Time (HEFT) [32], and Dominant Predecessor Duplication (DPD) [33].

HEFT is a task-offloading algorithm specifically designed for heterogeneous computational resources. In this algorithm, each incoming task is assigned a priority value based on its finishing time. Tasks with the shortest finishing times are given high priority values and vice versa.

In [34], the authors proposed a heuristic computation offloading algorithm in fog computing. The algorithm solves the computation offloading problem using a two-stage method to minimize energy consumption and communication costs.

A clustering-based task offloading algorithm is proposed in [35]. The authors divide the incoming tasks into clusters based on task similarities (e.g., completion time, memory, etc). Tasks with the same requirements are grouped in one cluster and then assigned to the computation node.

D. METAHEURISTIC

Metaheuristic algorithms are mostly inspired by the biological evolution observed in nature such as food searching maneuvers of birds and ants, the hunting strategies of animals, and human social behavior, etc. It is a very challenging task to get the global optimal solutions for multi-objective NP-complex optimization problems[36, 37]. Metaheuristics are designed with a framework that incorporates more stochastic operators and has a dynamic and versatile nature that enables them to get global optimal solutions. Due to random variables, search agents can explore multiple solution spaces simultaneously. These features enable the algorithm to tackle computation offloading in computing networks [38].

Genetic Algorithm (GA) is a primary method to tackle task offloading in fog computing to enhance system efficiency and resource utilization [9]. GA optimizes task offloading by initializing a set of chromosomes, representing tasks and their allocation to the computation node. Another GA-based task offloading algorithm for fog computing is proposed in [39]. The objective is to optimize communication and computation cost and makespan.

Ant Colony Optimization (ACO) based task offloading for IoT applications is proposed by Hussein et al. [16]. The objective is to optimize response time and load balancing. The focus of the study is to distribute the delay-sensitive IoT task to computational nodes while meeting the QoS requirements.

Particle Swarm Optimization-based task offloading is proposed in [40], with minimum energy consumption, low communication latency, and minimum execution cost.

TABLE I

SUMMARY OF WELL-KNOWN COMPUTATION OFFLOADING ALGORITHMS

Ref	Methods	Application	Objectives	Advantage	Limitation
[41]	Metaheuristic	General	Minimization of task loss probability and energy consumption	Improves resource utilization	High complexity
[42]	Metaheuristic	IoT, smart cities and healthcare	Minimize execution time and energy utilization of user device	Enhanced user experience and improves resource utilization	high complexity and network overhead
[43]	Metaheuristic	Smart cities	Minimize execution delay, energy consumption and charging cost of used resources	Improves resource utilization and QoS.	Less scalable and high network overhead

[44]	Metaheuristic	IoT , Fog computing	Minimize makespan cost, execution time and energy utilization	Improves resource utilization	Less scalable and high complexity
[45]	Mathematical Programming	General	Minimize latency	Achieves shorter round trip times and makespan	High complexity
[26]	Mathematical Programming	General	Optimize deployment cost	Reduce computational complexity	Does not consider latency in problem formulation
[46]	Mathematical Programming	Image Processing	Minimization of transmission delay	Reduce computation complexity and network overhead	Service processing rate is unrealistic
[47]	Mathematical Programming	Healthcare, smart cities	Minimization of energy consumption and latency	Maximize resource utilization and improves QoS	High computational cost and less scalable
[48]	Machine Learning	Tracking	Optimization of migration cost, energy consumption and transmission delay	High scalability due to consideration of mobility and multi-objective optimization	High computational complexity and network overhead
[49]	Machine Learning	Mobile crowd sensing	Quality of Service	More processing is perform on fog nodes	High computational complexity
[50]	Machine Learning	General	Quality of Service	Improve Quality of service	High training time
[51]	Heuristic	Smart cities	Optimizing delay	High scalability and reduces processing time	Privacy of confidential data is compromised
[52]	Heuristic	Smart cities, healthcare	Optimizing latency	High scalability	No realistic simulation
[53]	Heuristic	Healthcare	Optimizing energy consumption	Improve resource utilization	Mobility is not taken into account
[54]	Heuristics	Smart homes	Optimizing power consumption	Achieved trade-off between energy consumption and computational complexity	Not suitable for real-case scenarios

In [55], the authors utilized the moth-flame optimization (MFO) algorithm to efficiently allocate a set of tasks to computational nodes while ensuring QoS requirements like communication cost latency and energy consumption.

In [13], the authors employ the hunting strategy of hawks to optimize task scheduling in cloud-fog computing. The objective of the multi-objective optimization is to obtain a trade-off between communication latency and energy consumption.

Gray wolf optimization-based task offloading and resource allocation algorithm is proposed in [12]. The optimization objectives are energy consumption and communication latency while ensuring efficient workload distribution and maximizing resource utilization. Tasks are allocated based on their requirements. [56, 57], also utilized GWO to schedule the tasks to computational nodes in a computing network.

Although, numerous task offloading and scheduling algorithms have been proposed to optimize the performance of fog computing. Nevertheless, most of them are single objectives, focusing only on either energy consumption or latency and execution cost. Efficient performance of the computing environment is based on multiple objectives and should consider both optimization objectives (energy utilization and delay). Additionally, the baseline comparative algorithm i.e., Cloud-fog cooperation algorithm [15] uses heuristic approach to enhance the performance of the system by considering predetermined delay threshold and queuing theory. The algorithm employs static task allocation method and does not address the dynamic computing environment with heterogeneous IoT devices and tasks. This may not be adaptable to real-time applications. Another baseline comparative algorithm, MoGWO [12] utilizes the mathematical framework of GWO algorithm efficiently allocate task to computing resource with the objective to optimize energy utilization and communication latency of the computing network. However, the algorithm does not address the execution order of allocated tasks. This may result in the execution of low priority tasks before higher priority jobs. Consequently, the system efficiency is reduced by obtaining suboptimal solution in terms of designated objectives. Furthermore, the existing meta-heuristic based offloading strategies fail to achieve a balanced trade-off between multiple objectives due to the imbalance operation of repositioning search phases, local stagnation, and less stochastic variables, resulting in performance degradation. This provides an opportunity to improve the performance of computing by obtaining an efficient trade-off between multiple objectives i.e., delay and energy utilization. Table I contains the summary of the well-known computational offloading algorithms.

III. PRELIMINARIES

In this section, we present an inspiration of AOA along with the difference with its multi-objective variant i.e., MoAOA. Furthermore, we discuss the fundamental principal of cloud-fog computing architecture and the working operation of the proposed solution.

A. CLOUD-FOG COMPUTING ARCHITECTURE

Numerous architectures for cloud-fog computing have been proposed but mostly consist of a three-tier structure i.e., 1) IoT tier; 2) fog tier; and 3) cloud tier. The tiers are connected via wireless networks such as Wi-Fi, Bluetooth and LoRa, etc. The IoT tier is the lowest layer and consists of mobile devices incorporated with sensors to collect data from the environment and send it to upper tiers for further processing. IoT devices may generate delay-sensitive tasks that require immediate response and computational-intensive jobs that require heavy computational resources for processing. The generated task is forwarded to the fog layer which is an intermediate tier between the IoT layer and the cloud layer. The intermediate tier consists of fog device *FD* and fog controller *FC*. *FD* is a node with less storage capacity, communication capability, and computation power to process relayed tasks, thereby minimizing network overhead transmission delay. For jobs that demand high computational power, more storage capacity, and high communication capability, fog nodes forward these tasks to the cloud data centers. *FC* is a specialized node that is responsible for offloading, analyzing, estimating, and scheduling tasks by deliberating factors such as processing time and energy utilization. The proposed algorithm is installed on *FC* to determine an optimal offloading decision based on optimization objectives.

For computational-intensive tasks, *FC* forwards jobs to the cloud tier which composed of multiple computational servers with high computation power and storage capacity for execution. After executing a task, the results are sent back to *FC* to aggregate the results and then send a response to IoT devices.

B. AOA AND MOAOA

The AOA is a mathematic-inspired swarm intelligence optimization algorithm. Its inspiration is taken from the distributive characteristics of arithmetic operators (e.g.,) that guide the search solutions in diversification (exploration) and intensification (exploitation) phases via math optimizer accelerated function (MOA) [14]. The search process of AOA starts with the random initialization of search solutions, followed by exploration and exploitation search phases. Exploration behavior globally investigates the problem space in search of potential feasible solutions. The exploitation phase involves investigating the obtained feasible locations in pursuit of approaching the target optimal area.

AOA is computationally feasible and has a straightforward implementation, this ability attracts researchers to tackle a wide range of complex optimization problems like unimodal , multimodal, linear, and non-linear optimization problems not only in the field of computer science but also in medical science, engineering, electronics, and network security, etc. Multi-objective AOA (MoAOA) is an alternative to AOA for solving multi-objective optimization problems [58]. MoAOA generates multiple solutions for multiple objectives, which are then stored in an external archive as non-dominated solutions. Non-dominated solutions are called Pareto optimal solutions and form the Pareto front, representing a trade-off relationship between designated optimization objectives. At the end of each iteration, the solution that meets the requirements of the fitness function is selected as the best optimal solution, followed by a position updating mechanism around the best solution obtained thus far.

Deterministic approaches are inefficient to solve multi-objective optimization problems as they often generate a single dominant solution, and are likely to get trapped in local optima. Computation offloading and task scheduling within cloud-fog computing is also a multi-objective problem, therefore we utilize MoAOA to optimize multiple objectives i.e., energy and latency.

C. WORKFLOW OF PROPOSED ALGORITHM

When IoT devices generate tasks to be processed, at first the tasks are sent to the F_D (step.1). F_D forward the tasks to the F_C to analyze the requirements of each task and then assigns a priority to each received task (step.2-6). Afterward, the F_C invokes the proposed algorithm to craft a binary offloading matrix that indicates offloading of received tasks to computational devices along with the schedule of tasks on assigned computational devices (step.7-8). The dimension of the matrix represents the number of search solution and the count of computational resources. The optimization function evaluates the search solution, followed by repositioning the remaining search solutions around the optimal solution. The best solution contains an optimal strategy of task offloading and scheduling on computational resources. The tasks are then assigned and executed according to the optimal strategy (step.9-10). After the execution of tasks, the results are sent back to the IoT devices (step 11-12). The working operation of the proposed algorithm is given in Fig. 2.

