Enhanced PSO Optimized Leader in Cloud-Fog Task Scheduling for IoT and Mobile Crowdsensing Environments

Abbas M. Ali Al-muqarm[1,*] and Naseer Ali Hussien[2]
[1] Department of Computer Science, Faculty of Computer Science and Mathematics, University of Kufa, Najaf, Iraq
[2] Scientific Research Center, Alayen University, Wasit, Iraq
Email: abbasm.almuqarm@uokufa.edu.iq (A.M.A.), naseerali@alayen.edu.iq (N.A.H.)

***Abstract*—The data generated by the IoT needs a powerful platform such as cloud computing for data processing. However, the cloud faces challenges when dealing with various types of resources, high delay, and cost, this represents a substantial challenge in scheduling tasks. Therefore, the need appeared to introduce the concept of fog. To address these limitations, optimization algorithms such as PSO were used. In traditional PSO, all particles in the swarm are influenced by a single global best particle (G_{best}), if it becomes stuck in a local optimum, all the particles will move closer to it, thus, the PSO may easily get trapped in premature convergence. This paper proposed an adaptive cloud-fog integrated approach based on modified PSO called PSO Optimized Leader (PSO-OL). These modifications on four stages: Firstly, a method to ensure swarm diversity in the initialization phase is introduced. Secondly, to reduce the chance of the population getting trapped in a local optimum, the farthest-best particle is introduced. Third, in addition to the primary G_{best}, a second G_{best} represents a different good particle presented to explore multiple promising regions. Finally proposed a new crossover operator to get an optimized leader. The PSO-OL approach was evaluated and the results show the effectiveness of the enhanced leader by 40% with farthest-best, 45% with second-Gbest when compared to standard PSO, and when compared to scheduling algorithms where outperforms the other algorithms by minimizing makespan by 34%, cost by 14%, and increasing throughput by 75%, in comparison to existing load balancing and scheduling methods: RR, BLA, MPSO, ETS, and TCaS.**

***Index Terms*—cloud-fog, task scheduling, PSO, optimization, IoT, mobile crowd sensing**

I. Introduction

Internet of Thing (IoT) is a contemporary innovation that has had a powerfully affected on communication and information technologies. It has revolutionized how various devices and objects, such as surveillance cameras, cars, and smartphones, are linked to the Internet, enabling them to execute a wide range of applications and operations. These advancements, including M2M technologies, have expanded Internet access to these devices and have facilitated activities like controlling traffic, vehicular networking, power management, healthcare services, and healthcare. As a result, an enormous volume of data is generated by these smart devices, which requires management, processing, and analysis to extract valuable insights and ensure that client software and end users can access them [1]. While sensors and IoT have proven to be efficient in data collection and urban surveillance, the installation and deployment of sensor devices across a city are both time-consuming and costly. As an alternative to sensors and IoT, a novel approach called Mobile Crowdsensing (MCS) has arisen. MCS represents a new approach to data collection, enabling regular individuals to share data from their mobile devices. This information is pooled and processed in the cloud to extract crowd intelligence and deliver services focused on people's needs. The data gathered from various sources is integrated into the cloud, serving as a collector for storage and processing. This evolution towards MCS is gaining importance as it replaces conventional static sensors, offering a combination of traditional IT advantages and mobile communication, delivering cost effective, and top-quality services across diverse domains. At present, smartphones come embedded with a diverse collection of sensors, including cameras, microphones, GPS systems, and accelerometers. Consequently, MCS holds a distinct advantage over conventional sensor utilization due to this enhanced sensor suite [2].

Cloud computing, as the central component of the IoT, offers a range of services including storage capacity, powerful processing capabilities, and computing resources. It also facilitates the visualization of these resources. However, a common challenge with cloud servers is their physical distance from end devices. This can lead to significant delays in wide area networks (WAN) and a diminished Quality of Service (QoS), especially in applications that are sensitive to latency. Furthermore, as the increasing number of devices that are connected and IoT applications continue to grow, cloud computing encounters various optimization issues. These challenges include bandwidth limitations, privacy concerns, delays, storage capacity constraints, security, and the need to address the excessive concentration of computing resources [3, 4].

Manuscript received October 29, 2023; revised November 29, 2023; accepted December 2, 2023.
*Corresponding author

Fog computing is an emerging architectural concept that aims to address the remoteness issue between IoT devices and cloud resources. This novel approach, innovative by Cisco, extends the capabilities of cloud to the network's edge. Although still in its early stages, fog computing has gained recognition as a valuable addition to cloud. It can be described as a distributed computing infrastructure that brings computational power closer to the network's edge, enabling cloud resources to be readily accessible. Additionally, due to the notable network latency, the transfer of IoT tasks to the cloud leads to a heightened delay in the response time for data analysis [5].

Efficient task scheduling within cloud-fog architectures is a critical concern. Optimizing the utilization of cloud-fog resources to improve key factors like execution time of tasks, operational costs, and consumption of energy is of the greatest importance. Effective task scheduling within fog system plays a vital role in cost reduction, processing time, and communication delays. However, researchers often face difficulties in identifying an efficient task scheduling method that meets their requirements [6, 7].

Generally, the problem at hand is categorized as an NP-hard problem since it cannot be efficiently solved in polynomial time by adding more sensors and fog nodes. As a result, traditional methods are unable to address this challenge effectively. To overcome this issue, researchers have turned their attention to swarm and evolutionary algorithms. These algorithms have demonstrated remarkable potential in solving problems of real world efficiently within shorter time frames [8].

Meta-heuristic algorithms are employed to search for near-optimal solutions through randomized search processes. Popular meta-heuristics are applied for the purpose of scheduling tasks such as, Particle Swarm Optimization (PSO), Simulated Annealing (SA), Genetic Algorithm (GA) and Ant Colony Optimization (ACO) are commonly used for task scheduling such as in this paper [9] a Multi-Objectives Grey Wolf Optimizer (MGWO) algorithm has been introduced with the aim of minimizing QoS objectives specifically reducing energy and latency within the Fog-broker system. However, these meta-heuristic algorithms are not without limitations, including issues with randomness, limited capability for global search, and problem of low convergence in the late iterations, often leading to suboptimal local search solutions. Moreover, achieving a balance between global and local search poses a challenge.