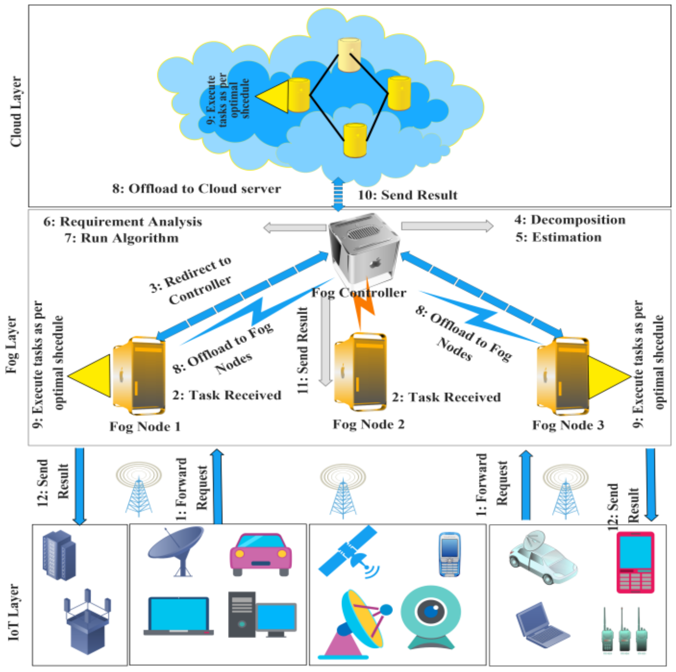


Figure 2. Working operation of proposed algorithm

IV. THE PROPOSED SOLUTION

In this section, we present details regarding the proposed solution to tackle joint optimization of task offloading and scheduling using a multi-objective arithmetic optimization algorithm. Our objective in the proposed algorithm is twofold, the first is to offload *Tn* independent tasks to different computational nodes (fog node or cloud server) by considering task requirements such as computational time, computation energy, and data size, and the second objective is to schedule the offloaded tasks on each assigned computational nodes (fog node or cloud server) according to the tasks requirements. If tasks are offloaded to the fog node then the proposed algorithm schedules the tasks according to the computational time, and in another case, the offloaded tasks are scheduled by considering the computational energy of each task. The proposed algorithm aims to execute tasks with minimal completion time, and energy consumption and balance the workload on each computational node. The proposed optimization algorithm encompasses five phases i.e., i) search space initialization ii) solution creation iii) solution evaluation using objective function iv) update location vector, and v) termination criteria. The symbols used in formulas and algorithms are described in Table II.

A. SYSTEM MODEL

The system model consist of fog nodes represented as $FN = \{FN_1, FN_2, FN_3, \ldots, FN_n\}$, heterogeneous cloud servers denoted as $CS = \{CS_1, CS_2, CS_3 \ldots, CS_n\}$, IoT devices represented as $ED = \{ED_1, ED_2, ED_3 \ldots ED_n\}$, number of search agents denoted as $SA = \{SA_1, SA_2, SA_3, \ldots SA_n,\}$ and number of tasks i.e., $T = \{T_1, T_2, T_3 \ldots \ldots T_n\}$. We make an assumption that the energy consumption E_{ED}^{CPU} and CPU capacity (C_{ED}^{CPU}) of *ED* is less than the *FN* (C_{FN}^{CPU},E_{FN}^{CPU}) and *CS* (C_{CS}^{CPU},E_{CS}^{CPU}), similarly the energy consumption and CPU capacity of *FN* is less than *CS*. Mathematically, the CPU capacity and energy consumption of each computational node respectively, can be expressed as $C_{ED}^{CPU} < C_{FN}^{CPU} < C_{CS}^{CPU}$ and $E_{ED}^{CPU} < E_{FN}^{CPU} < E_{CS}^{CPU}$. The offloading and scheduling decision of each incoming tasks is performed by the controller. The controller assigns priority to each incoming task based on their importance, required computational time and deadline.

The search agents are the candidate solutions that contain different strategies for offloading and scheduling tasks. The candidate solutions are evaluated with the optimization function to obtain the best candidate solution. Subsequently, the remaining candidate solutions change their location vector by employing the exploration and exploitation phase to come closer to the global optimal solution.

B. CREATION OF PROBLEM SPACE

In the proposed solution, at first, we create a problem space with the random initialization of the search solution along with the initialization of IoT devices, fog devices, and cloud servers. Each search solution investigates the problem space for an optimal solution using a Math Optimizer Accelerated (MOA) function. MOA is a function that aids the proposed algorithm in reaching the global optimal solution by employing different search strategies. In each iteration, the value of MOA is updated to maintain an equilibrium state between the search strategies i.e., exploration and exploitation search phase. MOA is calculated as follows:

$$MOA(itr_c) = V_{min} + itr_c \times \left(\frac{V_{max} - V_{min}}{itr_m}\right) \tag{1}$$

Where $MOA(itr_c)$ indicate the function value at current iteration. itr_c and itr_m represent the current and maximum iteration respectively, V_{max} and V_{min} represents the maximum and minimum vale of the function respectively.

C. SOLUTION CONSTRUCTION

Each search solution is encoded as a binary offloading matrix of T x C dimension where T denotes the count of independent tasks and C represents the computational node i.e., for the device or cloud server. The binary matrix indicates the offloading of incoming tasks to computational devices along with the schedule of tasks on assigned computational devices. The computational intensive tasks are offloaded to resource-rich computational nodes i.e., cloud servers while the delay-sensitive tasks are offloaded to fog nodes. Let *A(i , j)* refers to the allocation of i^{th} task to j^{th} computational node. Then we can expressed *A(i , j)* as follows:

$$A(i\,,j) = \begin{cases} 1\,, \text{i}^{\text{th}}\text{task to j}^{\text{th}}\text{ computational device} \\ 0\,, \text{Task is not yet assign to computational device} \end{cases} \tag{2}$$

The offloading of tasks crafted by search solutions deliberates factors such as the availability of computing resources, maximum resources of the computing node, offloading objectives, task requirements, and network conditions. The controller offload

TABLE II
LIST OF SYMBOLS

Notations	Meaning
Π_{Delay}	Total offloading delay
$Ʒ_{Energy}$	Total energy consumption
W	Weight factor
ζ_L	Input length of the task
μ	Service rate of IoT device
$\hbar$	Length of the task queue
A_{COTS}	Computation offloading and task scheduling matrix
$\mathfrak{p}_{IOT}^{i}$	Power consumption of IoT device
W_L	Workload
$Comp_{Node}$	Computing node
η	Task arrival rate
$\hbar$	Average length of a queue
Ψ_{IoT}^{i}	Task’s processing duration
M_{CS}^{k}	Count of virtual machines server k
Itr_N	Next iteration
U_{b_j}	Upper bound value
Itr_C	Current iteration
U_{b_j}	Lower bound value
Itr_M	Maximum iteration
$Ϙ_k$	Count of ON virtual machines
$Fitness_{Matrix}$	Fitness matrix of all candidate solution
$BSol_{Fitness}$	Fitness value f best solution
$OSol_{Fitness}$	Fitness value of best solution in current iteration
C_P	Control parameter
a	Escalation parameter
$ɣ_k$	ON/OFF state of virtual machine on server k

ϖ	Assigned workload
c_1, c_2, ι	constants

the task to a computing device only if the required resources of the task are less than the available capacity of the computing device. Moreover, while offloading a task, the search solution must ensure that the computational node is not overloaded and the workload is balanced across all computational nodes.

The offloaded tasks are scheduled on assigned computational nodes according to the assigned priority. The priority of tasks is based on their importance and deadline. A task with the shorter deadline has high priority and must be executed before the task with the higher deadline. To craft a solution, the candidate solutions must adhere to the following constraints:

- The offloading must not overload the computational node.
- The workload across all computational devices must be balanced
- The task must not be partitioned into sub tasks and must be assigned as a whole to computing node.

The pseudo code of the solution construction phase is given in Table III.

TABLE III
SOLUTION CONSTRUCTION

```
Algorithm 1. Binary Offloading Matrix
Function $A_{(COTS)}$= OffloadingMatrix(($T$, $Comp_{Node}$, $Search_{Agent}$)
1.  Initialize the workload of all computational node to zero i.e., $W_L$ ($Comp_{Node}$) = 0
2.  For ($i$=1→number of search agents) do {
3.      i). Determine task requirements i.e., task type, required
4.          memory, processing time, computation resources
5.          and task's deadline etc.
6.      ii). Determine the capacity of computing node (fog device
7.          or cloud
            server)
8.      For (j=1→number of tasks)
9.          If ( $T_j$ == Delay_Senstive do {
10.             i) Select k^th fog device
11.                 if ($W_L$ ($FN_k$) > $Cap_{Max}$) do {
12.                     i).  $FN_k = T_j$
13.                     ii).  $W_L(FN_k) = W_L(FN_k)+1$
14.                 else
15.                     i).  Find next suitable $FN_k$
16.                 End if  }
17.         Elseif ( $T_j$ == Computation_Intensive do {
18.             Select k^th cloud server
19.                 if ($W_L$ ($CS_k$) > $Cap_{Max}$) do {
20.                     i).  $CS_k = T_j$
21.                     ii).  $W_L(CS_k) = W_L(CS_k)+1$
22.                 else
23.                     i).  Find next suitable $CS_k$
24.                 End if  }
25.         End if  }
26.         i). Determine the priority of each offloaded task
                based on their importance and deadline, and sort
                in ascending order
27.         For (j=1→number of tasks)
28.             Schedule each offloaded task as per its priority level
29.         End
30.     End
31. End For
32. End Function
```

D. OPTIMIZATION FUNCTION

Search solutions are evaluated using the optimization function. The optimization function plays a substantial role in selecting the most suitable and optimal computational device for offloading decisions based on objectives. Each candidate solution employs a fitness function to compute the objective values. We use a multi-objective function to achieve the aim of minimizing both offloading delay and energy consumption, given the nature of our study as a Multi-objective Problem (MOP). We cannot compare the individual solutions via the optimization function, the optimal solution can be obtained using Pareto optimal front. The multi-objective optimization function is given below:

$$OF_{Fitness} = W * \Pi_{Delay} + (1 - W) * \text{Ʒ}_{Energy} \tag{3}$$

Where Ʒ_{Energy} and Π_{Delay} are total energy utilization and offloading delay respectively, W ∈ {0 , 1} is the weight factors that determine the priority of each objective. Here we assume an equal priority for each objective i.e., an optimization algorithm gives equal importance to each objective during the optimization process.

To compute the fitness value of search solutions, we need to calculate the offloading delay and energy consumption at each level of devices i.e., IoT devices, fog devices, and cloud servers.

1) DELAY MODEL

In this section, we explain the calculations of delay at different levels of devices. IoT devices employ an M/M/1 queueing system to process incoming tasks. The tasks are exponentially distributed across the system as per the defined service rate. Additionally, task arrival from IoT devices adheres to the Poisson process. The delay of the IoT device is calculated as follows:

$$D_{IoT}^{i} = \frac{\eta}{\mu(\mu - \eta)} \tag{4}$$

Where η is the arrival rate of each task generated by the IoT device, μ is the service rate of IoT device. Eq. 4compute delay a task experiences in the queue when there is congestion in the system. Here, the arrival rate of task i.e., η is the controllable variable. This means the computing network can control the number of offloaded tasks to computational resources (i.e., fog node or cloud server).By adjusting the number of η, the system can optimize the latency. For instance, managing the workload across all devices by controlling η is necessary to minimize entire system delay.

The fog device j is modeled to have an M/M/C task queueing system. Since the fog node is a computational node responsible for computing tasks, therefore the offloading delay experienced at the fog node can be expressed as a combination of communication and computation delay. The computation latency can be expressed as follows:

$$D_{FD_{(comp)}}^{j} = \left(\frac{\hbar}{\eta}\right) . \varpi_{FD_{(comp)}}^{j} \tag{5}$$

Where ϖ and $\hbar$ are the assigned workload to FD j and average length of the queue respectively. The communication latency experienced at fog node j can be expressed as follows:

$$D^{j}_{FD_{(comm)}} = c_1 * \zeta_L(T) \tag{6}$$

Where c_1 is constant and ζ_L denotes the input length of the task T. The communication delay of fog node is related to the input length of the task. So the total offloading delay experienced at fog node j can be calculated as follows:

$$D^{j}_{FD} = D^{j}_{FD_{(comp)}} + D^{j}_{FD_{(comm)}} \tag{7}$$

Similarly, the cloud server k is modeled with an M/M/∞ task queue, indicating an infinite capacity. Given the substantial computational capacity of the server, the computation delay is negligible. So communication delay experienced at the cloud server is expressed as follows:

$$D^{k}_{CS_{(comm)}} = c_2 * \zeta_L(k) \tag{8}$$

Where c_2 is a constant and $\zeta_L(k)$ indicate the input length of the cloud server k. Now we can expressed the total delay as a summation of all delays experienced at different level of devices.

$$\Pi_{Delay} = D^{i}_{IoT} + D^{j}_{FD} + D^{k}_{CS_{(comm)}} \tag{9}$$

2) ENERGY CONSUMPTION MODEL

In this section, we discussed the calculation of the energy consumption of each device across different computational devices. Since the task's processing duration of each device is different, the energy consumption associated with task computation on IoT device i is calculated by multiplying the energy consumption rate with the processing time of IoT device i. Mathematically, it can be expressed as follows:

$$E^{i}_{IoT} = p^{i}_{IOT} * \Psi^{i}_{IoT} \tag{10}$$

Where p^{i}_{IOT} and Ψ^{i}_{IoT} denotes the energy of the IoT device i and task's processing duration of IoT device i respectively. Task's processing time is calculated using the service rate of IoT device and the task arrival rate and expressed as follows:

$$\Psi^{i}_{IoT} = \frac{1}{\mu - \eta} \tag{11}$$

The energy consumption of a fog computational node is directly proportional to the amount of workload it handles. This means that if the volume of a task rises, the energy consumption of a device also increases monotonically. Mathematically, the energy consumption of a fog node is calculated as follows:

$$E^{j}_{FD} = \alpha * \varpi^{j^2}_{FD} + \beta * \varpi^{j}_{FD} + \partial \tag{12}$$

Where α, β, ∂ are constant and must be greater than 0, ϖ^{j}_{FD} denotes the allocated workload to fog node j.

The energy utilization across all cloud servers is the same due to an assumption of operating at an equal CPU processing frequency. This homogeneity in CPU utilization across all servers results in consistent energy consumption. However, the actual energy consumption of a server k is influenced by the individual CPU utilization of each computing virtual machine. The correlation between energy consumption and offloaded workload is directly proportional. This implies that upon increasing the workload on server k, the utilization of energy on server k also increases. During the period of reduced workload, certain computing virtual machines on servers are turned off to conserve energy. Mathematically, the power utilization of all servers is expressed as follows:

$$E^{k}_{CS} = \gamma_k * \left(\varphi_k(a_k M^{k}_{CS} + C_k)\right) \tag{13}$$

Where a_k and C_k are constant and their values must be greater than 0, M^{k}_{CS} is a count of virtual machine on cloud server k, φ_k is a count of ON virtual machines and γ_k indicate the state of virtual machine on server k i.e.,

$$\varphi_k = \begin{cases} 1, ON \\ 0, \text{OFF} \end{cases} \tag{14}$$

The total power consumption of the system can be expressed as a summation of energy consumption of all devices at different level. Mathematically,

$$\mathcal{Z}_{Energy} = E^{i}_{IOT} + E^{j}_{FD} + E^{k}_{CS} \tag{15}$$

For effective optimization of computation offloading and task scheduling, it is required to have a minimum value of $\mathcal{Z}_{Energy}$ and Π_{Delay}. Likewise, all the search solutions are evaluated to find out the energy consumption and delay of the system, and the search solution with the minimum energy utilization and delay is considered the optimal search solution.

E. NON-DOMINATED SOLUTION

The goal of multi-objective problems is to determine a group of solutions where none is superior to the other in all objectives. Particularly, one solution A dominates another solution B, if A performs better than B across all objectives. Conversely, non-dominated solutions or Pareto optimal solutions are the set of solutions that do not dominate each other and no solution is universally better across all objectives.

In each iteration, to store the non-dominated solution as the Pareto optimal solution we used an external repository of the same size as the number of search solutions. Within the repository, some of the Pareto optimal solutions have the same rank which needs to be eliminated to increase the diversity of the non-dominated solution. Consequently, it is necessary to update the repository to delete the same rank solution and to reduce the exhaustive search for the best optimal solution.

To this end, we implement a crowding distance function to update the external repository. It is a function that measures the density of solutions around a particular solution. It calculates the crowding distance of the neighboring solutions. High-crowding distance solutions are less crowded and contribute more to the diversity of the solution space and prevent local optimality.

The fitness value of search solutions is sorted in descending order to compute the crowding distance value. Upon reaching the maximum size of the external repository, a less crowding distance solution is replaced with a high crowding distance solution. Mathematically, we can express the crowding distance function below:

$$C_{dist} = \left(\frac{Dist^{i}}{(Dist^{max} - Dist^{min})}\right) \tag{16}$$

Where $Dist^{max}$, $D^{min}, Dist^{i}$ is the maximum and minimum distance between two solutions and distance value of the neighboring solution, respectively. Table IV contain the pseudo code of evaluation using optimization function.

TABLE IV
SOLUTION EVALUATION

```
Algorithm 2: Evaluation of Solution
1.  Function Fitness_Matrix = Fitness_func ($A_{COTS}$)
2.      Initialize $BSol_{Fitness}$ = Infinity
3.      For (i = 1→number of search agents)
4.      |   i). Calculate the fitness value of search agent using
        |       $OF_{Fitness}(i) = W * \Pi_{Delay} + (1-W) * \text{Ʒ}_{Energy}$
5.          ii) Fitness_Matrix (i)= $OF_{Fitness}(i)$
6.      End
7.      $OSol_{Fitness}$ = min($Fitness_{Matrix}$)
8.      if OSol_Fitness < B_Sol Fitness do {
9.      |   i).  BSol_Fitness = OSol_Fitness }
10.     |   ii). BSol_Loc = OSol_Loc
11.     End if
12. End Procedure
```

F. REPOSITIONING STRATEGY

The optimization function return the best search solution which encompasses the optimal task offloading and scheduling. Subsequently, other search solutions reposition their location vector by employing arithmetic operators (e.g., multiplication *M* "×", division *D* "÷", subtraction *S* "-", and addition *A* "+"). Repositioning process is performed by two search phases i.e., exploration and exploitation phase which guide the candidate solutions to assess potential solution nearby optimal solution. Transition between the search phases is conditioned by MOA function which is linearly increasing from 0.2 to 0.9, therefore it is vital to update the function value in each iteration using Eq. 1.

When the random value (*ran₁*) is greater than MOA function value, search solutions employs diversification phase to update the location. Conversely, when *ran₁* is less than MOA function value, the algorithm switches to exploitation phase to update the location vector. The pseudo code of update position is given in Table V.

TABLE V
REPOSITIONING STRATEGY

```
Algorithm 3: Repositioning Strategy
1.  Function CS_NewLoc= Repositioning(BSol_Fitness, BSol_Loc)
2.  Calculate MOA and MOP functions
3.  Generate ran_1, ran_2, and ran_3 between {0, 1}
4.  If (ran_1 ≤ MOA)
5.    | If (ran_3 ≤ 0.5) do {
6.    |   | Use '+' operator to update the ith solution location
7.    | Else
8.    |   | Use '- operator to update the ith solution location }
9.    | End if
10. Else
      | If (ran_2 ≤ 0.5) do {
11.   |   | Use 'x' math operator to update ith solution location
12.   | Else
13.   |   | Use '÷' math operator to update ith solution location }
14.   | End if
15. End if
16. End Procedure
```

1) EXPLORATION PHASE

In arithmetic operation, operators "×" and "÷" are deliberated as global search operators (exploration operator) because of their high distributed and desperation values. Such characteristics limit the ability of these operators in converging towards the target. Exploration operators globally investigate the problem space in search of potential feasible solutions via multiplication search and division search strategy. Transition between the searching strategies is conditioned by random number *ran₂*. The "÷" operator is activated to perform stochastic searching when *ran₂* exceed 0.5. Conversely, when *ran₂* falls below 0.5, "×" operator is engaged to perform operation in an attempt to explore the search space for potential feasible solutions. Mathematically, it is expressed as follows:

$$L_{(i,j)}Itr_N = \begin{cases} best(L_j) \div (MOP + \text{ι}) \times \left(\left(U_{b_j} - L_{b_j}\right) \times C_P + L_{b_j}\right), if\ ran_2 \le 0.5 \\ best(L_j) \times MOP \times \left(\left(U_{b_j} - L_{b_j}\right) \times C_P + L_{b_j}\right), if\ ran_2 \ge 0.5 \end{cases} \tag{17}$$

Where $best(L_j)$ is the location of best search solution, C_P is a control parameter to maintain a balance among different search strategies within the search space, ι is a constant, U_{b_j}, L_{b_j} represent the upper bound and lower bound respectively and MOP is a math optimizer probability and calculated as below:

$$MOP(Itr_C) = 1 - \left(\frac{(Itr_C)^{\frac{1}{a}}}{(Itr_M)^{\frac{1}{a}}}\right) \tag{18}$$

Where Itr_C, Itr_M denotes current and maximum iteration respectively. a is escalation parameter used to improve the intensification accuracy of the searching process. It help the algorithm in converging towards the optimal area.

2) EXPLOITATION PHASE

Exploitation phase involves investigating the obtained feasible locations in pursuit of approaching to the target optimal area. In arithmetic operation, operators " + " and " − " are deliberated as local operators (exploitative operators) because of their low dispersion, high density results and quick converging towards the target solution. The exploitation phase consist of addition and subtraction search strategy and activated when *ran₁* is less than MOA function value. Transition between the searching strategies is conditioned by another random number *ran₃*. The "+" operator is activated to perform exploitative searching when *ran₃* exceed 0.5. Conversely, when *ran₃* falls below 0.5, "−" operator is engaged to perform operation in an attempt to converge towards the near optimal solution. Mathematically, it is expressed as below:

$$L_{(i,j)}Itr_N = \begin{cases} best(L_j) + MOP \times \left(\left(U_{b_j} - L_{b_j}\right) \times C_P + L_{b_j}\right), if\ ran_3 > 0.5 \\ best(L_j) - MOP \times \left(\left(U_{b_j} - L_{b_j}\right) \times C_P + L_{b_j}\right), if\ ran_3 \le 0.5 \end{cases} \tag{19}$$

It is necessary to thwart the searching strategies to getting stuck in the local search area. To do this, we carefully designed C_P parameter that help the proposed algorithm to perform exploration stochastically not just at the beginning of iterations. This aid the algorithm in preventing local stagnation specifically in the last rounds.