Scheduling based on PSO achieves better optimization performance compared to GA. PSO leverages a more intuitive computational background, exhibits faster convergence, and is easier to implement in comparison to GA. PSO demonstrates versatility in handling both discrete and continuous problems, highlighting its efficiency in conducting global search within the problem space. By prioritizing global convergence, PSO aims to discover solutions with superior fitness values. However, PSO may encounter limitations in conducting effective local search and might not allocate sufficient attention to exploring the local subspace. Consequently, as a result, there may be a greater chance of becoming trapped in local optima, leading to reduced convergence rates during later stages [10].

While numerous variants of PSO have made significant enhancements PSO, they are still unable to a successful balance between exploration and exploitation. Issues like inefficient search efficiency remain, particularly when addressing complex global optimization problems. So PSO Optimized Leader (PSO-OL) was proposed in this paper to enhance the effectiveness of traditional PSO algorithm.

The main contributions of this paper are summarized as follows:

- This paper addresses the task scheduling issue as an multi objective optimization problem in a cloud-fog system for IoT/MCS.
- To mitigate the issue of premature convergence and enhance exploitation and exploration capability to reduce the local optimal problem, this paper introduces a new variant of PSO called PSO-OL.
- The proposed PSO-OL improves the PSO approach by introducing the selection strategy which is based on fitness and distance to find the farthest-best particle, in addition to introducing the second-G_{best}. Then a new crossover operator called Dynamic Crossover Window (DCW) to find enhanced leader was proposed.
- Comprehensive experiments were carried out with five recent state of the art utilizing CloudSim simulator under 11 distinct datasets and different cloud and fog nodes to confirm the effectiveness of the proposed system in managing the task scheduling challenge. The results of the simulation showed good results compared to the five algorithms. Further demonstrated that a balance between cloud and fog nodes produces superior outcomes.

The remaining sections of the paper are structured as follows:

In Section II, an overview of related literature on scheduling problems in cloud-fog systems is provided. Section III outlines the cloud-fog architecture. Section IV introduces standard PSO method. In section V the proposed model is presented. section VI introduces the proposed PSO-OL. In Section VII, performance metrics are introduced. In Section VIII, the implementation and experimental outcomes are presented. Section IX outlines the conclusion and potential future research directions.

II. Related Work

A comprehensive review of existing research on resource management and task scheduling challenges in Cloud and Fog computing for the IoT/MCS paradigm is presented. It includes an analysis of the advantages and restrictions of each of the reviewed studies.

This study proposes a semi dynamic for real time scheduling task algorithm for IoT services in the cloud-fog scheme. Leveraging a modified genetic algorithm, the algorithm optimally assigns tasks to virtual machines based on permutations, achieving minimal execution time. Comparative evaluations demonstrate its superiority over other algorithms in terms of time of execution, makespan, failure rate, and average delay time. The proposed algorithm offers a promising solution to enhance

efficiency in cloud data centers handling IoT applications and resource utilization [11]. However, the results show no improvement in makespan despite the increase in fog and cloud nodes, and this indicates the ineffective in used for cloud and fog nodes.

The authors in this paper utilized characteristics of two meta-heuristic approaches, namely Cuckoo Search Optimization (CSO) with PSO, to create a robust framework for addressing IoT allocation requests in a cloud-fog system. CPSO is focused on improving delay, balancing the load, computation cost, and energy consumption. The simulation outcomes clearly demonstrated the superior performance of this hybrid metaheuristic algorithm compared to baseline strategies [12]. However, the concept of security was not clear in the results for use in application allocation.

This paper presents a method for enhancing the optimization of task scheduling challenges in cloud-fog environments, focusing on reducing execution time and operational expenses. The newly introduced approach, referred to as TCaS, leverages an evolutionary algorithm GA and has been assessed using 11 datasets of varying sizes. The experimental outcomes demonstrate an enhancement in achieving an equilibrium between task completion time and operational cost [13]. But they used a two-point crossover operator and this may not maintain the quality of the genes.

Data transmission within a network results in elevated latency and unreliable traffic patterns. To optimize performance effectively, the authors introduced a new combined heuristic method named Hybrid Flamingo Search and Genetic Algorithm (HFSGA) is proposed for cost-efficient QoS-aware task scheduling. The aim is to minimize cost while enhancing QoS through efficient task scheduling [14]. However, using two-point crossover without adaptive to use in the field of task scheduling may not be guaranteed to generate improved offspring

In this research, a hybrid meta heuristic method, denoted as AO_AVOA, is formulated for IoT request scheduling within fog-cloud networks. AO_AVOA harnesses the Aquila Optimizer (AO) in conjunction with the African-Vultures Optimization Algorithm (AVOA) to improve the exploration phase of AVOA. In AO_AVOA, AO is used to enhance the exploration phase of AVOA. The results demonstrate the remarkable capability of AO_AVOA in effectively addressing the scheduling challenges within IoT-fog-cloud networks [15]. Nonetheless, the fitness was single-objective, which was to depend on makespan improvement only, as this means improvement in one direction contrary to multi-objective algorithms.

This paper introduces the Energy Efficient Makespan Cost Aware Scheduling (EMCS) algorithm, employing an evolutionary approach (GA) to enhance time of execution, cost efficiency, and energy utilization. Comprehensive simulations were conducted to assess its effectiveness. Findings indicate that optimizing the balance between fog and cloud nodes as their numbers increase results in improved performance across makespan, cost, and energy consumption metrics [16], however, using the GA algorithm without any modification may not be enough to suit the topic of task scheduling in a cloud-fog system.

This paper introduced an innovative method to enhance task scheduling within a cloud-fog environment, focusing on two key aspects: execution time (makespan) and cost expenses related to bag-of-tasks applications. The proposed approach introduces a task scheduling evolutionary algorithm, which incorporates a unique problem representation and a consistent uniform intersection strategy. Additionally, specialized initialization and perturbation procedures, including crossover and mutation operations, have been developed to address situations where the evolutionary algorithm encounters impractical solutions [17], however, the uniform crossover can not ensure enhancement in offspring because it works randomly.

This paper explores the equilibrium between two prevalent and competing goals in task scheduling within the distributed fog cloud environment, namely, makespan and cost. To address this challenge, a novel hybrid algorithm, known as Hybrid-Squirrel Search and Invasive-Weed (HSSIW), is employed to efficiently allocate tasks generated by IoT devices to suitable fog and cloud nodes. The experimental evaluation conducted using CloudSim demonstrates that the proposed approach reduces makespan and cost [18], but, the authors restricted to comparing with traditional algorithms without comparing with the literature.

An in-depth examination of associated works reveals and to the best of our knowledge, the researchers did not use the PSO algorithm to find a solution to the problem of scheduling (independent) tasks in a cloud-fog system, and they also did not use the concept of the second-G_{best} individual with the G_{best} using the dynamic crossover concept. Furthermore, the incorporation of Mobile Crowdsensing into the cloud-fog system was not introduced.

III. Cloud-Fog Task Scheduling Architecture

The hierarchy of three-layered architecture is illustrated in Fig. 1, consisting of three tiers: the end layer (IoT/MCS), the fog layer, as well as cloud layer. The end layer is responsible for managing various IoT and MCS devices, such as sensors, smart vehicles, mobile phones, and smart cards, and MCS such as smartphones. This layer is widely distributed geographically and in close proximity to the users. Devices within this layer gather data from physical objects or events and transmit it to the upper layers for processing and storage. The fog layer comprises a network of gateways, switches, routers, access points, and laptops. Devices can connect with fog devices to access services. These fog devices may be stationary or mobile and are linked to the cloud layer to leverage its powerful processing capabilities and extensive storage capacity. The fog layer efficiently supports time sensitive and low delay applications. Finally, the cloud layer consists of superior servers and storage machines, providing robust computing power and storage capabilities. This layer supports various computational analyses and offers services like smart home automation and smart manufacturing applications, among others. By integrating these three layers, the hierarchical fog architecture enables seamless and efficient

data processing and service delivery, catering to the diverse needs of IoT/MCS applications [19, 20].

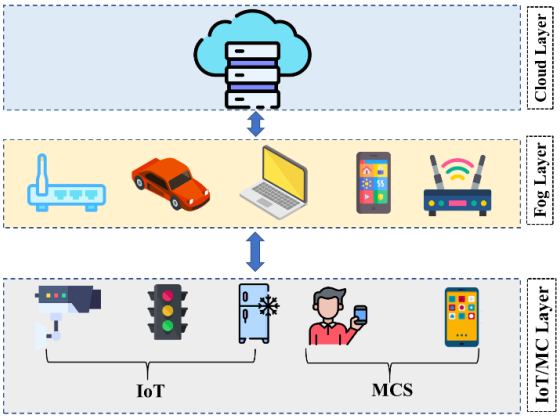

Fig. 1. Three layers architecture (cloud, fog and IoT/MCS).

IV. Standard PSO

PSO, which stands for Particle Swarm Optimization, belongs to the category of Swarm Intelligence (SI) algorithms. It draws its inspiration from the social behavior observed in animals. Initially proposed by Eberhart and Kennedy in 1995, PSO functions as a search optimization technique, seeking the best optimal solution by facilitating information sharing within the swarm of particles [21, 22].

A. Updating Particle Velocity

The control of velocity is regarded as a key component of PSO since it is the primary mechanism utilized to move a particle's position in order to find the best solution within the search space. The velocity of particle k in the population in iteration (t+1)th, is adjusted according to the following expression:

$$V_k^{t+1} = wV_k^t + c_1 r_1 \left(P_{b_k}^t - X_k^t\right) + c_2 r_2 (P_{\text{Gb}}^t - X_k^t) \quad (1)$$

where V_k^{t+1} depicts the particle k velocity in iteration (t+1)th, while X_k^t represents particle k position in iteration tth. The term $P_{b_k}^t$ refers to the particle k personal best position during the tth iteration, and P_{Gb}^t signifies the best global position within all particles in the tth iteration. The real acceleration coefficient c_1 is a cognitive coefficient and c_2 is social coefficient governs the impact of particles personal best position and global best position respectively. Additionally, to maintain diversity within the population, the uniformly distributed random numbers r_1 and r_2, both ranging $\in$ [0, 1] are used.

B. Updating Particle Position

The position for each particle k, at every iteration (t+1)th, is calculate according to (2):

$$X_k^{t+1} = X_k^t + V_k^{t+1} \quad (2)$$

where X_k^{t+1} represent particle k position in the (t+1)th iteration.

During each iteration, every individual particle adjusts its velocity and position using (1) and (2), respectively. Consequently, every particle explores the search space based on its updated velocity and position. This process persists until the particle reaches convergence towards the optimal solution.

After every particle adjusts information of its position X_k^{t+1}, calculates its fitness function value $F(X_k^{t+1})$ and compares it to the value of fitness $F_{b_k}^t$ at the particle's best historical position.

If $F(X_k^{t+1}) < F_{b_k}^t$, meaning that the current particle fitness function value is lower than the value of its fitness at its historical-best position, thereafter the algorithm updates the particles historical best position and assigns it as X_k^{t+1}. Otherwise, if the current fitness function value is not smaller, the historical particle best position remains the same. The calculation particle historical best-position can be determined as follows:

$$P_{b_k}^{t+1} = \begin{cases} X_k^{t+1};\ F(X_k^{t+1}) < F_{b_k}^t \\ P_{b_k}^t;\ F(X_k^{t+1}) > F_{b_k}^t,\ \text{otherwise} \end{cases} \quad (3)$$

Updated the particle P_{Gb} based on the comparison of the fitness function $F(X_k^{t+1})$ of particle $F_{b_k}^t$ with the fitness function value F_{Gb} of P_{Gb}.

$$P_{\text{Gb}} = \begin{cases} P_{b_k}^{t+1};\ F_{b_k}^{t+1} < F_{\text{Gb}} \\ P_{\text{Gb}};\ F_{b_k}^{t+1} > F_{\text{Gb}}\text{, otherwise} \end{cases} \quad (4)$$

The PSO algorithm pseudocode is shown in Algorithm 1 [22].