G. TERMINATION CRITERIA

The algorithm terminates when either of two conditions are met.

- When the algorithm reaches to the maximum iteration
- When there is no improvement in the best fitness value over predetermined number of consecutive iterations

H. PSEUDO CODE OF PROPOSED SOLUTION

The pseudo code of the proposed solution is depicted in Table VI. The algorithm takes number of independent and heterogeneous task, number of computing nodes and computation time and resource of each task as input to the system. Line # 1-3 initialize the candidate solution and simulation parameters to their default values. Line # 4-7 is another initialization related to location vector of best solution, fitness value of best candidate solution and binary offloading matrix. All of these parameters are initialized to their initial values. Line # 8-15 creates task offloading and scheduling matrix by considering distinct task requirements and importance, evaluation of the matrix and determines the best candidate solution as per the optimization function. Line #16-22, the best candidate solution is stored in external archive as Pareto optimal solution whose size is equal to the number of candidate solution. To update the external archive with new best candidate solution, the proposed algorithm employs crowding distance value i.e., a solution with low crowding distance value is replaced by high crowding distance value. In line # 23-26, the proposed algorithm utilizes exploration and exploitation searching strategies to update candidate solution location. Line # 27-30, after repositioning new candidate solutions are created, followed by re-evaluation. If the new candidate solution is better than the previous solution, then the previous solution is updated with the new one. Upon reaching to termination criteria, the solution of the best candidate solution contains an optimal strategy of task offloading and scheduling.

TABLE VI
PROPOSED ALGORITHM

Algorithm 4: Proposed Algorithm

```
1.  Start
2.  Randomly initialize the candidate solution
3.  Initialize all simulation parameters to their default values i.e.,
    $\mu$ (service rate),$\eta$ (task arrival rate),
    (average length of queue) and external archive etc.
4.  For ($Itr = 1 \rightarrow Itr_{Max}$) do {
5.      $A_{COTS} = \{\}$
6.      $BSol_{Fitness}$ = Infinity
7.      $BSol_{Loc} = \{0,0\}$
8.      For ( $i$ = 1 → Number of candidate solution) do {
9.          $A_{COTS}$ = OffloadingMatrix($T$, $Comp_{Node}$, CSol)
10.         $Fitness_{Matrix} = Fitness_{func}(A_{COTS})$
11.         $OSol_{Fitness} = \min(Fitness_{Matrix})$
12.         if $OSol_{Fitness} < BSol_{Fitness}$ do {
13.             $BSol_{Fitness} = OSol_{Fitness}$
14.             $BSol_{Loc} = OSol_{Loc}$ }
15.         End if
16.         Store the Pareto optimal solution in archive
17.         $Ext_{Arch} = BSol_{Fitness}$
18.         if ($Ext_{Rep}$ == Number of candidate solution) do {
19.             Calculate $Dist_{CW}$
20.             Update the archive and replace low $Dist_{CW}$ with high
21.             $Dist_{CW}$}
22.         End if
23.     End For
24.     For ( k=1→ Number of search agents) do {
25.         update candidate solution
26.         $CS_{NewLoc}$= Repositioning($BSol_{Fitness}$, $BSol_{Loc}$)}
27.     End For
28.     Create new binary offloading matrix
29.     Evaluate the matrix using optimization function
30.     Store the Pareto optimal solution in external archive
31.     Select the best candidate solution i.e. $Best_{CSol}$ from the
        archive with best fitness value i.e., $BSol_{Fitness}$
32.     Itr = Itr +1}
33. End For
34. Optimal task offloading & scheduling= $Best_{CSol}$
```

I. IMPLEMENTATION OF ALGORITHM 4 IN CLOUD-FOG COMPUTING NETWORK

The practical implementation of algorithm 4 in cloud-fog computing network requires IoT devices, fog devices, cloud servers, and network links. The IoT devices generate tasks with varying requirements and priorities (e.g., computationally intensive and delay-sensitive tasks). The tasks are queued at the IoT devices and forwarded to the fog controller for task analysis and prioritization. The fog controller creates a binary offloading matrix that contains offloading and scheduling configurations based on task requirements. The offloading and scheduling decision is encoded into the solution of the search agent i.e., candidate solution. For instance, if there are 30 candidate solutions, there will be 30 offloading and scheduling decisions. The decisions are evaluated using the fitness function. Afterward, the arithmetic operators are applied to find the optimal offloading and scheduling configuration that minimizes the optimization objectives. The system then uses the best configuration for offloading and scheduling tasks

J. COMPUTATIONAL COMPLEXITY OF PROPOSED ALGORITHM

The computational complexity of the proposed algorithm is calculated for each phase and then combined to represent the overall complexity.

Initialization: The initialization phase depends upon the swarm size, if swarm size is N, then initialization phase takes O(N).

Solution construction: The solution construction phase, creates a binary offloading metrics which represents the offloading and scheduling of tasks to computing nodes. If M represent the number of tasks, then task offloading takes O(N) to offload tasks, the allocated tasks are scheduled for execution after determining the priority of each task. The tasks are sorted in ascending order based on their priority. These steps takes O (M.log$M$). So the overall complexity of solution creation phase is O(N x (M+Mlog$M$)) or O(N x M.log$M$).

Solution Evaluation: The complexity of solution evaluation phase is mainly depends upon the fitness function and problem dimension. This phase evaluates the solution of each search agent, so it takes O(N).

Update Position: The repositioning strategy requires O(LxN) to update the position vector of each search solution, where L is number of iterations.

Overall Complexity: So the computation complexity of proposed algorithm for one search solution and L iteration is O(L(N + N x $M$log$M$+ N+N) which collapse to O(L(N + N x M.log$M$)), for N search solution, it becomes O(L.(N.(N + N x M.log$M$))) which collapse to O(L.(N^2. M.log$M$)).

The comparative analysis of the computational complexity of similar benchmark algorithms is shown in Table VII. The computational complexity highlights the efficacy of that proposed algorithm in solving multi-objective optimization problems because of its ability to balance multiple conflicting objectives like energy, latency, and throughput. However, it has slightly higher computational complexity due to archive sorting which can slow down the performance of large computing networks. ACO and GA have the same computational complexity and are well-suited for single-objective optimization problems. However, more iterations are required to converge to the optimal solution. CTS has linear time complexity and is well suited for single objective optimization problems i.e., either energy consumption or latency, but suffers from premature convergence.

TABLE VII
COMPARATIVE ANALYSIS OF COMPUTATIONAL COMPLEXITY

Scheme	Time Complexity 1 search solution	Time Complexity N search solution
Proposed	O(L(N + N x M.log$M$))	O(L(N (N + N x M.log$M$)))
MoGWO	O(L(N^2 + M.log$M$))	O(L(N (N^2 + M.log$M$))
ACO	O(L(N +N^2 x M)))	O(L (N (N +N^2 x M)))
GA	O(L(N +N^2 x M)))	O(L (N (N +N^2 x M)))
CTS	O (L x M x N)	O (L (N x M x N))

V. EXPERIMENTAL SETUP AND RESULT DISCUSSION

The extensive experimental simulations were conducted using MATLAB 2021a on a 64-bit Windows 11 machine with 16 GB RAM and 2.5 GHz core i7 Intel processor. The simulation environment consists of varying fog nodes (ranging from 5 to 40) and heterogeneous IoT tasks (ranging from 60 to 500). The tasks were divided into delay-sensitive and computational-intensive tasks. The performance of the proposed solution was evaluated with the comparative approaches namely, MoGWO and cloud-fog cooperation scheduling algorithm (CTS), Ant Colony Optimization (ACO), and Genetic Algorithm (GA). The baseline algorithms and proposed algorithm were implemented under the same simulation parameters in Table 8 to ensure fairness in the evaluation. The comparative algorithms were implemented based on a standard approach with necessary modifications to fit the same optimization problem. The results demonstrate the operational efficacy of the proposed solution in terms of minimizing energy consumption, transmission latency while maximizing network throughput. Consequently, joint optimizing task offloading and scheduling prolong the network lifetime and improves the reliability, stability and scalability of the computing network by finding an optimal task offloading and scheduling strategy. This shows that the proposed solution can efficiently handle large number of diverse requests from various IoT devices, each with different requirements. The proposed approach is implemented as per the algorithm presented in Table VI.

The performance of the proposed solution was tested against three performance metrics i) latency ii) power consumption and iii) throughput. The study of these metrics was conducted under various simulation parameters as shown in Table VIII.

TABLE VIII
SIMULATION PARAMETERS

Parameters	Proposed Algorithm
Fog devices	5-40
Tasks count	60-500
Search solution	100
V_{max}	1
V_{min}	0.2
μ (service rate)	4.6
η (Task Arrival rate)	3.2
Archive size	100
Maximum Iteration	150
W	0.5
p_{IoT}^{i}	25
Simulation Run	10
a_k	4.9
C_k	53.2 x 10^{-20}
Escalation parameter	5
Control Parameter	0.5
Dimension	2

A. LATENCY VERSUS NUMBER OF TASKS

This simulation setup was designed to compute the combined latency of all three tiers under varying workloads, particularly with task counts varying as (60, 120, 150, 200, 250) and fog devices ranging from 5 to 10. The simulation results reveal the operational efficacy of the AOA-based task offloading and scheduling algorithm in terms of obtaining minimal transmission latency as compared to the state-of-the al gorithms. The baseline scheduling algorithms offload and execute the tasks without considering their significance and priority which causes higher transmission latency. However, the proposed algorithm used an efficient offloading and scheduling strategy to process the tasks based on their importance and priority, which minimize the transmission latency of the system. Additionally, the delay-sensitive tasks are executed and stored on the network edge which reduces the distance between IoT devices and fog nodes, thereby contributing to achieving minimum transmission latency. We can see in Fig. 3, that the transmission latency follows a monotonically increasing behavior, indicating the transmission latency linearly increases with the rise in workload. This verifies the stability and

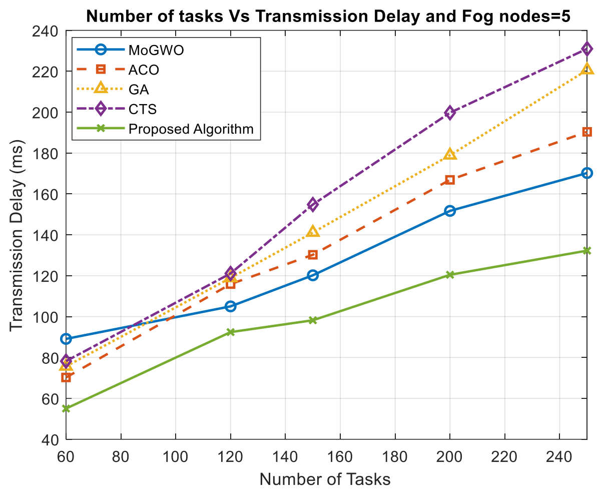


Figure 3. Latency verses Number of tasks & Fog nodes =5

scalability of the proposed algorithm against substantial task generation from various IoT devices, each with different requirements.