Algorithm 1: PSO Algorithm

Input: lower bound LB, upper bound UB, Fitness function, Population numbers P_n, Max iteration M_{iter}, ω, c_1, c_2 (user-defined).
Output: Solution with the best fitness.

Initialize a random population P with velocity V_k and position X_k of the kth particle within the bounds;
for $t = 1$ to M_{iter} do
  for k = 1 to P_n do
    if $F(X_k^{t+1}) < F_{b_k}^t$ then
      $P_{b_k}^{t+1} = X_k^{t+1}$
      $F_{b_k}^{t+1} = F(X_k^{t+1})$
    end
    else
      $P_{b_k}^{t+1} = P_{b_k}^t$
    end
    if $F_{b_k}^{t+1} < F_{\text{Gb}}$ then
      $P_{\text{Gb}} = P_{b_k}^{t+1}$
      $F_{\text{Gb}} = F_{b_k}^{t+1}$
    end
    Adjust the particle's velocity information based on Eq. (1).
    Adjust the particle's position information based on Eq. (2).
  end
end
Continue the process until the termination criteria are satisfied, which can occur when either the maximum number of iterations is achieved or the desired accuracy level is achieved.

V. Proposed Scheduling Model

In this section, we introduce the proposed PSO-OL approach to improve the standard PSO algorithm to handle the task scheduling challenge within virtualized fog-cloud systems as in Fig. 2. The major goal of this algorithm is to reduce the system makespan and cost while maximizing resource utilization and throughput. PSO-OL includes particle encoding, initializing population, fitness function, and crossover operator to generate a new optimized leader, in subsequent sections, we explain in-depth every stage and the entire PSO-OL methodology for addressing the specified problem.

A. Cloud-Fog-IoT Ecosystem Mechanism

In the fog environment, data processing occurs within a hub located on smart devices, smart routers, gateways, switches, laptops, and similar devices. Due to limited processing capacity, certain fog nodes collaborate within a regional context and connect to cloud nodes, commonly known as cloud virtual machines, to fulfill the requirements of mobile users. Our system assumes the presence of cloud virtual nodes C_n, fog virtual nodes F_n, cloud-fog broker broker$^{c\text{-}f}$, and IoT/MCS users $U_{\mathrm{IoT/MCS}}$. All user requests are promptly forwarded to broker$^{c\text{-}f}$. broker$^{c\text{-}f}$ plays a crucial role in resource monitoring, task monitoring, and scheduling tasks within the cloud-fog system.

In order to ensure the performance of the system, the proposed PSO-OL algorithm is implemented in broker$^{c\text{-}f}$. Its primary objective is to discover the most efficient schedule for task execution, optimizing both time and resource utilization factors. The step-by-step explanation of the ecosystem mechanism is shown in Fig. 3. The process starts by sending the task request from (service requester $U_{\mathrm{IoT/MCS}}$) which could be a person or a company to broker$^{c\text{-}f}$ (broker$^{c\text{-}f}$ installed in MCS platform). MCS platform sends requests to MCS workers and waits for a response, after getting a response broker$^{c\text{-}f}$ is responsible for executing the scheduling algorithm. This algorithm calculates the expected task completion time, considering the capabilities of the available resources and the specific requirements of each task, and allocates to suitable cloud or fog nodes. Each node is responsible for processing the received tasks and returning the results to broker$^{c\text{-}f}$. Finally, the corresponding responses are returned to the requester.

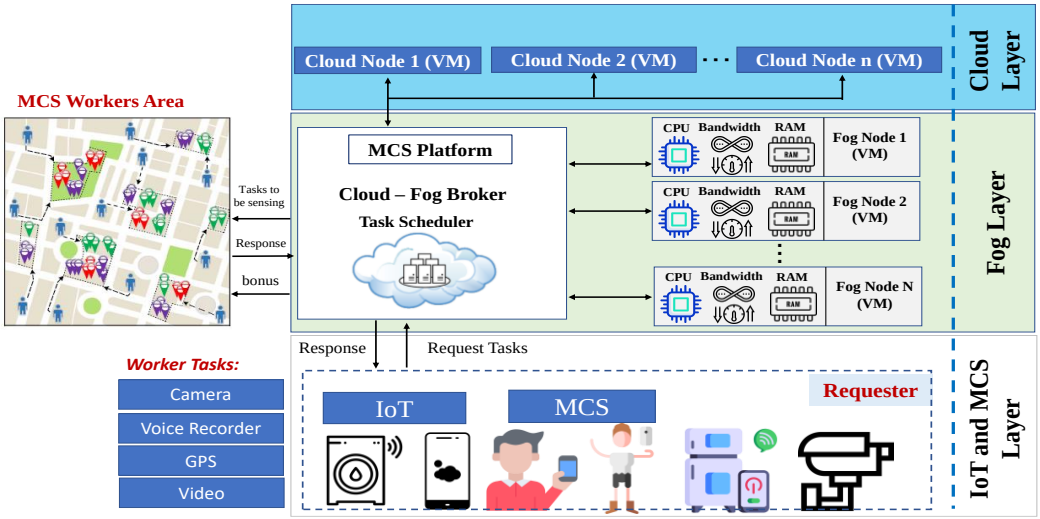

Fig. 2. Proposed fog-cloud task scheduling model for IoT/MCS environment.

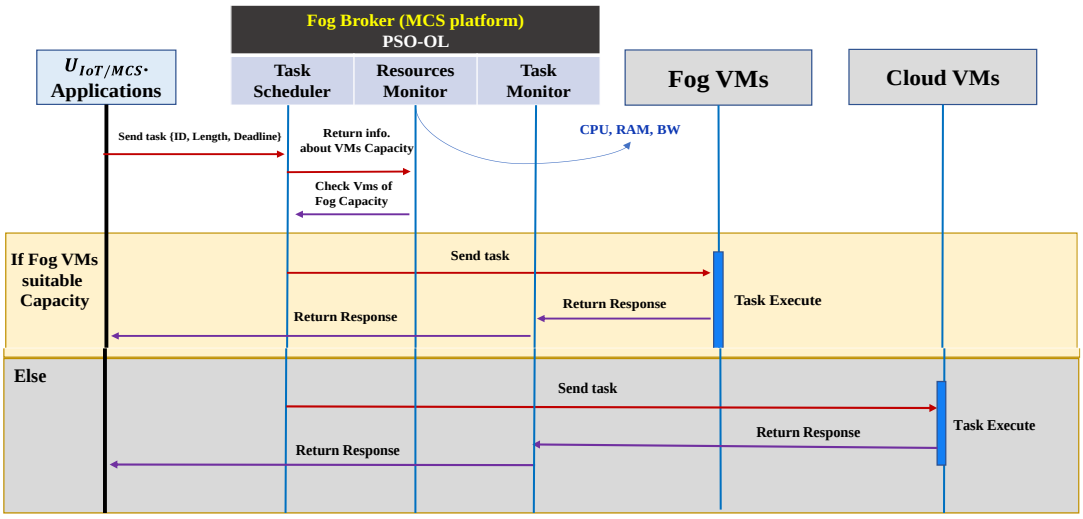

Fig. 3. The mechanism of cloud-fog system.

B. Problem Formulation

If jobs are appropriately assigned to virtual machines (VMs), cloud-fog task scheduling can be effective and achieve high performance. When requests from $U_{\mathrm{IoT/MCS}}$ applications are sent to broker$^{c\text{-}f}$, for processing across the cloud-fog system. This explanation assumes that the cloud-fog system is housed in a datacenter with a variety of servers, each of which hosts a number of virtual machines VMs. Let's assume there are *n* tasks, and they are as follows:

$$T = \{T_1, T_2, T_3, \dots, T_n\} \tag{5}$$

T_i represents the *i*th independent task, $i \in \{1,2, ..., n\}$. The attributes of each task T_i consist of various factors such as $\{T_{\mathrm{id}}$, task length, memory demand, file sizes for the input/output$\}$. The task length is measured in terms of Millions-of-Instructions (MI).

The cloud-fog infrastructure is composed of CPUs, denoted as cloud-nodes C_n and fog-nodes F_n, each of which has multiple characteristics, including various processor powers, different bandwidth, memory sizes, and storage capacity. Consider a set of *R* nodes from cloud-fog defined as follows: The collection of processors *m* encompassing both C_n and F_n in the system, $R = \{C_n \cup F_n\}$ is formulated as:

$$R = \{R_1, R_2, R_3, \dots, R_m\} \tag{6}$$

where R_j denotes the virtual processing for the *j*th virtual node. Each T_i is assigned to only one virtual node R_j, that denoted by T_i^j. Every *R* node can be allocated a set of tasks $R_j (j = 1, 2, 3, \dots, m)$ as illustrated in the following:

$$R_j^T = \{T_x^j, T_y^j, \times\times\times, T_z^j\} \tag{7}$$

In general, cloud nodes tend to be more powerful compared to fog nodes. However, utilizing cloud nodes typically incurs higher costs in comparison.

Our proposed model consists of a limited number of R_j which are heterogeneous virtual nodes, each of which has a different ability to run a given task.

The time taken for task *i* to execute on R_j is represented as execution time ET_{ij} and can be determined using the following mathematical expression [23]:

$$\mathrm{ET}_{ij} = \frac{L(\mathrm{T}_i)}{N_{\mathrm{PEj}} \times \mathrm{R}_{\mathrm{MIPS}_j}} \tag{8}$$

where $L(\mathrm{T}_i)$ is the task length measured in MI, N_{PEj} refers to the count of processing elements in R, and $\mathrm{R}_{\mathrm{MIPS}_j}$ is the processing speed of *R*, expressed in million-instructions-per-second (MIPS).

Eq. (9) computes total execution time TET_{R_j} for executing set of tasks in R_j:

$$\mathrm{TET}_{R_j} = \sum_{i=0}^{n} \mathrm{ET}_{ij} \tag{9}$$

Let's consider that makespan refers to the total time needed to finish a set of tasks on R_j. The makespan can be calculated using the following [24]:

$$\mathrm{Makespan} = \mathrm{Max}_{1 \le j \le m}[\mathrm{TET}_{R_j}] \tag{10}$$

The $\mathrm{Cost}(T_i^j)$ can be defined as the total cost associated with completing the T_i^j on R_j. This cost encompasses the processing cost $C_p(T_i^j)$, memory required cost $C_m(T_i^j)$, and bandwidth usage cost $C_B(T_i^j)$.

The calculation of $\mathrm{Cost}(T_i^j)$ is described as follows [11]:

$$\mathrm{Cost}(T_i^j) = C_p(T_i^j) + C_m(T_i^j) + C_B(T_i^j) \tag{11}$$

The three costs mentioned above are defined as follows:

$$C_p(T_i^j) = \mathrm{Cost}_{\mathrm{CPU}-j} \times \mathrm{ET}_{ij} \tag{12}$$

$$C_m(T_i^j) = \mathrm{Cost}_{M-j} \times \mathrm{Memory}\,(T_i^j) \tag{13}$$

$$C_B(T_i^j) = \mathrm{Cost}_{B-j} \times \mathrm{Bandwidth}(T_i^j) \tag{14}$$

where $\mathrm{Cost}_{\mathrm{CPU}-j}$ is the cost associated with utilizing the CPU for task execution on node R_j within a specific time, Cost_{M-j} represents the memory cost which usage in node R_j, $\mathrm{Memory}\,(T_i^j)$ denotes how much memory T_i consumed in node R_j, Cost_{B-j} refers to the cost of utilizing bandwidth, and bandwidth (T_i^j) represents the required bandwidth for transferring T_i to be executed on R_j.

The total cost of executing all tasks within a cloud-fog environment can be expressed in the following:

$$\mathrm{Total}_{\mathrm{Cost}} = \sum_{T_i^j \in T^{\mathrm{node}}} \mathrm{Cost}(T_i^j) \tag{15}$$

Table I presents the essential symbols, terms, and concepts used in the proposed scheduling model.