The next simulation setup was designed to compute the same performance metric but with increased workloads ranging from 300-500 tasks as shown in Fig. 4. This time, again the proposed algorithm outer perform the similar competitors in terms of optimizing communication delay.

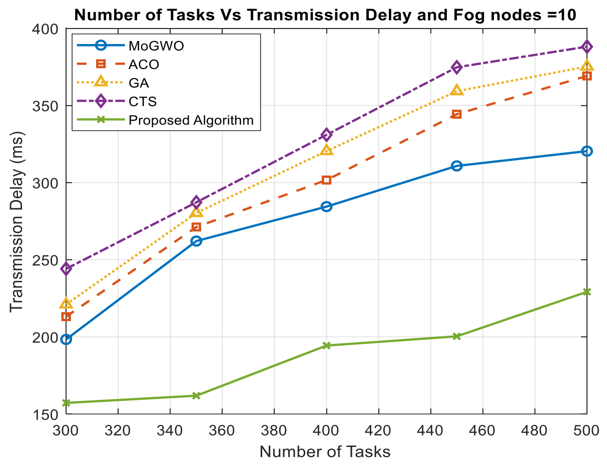


Figure 4. Latency verses Number of tasks & Fog nodes =10

Referring to the results of Fig. 3 and Fig. 4, we conclude that the proposed algorithm is useful for delay-sensitive IoT applications.

B. ENERGY CONSUMPTION VERSUS NUMBER OF TASKS

This performance metrics represent the collective energy consumption of the devices at all three tiers. The power utilization of a device mainly depends on the network bandwidth and CPU frequency of a computing node. This simulation was carried out to find the energy consumption of devices relative to the number of tasks ranging from 60 to 250. The simulation results in Fig. 5 reveals the effective performance of the proposed algorithm in terms of optimizing the power consumption of the scarce computational devices. Consequently, the resource utilization and lifetime of the network is extended.

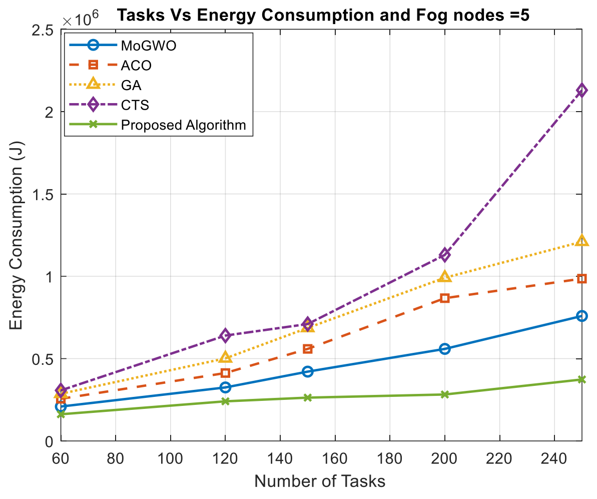


Figure 5. Tasks verses Energy Consumption & Fog nodes=5

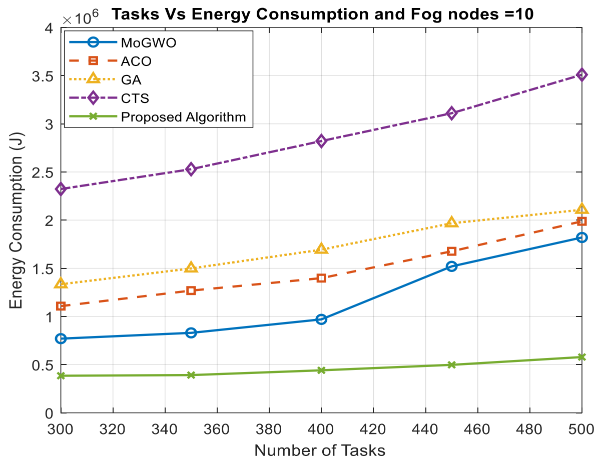


Figure 6. Task verses Energy Consumption & Fog nodes=10

In Fig. 6, we performed the same experiment but this time we increased the workload, ranging from 300-500 tasks. Again, the proposed algorithm outer perform the state of the art scheduling algorithms. This indicates the stability and scalability of the proposed solution, even when the number of tasks increases, the algorithm remains efficient in achieving the global optimal solution.

C. LATENCY VERSUS NUMBER OF FOG NODES

This experimental setup compute the transmission latency against the number of fog nodes ranging from 5 to 40 while keeping the workload constant i.e., 300 tasks. The simulation results in Fig. 7 and Fig. 8, illustrate the efficient performance of the proposed solution as compared to the competitors. Upon increasing the number of fog devices, the workload is effectively distributed across multiple computation devices, thereby the response time is increased and collective transmission latency is minimized.

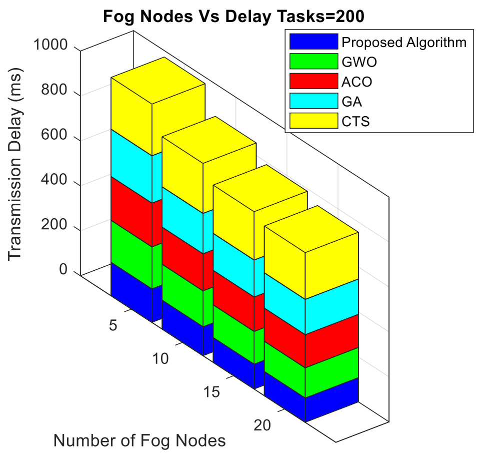


Figure 7. Delay verses number of fog node & task counts=200

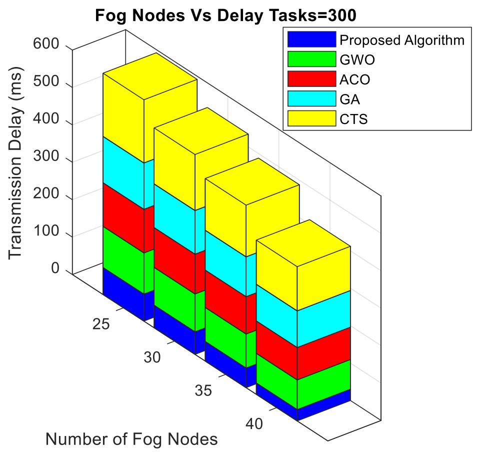


Figure 8. Delay verses number of fog node & task counts=300

D. ENERGY CONSUMPTION VERSUS NUMBER OF FOG NODES

This simulation setup calculate the power consumption of devices at each tier relative to the number of fog device varying from , while the workload remains constant at a value of 300 and 400 tasks. The relationship between the number of fog nodes and energy consumption is directly proportional to each other i.e., when we increase the count of fog devices, the energy consumption of the entire system also increases, because fog devices requires energy for data transmission, operation and communication. Nevertheless, the optimal resource allocation, effective and balance workload distribution can efficiently reduce the energy consumption of the system. The results in Fig. 9 demonstrate the operational efficacy of the proposed offloading and scheduling algorithm in terms of achieving optimized energy consumption as compared to the comparative resource allocation algorithms. The baseline resource allocation algorithms do not achieve a trade-off between offloading task to fog devices and cloud servers. However, the proposed algorithm achieves an efficient trade-off by intelligently determining a suitable computing device for task offloading and execution. This results in a reduction of energy consumption.

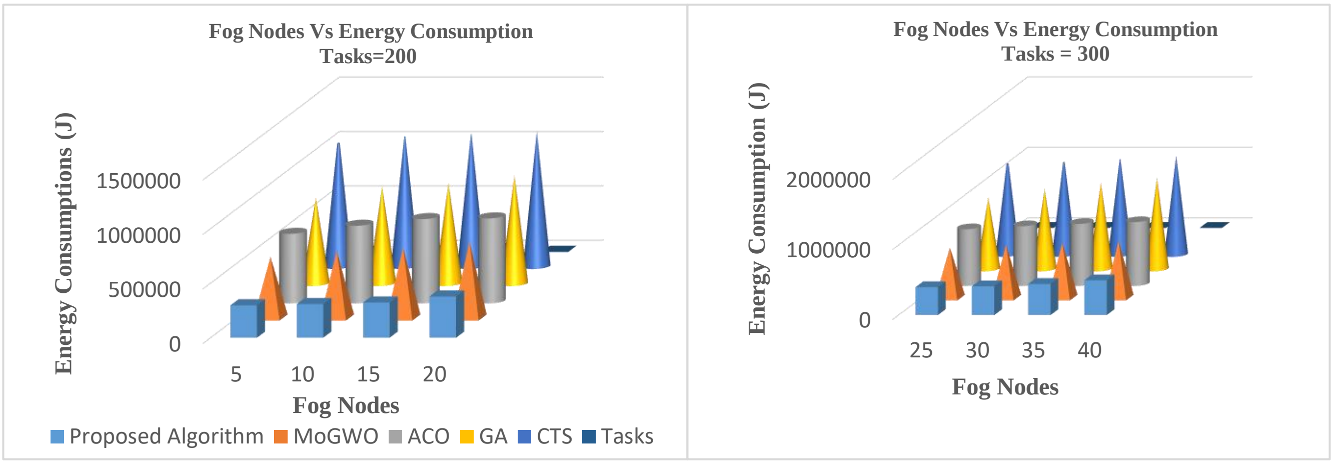


Figure 9. Energy Consumption verses Fog nodes (workload = 200 and 300)

D. THROUGHPUT VS NUMBER OF TASKS

This experimental framework assess the performance of the proposed solution concerning throughput relative to the workloads ranging from 50-500 tasks, while the fog nodes are maintained at a constant value i.e., 5 and 10. Throughput is measured by counting the number of successfully completed tasks in a stipulated period of time. The analysis of the performance metric is illustrated in Fig. 10, revealing efficient performance of the proposed algorithm in terms of number of successfully processed tasks as compared to the similar competitors. This efficiency is attributed to the joint optimal strategy of computation offloading and task scheduling that efficiently utilize the capacity of the computing resources without causing delays and avoiding unnecessary energy utilization. Consequently, the network lifetime increases and maximum throughput achieved in a less stipulated period of time.