TABLE I: PSO-OL NOTATIONS USED

Notation	Definition
Cloudlet	Task representation in CloudSim.
VM_s	Virtual-Machines.
MI	Million-Instructions data of cloudlet.
MIPS	Million-Instructions Per Second.
$N_{\mathrm{PE}\,j}$	Processing elements number in VM.
$\mathrm{R}_{\mathrm{MIPS}_j}$	Speeds of VM.
$\mathrm{VM}_{\mathrm{id}}$	The identification number of VM.
T_{id}	Identification number of cloudlet.
S_T	Number of tasks that have been completed successfully.
C_n	Cloud virtual nodes.
F_n	Fog virtual nodes.
$\mathrm{broker}^{\mathrm{C-F}}$	Cloud-Fog broker.
$U_{\mathrm{IoT/MCS}}$	IoT/MCS users
t	Iteration.
P_n	Population or individuals number.
P^{dim}	Particle dimension.
M_{iter}	Maximum iteration.
P_{sim}	Similarity factor between particles
d_{af}	Dynamic adjustment factor.
cr	Crossover rate.
ⓧ	crossover operation

C. Particle Encoding for Task Scheduling

Adhering to the core concept of population based algorithms, each particle within the population is characterized as a scheduling solution. In PSO-OL a particles position is represented as an array, indicating the assignment of T_i to R_j. The length of this array determined

by the particle dimension $P^{\dim}$, relates to the overall count of tasks as illustrated in Fig. 4.

Every element contains the R_j index for executing its corresponding T_i. For instance, if position[$P^{\dim}$]=5, it implies that task T_i will be executed on R_5. The particles minimum and maximum positions range from 1 to the total number of R that is m, respectively.

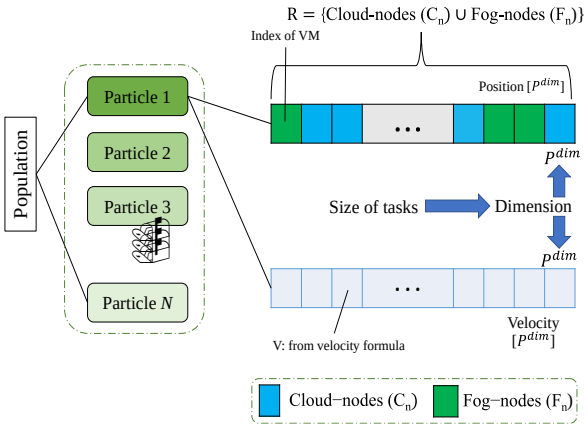

Fig. 4. Particles encoding representation for proposed task scheduling problem.

The particle velocity is represented as an array of the same size as the position array. Within each iteration, each element includes the result of a velocity update equation. The result is then utilized to adjust the index of a VM in order to execute the associated task.

In a scenario where n tasks need to be scheduled across m nodes in the cloud-fog, will be P_n particle with a dimension of n that means $P^{\dim}$. To represent the position information of each particle in the particle swarm, use the following:

$$x_i = \{x_{i1}, x_{i2}, \dots, x_{in}\}; \quad \forall 1 \le i \le m \tag{16}$$

This indicates that every x_i describes a feasible solution for the PSO method, whereas x_{i1} represents that the ith task is allocated to the resource x_{i1}. The velocity of particles is defined as:

$$v_i = \{v_{i1}, v_{i2}, \dots, v_{in}\}; \ \forall 1 \le i \le m \tag{17}$$

Assuming that there are eight particles in the population and that the information of particle position is represented as (5, 7, 3, 4, 2, 1, 2, 6, 4), this means that the particle has nine dimensions, Table II. displays the task allocation to resources as a particle.

TABLE II: A SCHEME FOR ASSIGNING TASKS

Tasks (dimensions)	1	2	3	4	5	6	7	8	9
Resources (VMs)	5	7	3	4	2	1	2	6	4

D. Fitness Function

The fitness function calculates the optimal value for each individual element referred to as personal best fitness $F_{b_k}^t$ as in (18) in the case of a minimization or maximization problem. The fitness value among whole individuals is denoted as the global best fitness P_{Gb}^t as in (19).

$$F_{b_k}^t = \min(F_{b_k}) \tag{18}$$

$$P_{\text{Gb}}^t = \min(F_{b_k}^t) \tag{19}$$

In this paper, we formulated a fitness function that incorporates two important metrics as in Eq. (20):

$$F(x) = \alpha \times \text{Makespan} + \beta\left(\frac{1}{\text{RU}}\right) \tag{20}$$

where the fitness function $F(x)$ = Minimization (Objective), $\beta = 1 - \alpha$ and

$$\text{RU} = \frac{\sum_{j=1}^{m} ET_j}{\text{Makespan} \times m} \tag{21}$$

This function aims to achieve a balance between the normalized makespan and resource utilization of a fog-cloud devices trade-off. In this paper, we transformed the problem into a single-objective optimization problem by means of a weighted sum approach.

The primary goal of the algorithm introduced in this paper is to allocate resources to tasks efficiently, aiming to maximize resource utilization and minimize the makespan (i.e., Maximize RU; Minimize Makespan).

α and β are the trade-off coefficients between the makespan and resource utilization and $\alpha + \beta = 1$. When the value of α is greater than 0.5, this means the β is less than 0.5 and the task assignment approach places a priority on decreasing makespan over overall resource utilization. Conversely, if α is less than 0.5, makespan becomes less significant compared to resource utilization. When both makespan and resource utilization are given equal priority, the value of α is set to 0.5 in this work.

VI. PROPOSED PSO OPTIMIZED LEADER (PSO-OL)

In traditional PSO, a significant challenge arises. The concern pertains to the potential premature convergence due to all particles gravitating toward the swarm leader and if it becomes stuck in a local optimum, all the particles will move closer to it, thus, the PSO may easily get trapped in premature convergence. The diversity of the swarm as it moves across the solution space has an impact on the solution quality. High particle diversity in the early stage is sought for the greatest solution space to locate a good seed of search. All particles in PSO are naturally drawn towards the swarm leader P_{Gb}. Therefore, having a high-quality leader can greatly enhance the efficiency of the search process, much like how a competent leader in a society or organization can lead to greater success. PSO-OL represents a new variant of PSO that distinguishes itself by continually enhancing the swarm leader at every iteration of the search process.