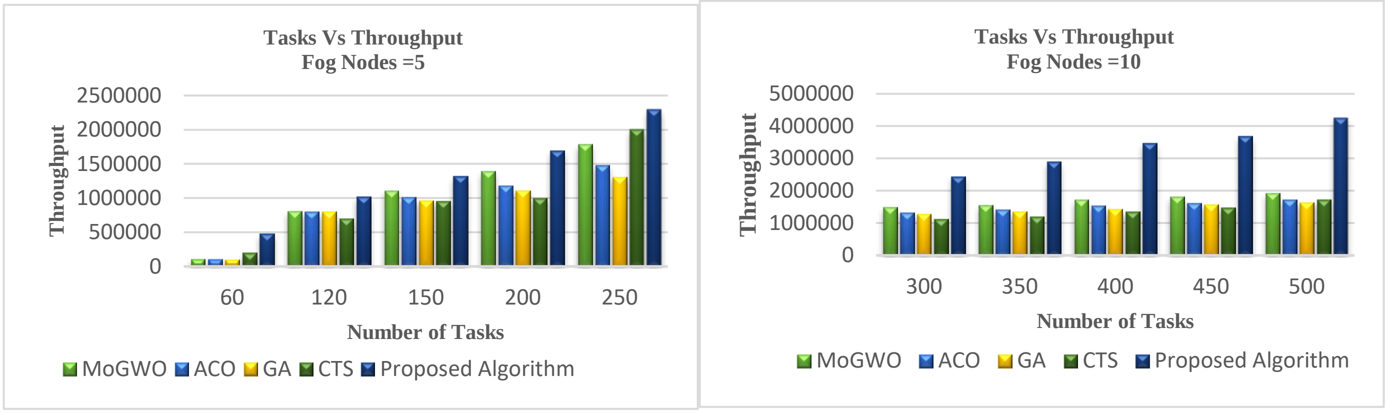


Figure 10. Throughput verses Tasks (Fog Nodes 5 and 10)

The numerical and performance analysis of the proposed solution in depicted Table IX and X respectively. The numerical data indicate the efficient performance provided by the proposed solution and performance enhancement up to a 25% in terms of better energy consumption and transmission delay.

Fig 11, unfolds the convergence behavior of proposed algorithm compared to the baseline approaches. The result unveils the efficient performance of the proposed algorithm, characterized by reliable and stable convergence behavior toward best solution. Moreover, the proposed algorithm achieves best fitness value at a faster convergence rate compared to the state-of-the-benchmark algorithms.

Table XI unfolds the convergence performance of the proposed and other comparative algorithms. The table has mainly three variables regarding convergence behavior i.e., maximum fitness value, best fitness value, and the standard deviation (S.D). The convergence values of these variables are recorded for workloads 200 and 400 tasks while the fog nodes remain constant i.e., 5 and 10. S.D measures the amount of dispersion in a set of values. A convergence behavior of an algorithm, it

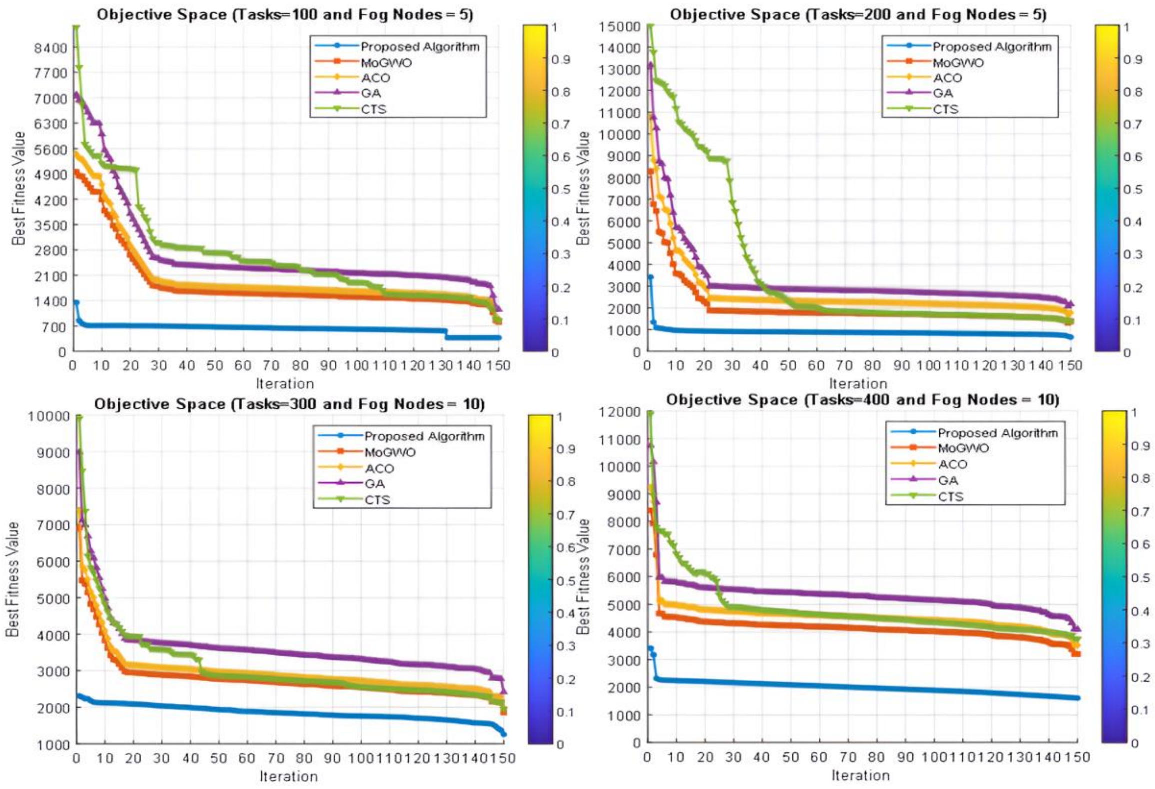


Figure 11: Convergence behavior of proposed and baseline algorithms

provides insights into the reliability and stability of the algorithm's performance. When the SD of the fitness values is relatively low, it reveals that the algorithm consistently converges toward the optimal solution across multiple rounds. We can see in Table X, that the best fitness value and SD of the proposed algorithm is comparatively low, which indicates the high stability and reliability of the algorithm's performance in obtaining optimal solution with less variations.

E. TASK COMPLETION RATE VERSUS NUMBER OF TASKS

Task completion rate specifies the number of tasks completed within a given deadline. This simulation setup calculates the task completion rate against the total number of tasks (60-500) generated by the IoT devices, while the computational nodes i.e., fog devices and cloud servers are kept constant i.e., 10. The simulation result in Fig 12, demonstrates that the proposed algorithm achieves a high (up to 97%) task completion rate compared to the baseline algorithms. As shown in the result, with the increase in the number of tasks, the completion rate begins to decline due to scarce computational resources. However, the proposed algorithm still achieves significant performance compared to benchmark algorithms, especially under higher workloads.

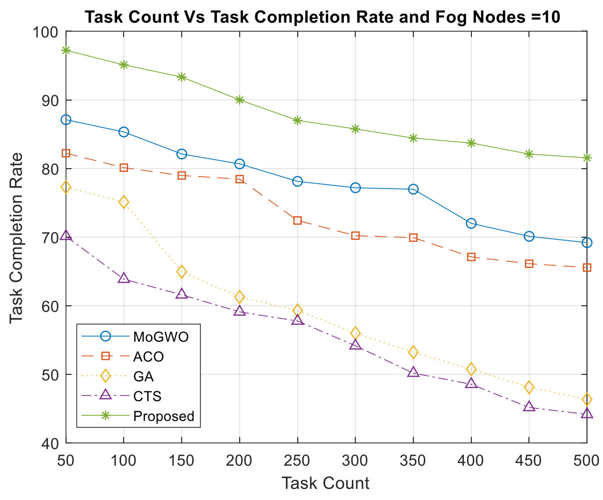


Figure 12: Number of tasks versus Task completion rate

F. FAIRNESS INDEX VERSUS NUMBER OF TASKS

The fairness index determines how fairly computational resources (e.g., bandwidth) are distributed among tasks. The system must fairly distribute resources among tasks and prevent resource starvation of low-priority tasks. The imbalance of resource distribution can lead to task drop-offs and performance degradation. The equation below is used to calculate the fairness index:

$$F_{index} = \frac{(\sum_{k=1}^{T} x_k)^2}{T \ \sum_{k=1}^{T} x_k^2} \qquad (20)$$

Where $x_k$represent the resource allocation to task k and T is the total number of tasks. When a fairness index value is near to 1, this indicates a fairness distribution of resources across all tasks, while when it is close to 0, this represents that resources are not evenly distributed among tasks.

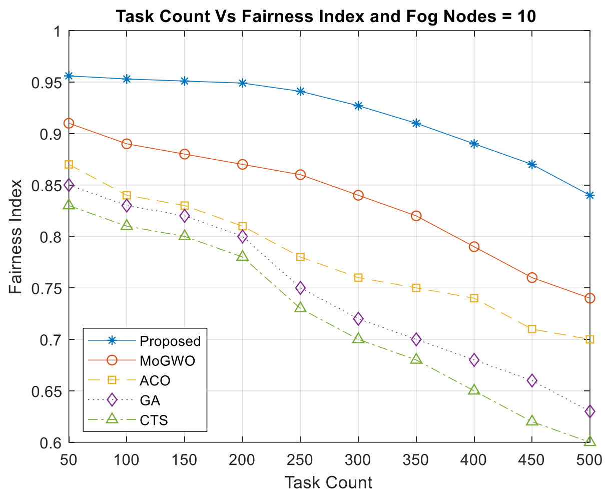


Figure 13: Number of tasks versus Fairness Index

Fig. 13 demonstrates the results of the fairness index of proposed and other comparative algorithms against the total number of tasks (60-500). For the simulation, the number of computational nodes is kept constant i.e., 10. The results show that the proposed algorithm obtains a high fairness index value compared to other baseline algorithms. It is observed that with the increase in the number of tasks maintaining a high fairness index value is more challenging. The results highlight the efficiency of the proposed algorithm in distributing resources evenly across all tasks, especially under higher workloads. This minimizes the risk of resource bottlenecks and resource starvation.

As a whole, we can safely infer that the proposed algorithm efficiently solves the multi-objective NP-hard optimization problem. This efficiency is attributed to the inherent characteristics of MoAOA. Its rich repertoire of stochastic operators and their efficient utilization enable the algorithm to equalize different search strategies during both the diversification and intensification phase. Additionally, the strategic design of control variable allow candidate solution to engage in exploration not just at the beginning of the iteration but also towards the end of the iterations. This capability help the algorithm in preventing local optima solution and providing global optimal solution for complex multi-objective optimization problem. Considering the efficient outcomes of the proposed algorithm in terms of optimization objectives, we can efficiently apply the proposed algorithm to real-world applications. For example, in a smart healthcare system within the Internet of Medical Things (IoMT), the proposed scheme offers superior performance. The IoMT network consist of wearable health sensors to measure important indicators like blood pressure, heart beat and sugar level etc. These devices generates massive amount of tasks with varying requirements and priorities that needs to be processed accordingly. For instance, an emergence alert message from a wearable sensor is considered as latency-sensitive task and must be given higher priority. These tasks are offloaded to nearby fog devices and executed before the lower priority tasks to ensure that they meet their deadlines. Since wearable sensors often have limited energy and battery lifetime, therefore the proposed algorithm could intelligently allocate computational-intensive tasks (e.g., medical image processing and genomic data analysis) to powerful computing resources like server, resulting in better energy management.