The PSO technique is enhanced by incorporating four primary modifications in the proposed method PSO-OL. Firstly, a method to prevent the similarity between particles to get diversity in initial stage is introduced. Secondly, a strategy to select a particle from the population in each iteration, namely, Farthest-best (F_b), is introduced based on selecting a particle to use with P_{Gb} to find a new enhanced leader after applying the proposed new crossover on them, which considers both the current fitness value of

particle and their distance to P_{Gb} within the present swarm, and we designed a new dynamic adjustment factor to control on the selection of particle. This strategy balances an exploration by giving a distance a high impact in early iterations and works on enhancing the exploitation by a focus on fitness in the latest iterations. Third, when all particles converge to one position, the search process no longer evolves. To overcome this challenge, in this paper, we focus on utilizing information from the second-best particle in addition to P_{Gb} by using the crossover between them to find an optimized leader that has an updated position vector that will combine information from the G_{best} and the second-G_{best}. Finally, proposed a new crossover operator: In PSO-OL, a successive crossover strategy is applied to get a better new swarm leader at each iteration based P_{Gb}, F_b and second-G_{best}. After finding the F_b and second-G_{best} particles, the proposed crossover is applied to generate offspring then choose the best one to become as leader in each iteration.

This process iterates until a predefined termination condition is met, which may involve reaching an acceptable solution or exceeding the maximum iteration limit. The best solution discovered during these iterations is then presented as the final result.

A. Proposed Initial Population Method PPIS

The algorithm convergence rate is affected by population diversity which can be achieved in the initial stage. The primary phase in the optimization process of metaheuristic techniques involves establishing the initial population. To initialize the swarm, a vector of P particles is generated within a number of population P_n. For every particle, the initial position and velocity are randomly generated in standard PSO.

In this paper, the population is created randomly, and then apply the proposed method to Prevent Particle Index Similarity (PPIS). After the generation of all particles, start checking the indexes for instance, particle P_2 with the corresponding indexes of P_1 for all genes in the particle if the similarity factor P_{sim} is greater than 50%, then the particle is rejected. Otherwise, the replacement mechanism will apply as in (22). This means examining each individual with the previous one. As for setting the P_{sim} to 50%, it is for the purpose of reducing the number of replacements.

Replacing is used to achieve diversity for mapping VMs in the initial phase. This means if the index in particle P_2 is X_i, then should be not particle P_1 index is the same value in X_i, if is same index value is replaced with X_{i+1} value as shown in Fig. 5, and Algorithm 2.

$$P_{i+1} = \begin{cases} \text{Reject; if } P_{\text{sim}} > 50\% \\ \text{Accepte; replace as: new } X_i = X_{i+1} \end{cases} \quad (22)$$

The position of every particle is assigned to its related local best solution P_{b_k}. The lowest value of $F_{b_k}^t$ among entire particles is set to the F_{Gb} of the population. Furthermore, the essential parameters of PSO are defined, including the inertia weight w, learning factors c_1, c_2, and the random variables within the range [0,1]. Moreover, the rand() is the random function to generate random numbers ∈ [0,1], and M_{iter} defines the maximum iteration number. Subsequently, the fitness value of whole particles is calculated using (20) to obtain the fitness for each particle.

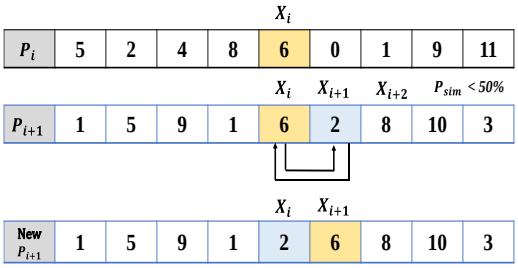

Fig. 5. The proposed initial population method PPIS example.

Algorithm 2: Prevent particle index similarity PPIS in the initial population.

Input: P_n, $P^{\dim}$, k=0.
Output: Dissimilar individuals in the population.
Initialize position and velocity randomly, Rand-F ∈ (0, m).
  int [][] particle = generate P_n particles with $P^{\dim}$∈ Rand-F [P_n][$P^{\dim}$];
  /*Apply PSPI procedures*/
for i=1 to P_n do
  for j = 0 to $P^{\dim}$ do
    if (particle[i][j] == particles[i–1][j]) then
      k++
  end
  if (k > $P^{\dim}$/2) then /*means k >50%*/
    Reject particle[i]
    Generate new random individual
  end
  else
    for j = 0 to $P^{\dim}$ do
      if (particles[i][j] = = particles[i–1][j]) then
        particles[i][j] = particles[i][j+1]
      end
    end
  end
end
Note: in the same way repeat steps from 2 to 17 to check all particles.

B. Farthest-Best Particle Mechanism (FbPM)

If the entire population has already converged to a small area, easy to trap in a local optimum, in this case, we introduced another particle in the search space called Farthest-best (F_b). F_b is the farthest distance from P_{Gb} with good fitness and can explore another region and avoid repeated searching in the subspace that the G_{best} has already searched.

In this strategy, the P_{Gb}individual selects one particle in each iteration from the population that is F_b which is determined by fitness value and the farthest distance value between P_{Gb} and each particle in population. This paper utilizes the Euclidean distance difference to find F_b particle. Eqs. (23) and (24) represent the P_{Gb} and F_b information respectively.