TABLE IX
NUMERICAL ANALYSIS

Algorithms	Delay (ms)	Energy (J)	Delay (ms)	Energy (J)	Throughput
Nodes/ Task	5-40 Node	5-40 Node	60-500 Tasks	60-500 Tasks	60-500 Tasks
CTS	181.41	1.25E+06	240.99	1.90E+06	9.03E+05
MoGWO	126.12	6.93E+05	201.23	8.18E+05	1.37E+06
GA	144.67	1.04E+06	229.13	1.23E+06	1.16E+06
ACO	133.62	7.96E+05	217.30	1.05E+06	1.22E+06
Proposed	88.83	3.82E+05	144.13	3.62E+05	2.36E+06

TABLE X
PERFORMANCE ANALYSIS

Algorithms	Performance Latency	Performance Energy Consumption	Performance Throughput	Net Performance
Proposed	51.95%	84.35%	96.6%	77.64%
GA	23.62%	49.15%	49.66%	40.81%
ACO	27.56%	57.34%	52.54%	45.81%
CTS	19.66%	21.5%	36.09%	25.75%
MoGWO	32.92%	68.18%	58.99%	53.7%

TABLE XI
OBJECTIVE VALUES AND S.D OF PROPOSED AND COMPARATIVE ALGORITHMS

Algorithms	Tasks/Fog Nodes	Max Fitness	Best Fitness	S.D	Tasks/Fog Nodes	Max Fitness	Best Fitness	S.D
Proposed		3.40E+03	1.60E+03	239.30		3.42E+03	6.58E+02	222.21
MoGWO		8.39E+03	3.20E+03	584.92		8.28E+03	1.37E+03	1035.72
ACO	400/10	9.24E+03	3.52E+03	643.05	200/5	1.08E+04	1.78E+03	1346.44
GA		1.08E+04	4.09E+03	748.27		1.32E+04	2.17E+03	1646.82
CTS		1.91E+04	3.75E+03	1098.01		1.50E+04	1.40E+03	3502.82

VI. CONCLUSION

In this paper, we present a novel joint optimization of computation offloading and task scheduling algorithm based on MoAOA for cloud-fog networks. The proposed algorithm classifies the incoming tasks based on their requirements and importance and takes energy and delays efficient offloading and scheduling decisions accordingly. The proposed algorithm is implemented on the fog controller to estimate, analyze, offload, and schedule tasks to the appropriate computing resource. The optimization process starts with the initialization of search agents within the problem space where all the search agents cooperatively explore and exploit the problem space to find a set of trade-off solutions via Pareto-optimality.

Extensive simulation is performed in MATLAB and the results are compared with similar methodologies. The results validate the streamlined functionality of the proposed algorithm in terms of optimizing energy consumption, transmission latency, and network throughput. Moreover, the results demonstrate the stability and scalability of the proposed solution, as it is growing steadily with the increase in the workload.

In the future, we can explore other swarm intelligence algorithms (e.g., Bat Optimization algorithm, whale optimization algorithm, etc.) to investigate the multi-objective task offloading and scheduling optimization problem to optimize conflicting optimization objectives. Moreover, we can also integrate machine learning models to optimize objectives like transmission cost, security, privacy and load balancing, etc. The proposed algorithm is computationally expensive for a larger network size, as the network grows, the algorithm may take longer optimization times. In the future, we can address this limitation by exploring parallel processing techniques and distributed computing frameworks to optimize computational time.

CRedit Authorship Contribution Statement

Conceptualization, A.A. and N.A and M.T.B; methodology, A.A and M.A, M.T.B and H.H; software, N.A, M.Al.A. A.R, and H.H; validation, F.U.K., M.O and A.R.; formal analysis, M.A, M.Al.A. and F.U.K.,; investigation, A.R. M.Al.A and F.U.K; data curation, M.A, H.H. and M.O; writing—original draft preparation, A.A, A.R, M.O; writing—review and editing, All Authors.; funding acquisition, M.T.B, H.H, and A.R.

Acknowledgement Statements

The Researchers would like to thank the Deanship of Graduate Studies and Scientific Research at Qassim University for financial support (QU-APC-2024-9/1).

Declaration of conflict of interest

The authors declare that they have no known conflict of interest to be reported in this paper.

References

1. Gai, K., et al., *Blockchain meets cloud computing: A survey.* IEEE Communications Surveys & Tutorials, 2020. **22**(3): p. 2009-2030.

2. Cao, K., et al., *A survey on edge and edge-cloud computing assisted cyber-physical systems.* IEEE

Transactions on Industrial Informatics, 2021. **17**(11): p. 7806-7819.

3. Miao, Y., et al., *Intelligent task prediction and computation offloading based on mobile-edge cloud computing.* Future Generation Computer Systems, 2020. **102**: p. 925-931.
4. Iftikhar, S., et al., *HunterPlus: AI based energy-efficient task scheduling for cloud–fog computing environments.* Internet of Things, 2023. **21**: p. 100667.
5. Abd Elaziz, M., L. Abualigah, and I. Attiya, *Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments.* Future Generation Computer Systems, 2021. **124**: p. 142-154.
6. Guevara, J.C. and N.L. da Fonseca, *Task scheduling in cloud-fog computing systems.* Peer-to-Peer Networking and Applications, 2021. **14**(2): p. 962-977.
7. Sabireen, H. and V. Neelanarayanan, *A review on fog computing: Architecture, fog with IoT, algorithms and research challenges.* Ict Express, 2021. **7**(2): p. 162-176.
8. Rahbari, D. and M. Nickray, *Task offloading in mobile fog computing by classification and regression tree.* Peer-to-Peer Networking and Applications, 2020. **13**: p. 104-122.
9. Khiat, A., M. Haddadi, and N. Bahnes, *Genetic-Based Algorithm for Task Scheduling in Fog–Cloud Environment.* Journal of Network and Systems Management, 2024. **32**(1): p. 3.
10. Kishor, A. and C. Chakarbarty, *Task offloading in fog computing for using smart ant colony optimization.* Wireless personal communications, 2022. **127**(2): p. 1683-1704.
11. Liu, Z., et al., *A distributed algorithm for task offloading in vehicular networks with hybrid fog/cloud computing.* IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2021. **52**(7): p. 4388-4401.
12. Saif, F.A., et al., *Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing.* IEEE Access, 2023. **11**: p. 20635-20646.
13. Ali, A., et al., *Multi-Objective Harris Hawks Optimization Based Task Scheduling in Cloud-Fog Computing.* IEEE Internet of Things Journal, 2024.
14. Abualigah, L., et al., *The arithmetic optimization algorithm.* Computer methods in applied mechanics and engineering, 2021. **376**: p. 113609.
15. Li, G., et al., *Energy consumption optimization with a delay threshold in cloud-fog cooperation computing.* IEEE access, 2019. **7**: p. 159688-159697.
16. Hussein, M.K. and M.H. Mousa, *Efficient task offloading for IoT-based applications in fog computing using ant colony optimization.* IEEE Access, 2020. **8**: p. 37191-37201.
17. Bansal, S., H. Aggarwal, and M. Aggarwal, *A systematic review of task scheduling approaches in fog computing.* Transactions on Emerging Telecommunications Technologies, 2022. **33**(9): p. e4523.
18. Hosseinzadeh, M., et al., *Task scheduling mechanisms for fog computing: A systematic survey.* IEEE Access, 2023.
19. Salaht, F.A., F. Desprez, and A. Lebre, *An overview of service placement problem in fog and edge computing.* ACM Computing Surveys (CSUR), 2020. **53**(3): p. 1-35.
20. Liu, C., et al., *Solving the multi-objective problem of IoT service placement in fog computing using cuckoo search algorithm.* Neural Processing Letters, 2022. **54**(3): p. 1823-1854.
21. Chang, Z., et al., *Dynamic resource allocation and computation offloading for IoT fog computing system.* IEEE Transactions on Industrial Informatics, 2020. **17**(5): p. 3348-3357.
22. Wu, C.-g., et al., *An evolutionary fuzzy scheduler for multi-objective resource allocation in fog computing.* Future Generation Computer Systems, 2021. **117**: p. 498-509.
23. Gu, L., et al., *Cost efficient resource management in fog computing supported medical cyber-physical system.* IEEE Transactions on Emerging Topics in Computing, 2015. **5**(1): p. 108-119.
24. Mukherjee, M., et al., *Task data offloading and resource allocation in fog computing with multi-task delay guarantee.* Ieee Access, 2019. **7**: p. 152911-152918.
25. Vu, T.T., et al., *Optimal task offloading and resource allocation for fog computing.* arXiv preprint arXiv:1906.03567, 2019.
26. Daneshfar, N., et al. *Service allocation in a mobile fog infrastructure under availability and qos constraints*. in *2018 IEEE Global Communications Conference (GLOBECOM)*. 2018. IEEE.
27. Aazam, M., S. Zeadally, and E.F. Flushing, *Task offloading in edge computing for machine learning-based smart healthcare.* Computer Networks, 2021. **191**: p. 108019.
28. Fan, W., et al., *Joint task offloading and resource allocation for accuracy-aware machine-learning-based IIoT applications.* IEEE Internet of Things Journal, 2022. **10**(4): p. 3305-3321.
29. Razaq, M.M., et al., *Fragmented task scheduling for load-balanced fog computing based on Q-learning.* Wireless communications and mobile computing, 2022. **2022**.

30. Yang, Z. and W. Bai, *Distributed Computation Offloading in Mobile Fog Computing: A Deep Neural Network Approach*. IEEE Communications Letters, 2021. **26**(3): p. 696-700.

31. Yuan, L.-L., et al., *Dynamic level scheduling based on trust model in grid computing*. CHINESE JOURNAL OF COMPUTERS-CHINESE EDITION-, 2006. **29**(7): p. 1217.

32. Mazrekaj, A., et al. *The Experiential Heterogeneous Earliest Finish Time Algorithm for Task Scheduling in Clouds*. in *CLOSER*. 2019.

33. Lai, K.-C. and C.-T. Yang, *A dominant predecessor duplication scheduling algorithm for heterogeneous systems*. The Journal of Supercomputing, 2008. **44**: p. 126-145.

34. Li, K., *Heuristic computation offloading algorithms for mobile users in fog computing*. ACM Transactions on Embedded Computing Systems (TECS), 2021. **20**(2): p. 1-28.

35. Kanemitsu, H., M. Hanada, and H. Nakazato, *Clustering-based task scheduling in a large number of heterogeneous processors*. IEEE Transactions on Parallel and Distributed Systems, 2016. **27**(11): p. 3144-3157.

36. Ali, A., et al., *Harris hawks optimization-based clustering algorithm for vehicular ad-hoc networks*. IEEE Transactions on Intelligent Transportation Systems, 2023.

37. Gandomi, A.H., et al., *Metaheuristic algorithms in modeling and optimization*. Metaheuristic applications in structures and infrastructures, 2013. **1**: p. 1-24.

38. Wu, B., et al., *Optimal deploying IoT services on the fog computing: A metaheuristic-based multi-objective approach*. Journal of King Saud University-Computer and Information Sciences, 2022. **34**(10): p. 10010-10027.

39. Keshavarznejad, M., M.H. Rezvani, and S. Adabi, *Delay-aware optimization of energy consumption for task offloading in fog environments using metaheuristic algorithms*. Cluster Computing, 2021: p. 1-29.

40. Potu, N., C. Jatoth, and P. Parvataneni, *Optimizing resource scheduling based on extended particle swarm optimization in fog computing environments*. Concurrency and Computation: Practice and Experience, 2021. **33**(23): p. e6163.

41. Yuan, H., J. Bi, and M. Zhou, *Energy-efficient and QoS-optimized adaptive task scheduling and management in clouds*. IEEE Transactions on Automation Science and Engineering, 2020. **19**(2): p. 1233-1244.

42. Li, J., et al., *Multiobjective oriented task scheduling in heterogeneous mobile edge computing networks*. IEEE Transactions on Vehicular Technology, 2022. **71**(8): p. 8955-8966.

43. Hosny, K.M., et al., *Optimized multi-user dependent tasks offloading in edge-cloud computing using refined whale optimization algorithm*. IEEE Transactions on Sustainable Computing, 2023.

44. Vispute, S.D. and P. Vashisht, *Energy-efficient task scheduling in fog computing based on particle swarm optimization*. SN Computer Science, 2023. **4**(4): p. 391.

45. Skarlat, O., et al. *Resource provisioning for IoT services in the fog*. in *2016 IEEE 9th international conference on service-oriented computing and applications (SOCA)*. 2016. IEEE.

46. Zeng, D., et al., *Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system*. IEEE Transactions on Computers, 2016. **65**(12): p. 3702-3712.

47. Movahedi, Z., B. Defude, and A.M. Hosseininia, *An efficient population-based multi-objective task scheduling approach in fog computing systems*. Journal of Cloud Computing, 2021. **10**(1): p. 53.

48. Tang, M. and V.W. Wong, *Deep reinforcement learning for task offloading in mobile edge computing systems*. IEEE Transactions on Mobile Computing, 2020. **21**(6): p. 1985-1997.

49. Li, H., K. Ota, and M. Dong, *Deep reinforcement scheduling for mobile crowdsensing in fog computing*. ACM Transactions on Internet Technology (TOIT), 2019. **19**(2): p. 1-18.

50. Poltronieri, F., et al. *Reinforcement learning for value-based placement of fog services*. in *2021 IFIP/IEEE international symposium on integrated network management (IM)*. 2021. IEEE.

51. Xia, Y., et al. *Combining hardware nodes and software components ordering-based heuristics for optimizing the placement of distributed IoT applications in the fog*. in *Proceedings of the 33rd annual ACM symposium on applied computing*. 2018.

52. Mahmud, R., K. Ramamohanarao, and R. Buyya, *Latency-aware application module management for fog computing environments*. ACM Transactions on Internet Technology (TOIT), 2018. **19**(1): p. 1-21.

53. Mahmoud, M.M., et al., *Towards energy-aware fog-enabled cloud of things for healthcare*. Computers & Electrical Engineering, 2018. **67**: p. 58-69.

54. Gu, B., et al., *A distributed and context-aware task assignment mechanism for collaborative mobile edge computing*. Sensors, 2018. **18**(8): p. 2423.

55. Abdel-Basset, M., et al., *Multi-objective task scheduling method for cyber–physical–social systems*

in fog computing. Knowledge-Based Systems, 2023. **280**: p. 111009.

56. Shukla, P. and S. Pandey, *DE-GWO: a multi-objective workflow scheduling algorithm for heterogeneous fog-cloud environment.* Arabian Journal for Science and Engineering, 2024. **49**(3): p. 4419-4444.

57. Abualigah, L., et al., *Ts-gwo: Iot tasks scheduling in cloud computing using grey wolf optimizer*, in *Swarm intelligence for cloud computing*. 2020, Chapman and Hall/CRC. p. 127-152.

58. Dhal, K.G., et al., *A comprehensive survey on arithmetic optimization algorithm.* Archives of Computational Methods in Engineering, 2023. **30**(5): p. 3379-3404.

ASAD ALI received his BS degree in Telecommunication and Networking (with distinction) from COMSATS University Islamabad, Attock campus and MS degree in Computer Science (with distinction) from University of Engineering and Technology, Peshawar in 2016 and 2020 respectively. Currently, working as a Lecturer in computer science department of Mardan Institute of Science and Technology affiliated with Abdul Wali Khan University Mardan. He has over 8 years of experience in teaching and research in computer science. His current research interest includes VANET, Swarm Intelligence, Resource optimization, IoTs, SDN, Fog Computing, Route optimization, Cybersecurity and 5G networking.

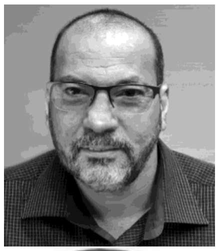
NAZIA AZIM received the M.S. degree in computer science from Agricultural University Peshawar and Ph.D. degree from Abdul Wali Khan University Mardan in 2011 and 2021 respectively. She is currently working as Lecturer at the department of CS, Abdul Wali Khan University Mardan, Pakistan. Her research interests include image processing, bioinformatics, and computational modeling.

MOHAMED TAHAR BEN OTHMAN (Senior Member, IEEE) received the Ph.D. degree in computer science from the National Institute of Polytechnic of Grenoble (INPG), France, in 1993, the master's degree in computer science from the École Nationale Supérieure d'Informatique et de Mathématiques Appliquées de Grenoble (ENSIMAG), in 1989, and the Senior Engineer Diploma degree in computer science from the Faculty of Sciences of Tunis. He worked as a Postdoctoral Researcher at the Laboratoire de Génie Logiciel (LGI), Grenoble, France, from 1993 to 1995, and the Dean of the Faculty of Science and Engineering, University of Science and Technology, Sana'a, Yemen, from 1995 to 1997. He was a Senior Software Engineer with Nortel Networks Corporation, Canada, from 1998 to 2001, and an Assistant Professor with the Computer College, Qassim University, Saudi Arabia, from 2002 to October 2010, and also an Associate Professor. He has been a Professor of computer science, since November 2017. His research interests include data mining, artificial intelligence, information security, and bioinformatics.

ATEEQ UR REHMAN (Senior Member, IEEE) received his BS Degree in Electrical (Telecommunication) Engineering from COMSATS Institute of Information Technology, Lahore, Pakistan, in 2009 and MS degree in Electrical Engineering with a specialization in Telecommunications from the Blekinge Institute of Technology (BTH), Karlskrona, Sweden in 2011. He completed his Ph.D. degree in the College of Internet of Things (IoT) Engineering, Hohai University (HHU), Changzhou Campus, China, in 2022. Currently, he is an assistant professor with the School of Computing, Gachon University, South Korea. He has contributed to various international IEEE conferences and journals of repute. His research interests include but are not limited to Biomedical Signal Processing, Internet of Things (IoTs), Social Internet of Things (SIoTs), Big Data, and Renewable Energy Technologies.

MASOUD ALAJMI (Member, IEEE) received the B.S. degree in electrical engineering from the King Fahad University of Petroleum and Minerals (KFUPM), in 2004, and the M.S. degree in electrical engineering and the Ph.D. degree in electrical and computer engineering from Western Michigan University, Kalamazoo, MI, USA, in 2010 and 2016, respectively. He has over four years of experience in industry. He was with Zamel and Turbag Consulting Engineers, Al-Khobar, Saudi Arabia, as an Electrical Engineer, for three months. Then, he was with Saudi Electricity Company (SEC), Abha, Saudi Arabia, where he was a Pre-Commissioning Engineer, from 2004 to 2008. During that period, he completed many training programs in the technical and administrative fields with well-known institutes. Also, he was assigned to be a commissioning leader of many projects in Saudi Arabia. He was assigned to be the SEC Representative to supervise factory acceptance tests for Siemens Company, Berlin, Germany, in 2007, and Hyundai Heavy Industries Company Ltd., Ulsan, South Korea, in 2008. From 2012 to 2015, he was a Teaching Assistant with the Electrical and Computer Engineering Department, Western Michigan University. He is currently an Associate Professor with the Computer Engineering Department, Taif University, Taif, Saudi Arabia. He has involved in various technical committees. He is the coauthor of about 30 papers in international journals and conference proceedings. His research interests include signal processing, biomedical image processing, image encryption, watermarking, steganography, data hiding, machine learning, smart grids, and renewable energy. He received the 2014–2015 Graduate Teaching Effectiveness Award from Western Michigan University for excellent teaching skills

FAHEEM ULLAH KHAN received his PhD degree from United Kingdom (UK). He is currently working as Assistant Professor at Department of Software Engineering, University of Science and Technology, Bannu, Pakistan. His research interests include cybersecurity, medical image processing, and machine/deep learning, the Internet of Things (IoT), distributed computing and computer vision



MOSLEH HMOUD AL-ADHAILEH received the Ph.D. degree in computer science (AI). He is currently the Director of e-learning and distance education for operation with King Faisal University. His current research interests include artificial intelligence(AI), machine learning (ML), natural language processing, robotics programming, knowledge representation, and e-learning strategies and technologies.

ORKEN MAMYRBAYEV received the B.S. and M.S. degrees in information systems from Abai University, Almaty, Kazakhstan, and the Ph.D. degree in information systems from Kazakh National Technical University named after K. I. Satbayev. He was an Associate Professor with the Institute of Information and Computational Technologies, Kazakhstan. He has been a Senior Researcher with the Laboratory of Computer Engineering of Intelligent Systems, Institute of Information and Computational Technologies. He is currently the Deputy General Director and the Head of the Laboratory of Computer Engineering of Intelligent Systems, Institute of Information, Kazakhstan. He is also a member of the Dissertation Council "Information Systems," L. N. Gumilyov Eurasian National University in the specialties computer sciences and information systems. He is the author of five books, more than 130 articles, and more than 20 inventions and copyright certificates for an intellectual property object in software. His main research interests include machine learning, deep learning, and speech technologies.

HABIB HAMAM (Senior Member, IEEE) received the B.Eng. and M.Sc. degrees in information processing from the Technical University of Munich, Germany, 1988 and 1992, respectively, the Ph.D. degree in physics and applications in telecommunications from Université de Rennes I conjointly with France Telecom Graduate School, France, in 1995, and the Postdoctoral Diploma degree ''Accreditation to Supervise Research in Signal Processing and Telecommunications'' from Université de Rennes I, in 2004. From 2006 to 2016, he was the Canada Research Chair of Optics in Information and Communication Technologies, for a period of ten years. He is currently a Full Professor with the Department of Electrical Engineering, University of Moncton. His research interests include optical telecommunications, wireless communications, diffraction, fiber components, RFID, information processing, data protection, COVID-19, and deep learning. He is a Senior Member of OSA and a Registered Professional Engineer in New-Brunswick. He is among others the Editor-in-Chief of *CIT-Review* and an Associate Editor of *IEEE Canadian Review*.