When determining the F_{Gb} and updating the fitness of each particle in population, then, calculated the Euclidean

distance $D(P_{\mathrm{Gb}} - P_i)$ between the $P_{\mathrm{Gb}}$and ith individual as stated in Eq. (25).

$$\mathrm{Gb}_1 = \{P_{\mathrm{Gb}},\ F_{\mathrm{Gb}}\}, \qquad \forall\ 1 < P_{\mathrm{Gb}} < m \tag{23}$$

$$F_b = \{P_{\mathrm{Fb}}\ ,\ F_{\mathrm{Fb}}\ \}, \qquad \forall\ 1 < P_{\mathrm{Fb}}\ < m \tag{24}$$

where Gb_1: a global best particle. {position, fitness}, P_{Fb}: Farthest-best position, and F_{Fb}: Farthest-best fitness.

$$(D_{P_{\mathrm{Gb}}-P_i}) = \sqrt{\left(P_{\mathrm{Gb}}^1 - x_i^1\right)^2 + \left(P_{\mathrm{Gb}}^2 - x_i^2\right)^2 + \ldots + \left(P_{\mathrm{Gb}}^{P\dim} - x_i^{P\dim}\right)^2}$$
$$\forall\ 0 < i < P_n \tag{25}$$

The vector of distance $(\mathbf{D}_{P_{\mathrm{Gb}}-}\mathbf{P}_i)$ and the vector of fitness $\mathbf{F}_P$ which are generated in each iteration for candidates solution of global best and particles as given in (26) and (27).

$$(\mathbf{D}_{P_{\mathrm{Gb}}-}\mathbf{P}) = \left[D_{P_{\mathrm{Gb}}-}P_1, D_{P_{\mathrm{Gb}}-}P_2\ , \ldots\ ,\ D_{P_{\mathrm{Gb}}-}P_{P_n}\right] \tag{26}$$

$$\mathbf{F}_P = \left[\mathrm{F}_{\mathrm{P}_1}, \mathrm{F}_{\mathrm{P}_2}, \ldots, \mathrm{F}_{P_n}\ \right] \tag{27}$$

This method calculates the distance only from P_{Gb} to particles, which reduces the complexity of the calculation when compared to a method that computes the distance between each particle and all other particles in the population.

The combined rating based fitness-distance (R_{fd}) of every particle is calculated, and select particle depending on which one receives the highest rating.

R_{fd}represents the index of a chosen example, created through the implementation of the R_{fd} strategy. Initially, the fitness and distance vectors of data are normalized as follows:

$$N_{(\mathbf{D}_{P_{\mathrm{Gb}}-}\mathbf{P}_i)} = \frac{D_{P_{\mathrm{Gb}}-P_i} - \min(\mathbf{D}_{P_{\mathrm{Gb}}-}\mathbf{P})}{\max(\mathbf{D}_{P_{\mathrm{Gb}}-}\mathbf{P}) - \min(\boldsymbol{D}_{P_{\mathrm{Gb}}-}\boldsymbol{P})} \tag{28}$$

$$N_{(\mathbf{F}_{P_i})} = \frac{F_{P_i} - \min(\mathbf{F}_P)}{\max(\mathbf{F}_P) - \min(\mathbf{F}_P)} \tag{29}$$

where min(), and the max() represent the minimum and maximum values within vector $\mathbf{F}_P$ and vector $\mathbf{D}_{P_{\mathrm{Gb}}-}\mathbf{P}$, respectively.

Now we designed a new dynamic adjustment factor d_{af} as in (30)

$$d_{\mathrm{af}}(t+1) = d_{\mathrm{af}}(t) + \left(\frac{0.5}{M_{\mathrm{iter}}}\right) \tag{30}$$

$d_{\mathrm{af}}(t)$: initial value is 0.5. where $0 \le t \le M_{\mathrm{iter}}$

Equation (31) can be used to compute the R_{fd} of particles with P_{Gb}, resulting in a symmetric matrix. Then, the individual with the maximum rating M–R_{fd} within the matrix is selected, as specified in Eq. (32).

$$R_{\mathrm{fd}} = d_{\mathrm{af}}(1 - N_{(\mathbf{F}_{\mathbf{P_i}})}) + (1 - d_{\mathrm{af}})N_{(\mathbf{D}_{P_{\mathrm{Gb}}-P_i})} \tag{31}$$

$$M\text{-}R_{\mathrm{fd}} = \max(R_{\mathrm{fd}}) \tag{32}$$

By dynamically adjusting the weight of d_{af}. The d_{af} value starts with a small value of 0.5, and after increasing the iteration it will gradually increase until it reaches 1 at the last iteration M_{iter}, and then there will be no effect on the distance, and this is good for focusing on the diversity when the small value (enhancing exploration within the population). When a large value, it focuses on fitness to enhance exploitation and in the end, when the value becomes equal to 1, the focus becomes entirely on fitness, meaning that the algorithm emphasizes exploitation to search for the best solutions as shown in Table III.

TABLE III: EFFECTIVE OF d_{af} VALUE

d_{af} **Value**	**Probability of focusing on fitness**	**Probability of focusing on diversity**
0.5	Equal	Equal
$d_{\mathrm{af}}(t)$		
.	.	.
.	.	.
.	.	.
$d_{\mathrm{af}}(M_{\mathrm{iter}}/2)$	High	Low
.	.	.
.	.	.
1	100%	0%
$d_{\mathrm{af}}(M_{\mathrm{iter}})$		

C. Second Global-Best Particle (SGbP)

PSO stands as a population-centered optimization technique that strives to strike a balance between exploration which involves seeking out new promising areas, and exploitation, which focuses on refining the best-discovered solution. The best particle represents the current global best solution, while the second-best particle may represent a different solution that is also quite good.

Nonetheless, if a particle exclusively acquires knowledge from the farthest individual, it will exhibit erratic oscillations, making it challenging to attain an improved solution. Consequently, a learning direction from the second-best particle is introduced to ensure that particles do not significantly diverge during exploration.

The concept of using both the P_{Gb} and a second-best with the crossover process in swarm intelligence and optimization algorithms represents a new approach to enhance the search for optimal solution. This innovative strategy leverages the strengths of two distinct leaders within a population of agents, allowing for more diverse exploration and potentially leading to the discovery of even better solutions.

We need to determine the fitness and positions of first (Gb_1) (Gb_1) as in (23) and second-best (Gb_2) as in (33) and Fig. 6.

$$Gb_2 = \{P_{\mathrm{Gb}^s}, F_{\mathrm{Gb}^s}\}, \forall\ 1 <\ P_{\mathrm{Gb}^s} < m \text{ and } F_{\mathrm{Gb}^s} > F_{\mathrm{Gb}} \tag{33}$$

Gb_2: is a second global best particle with $\{P_{\mathrm{Gb}^s}$ as position, and F_{Gb^s} as fitness$\}$.

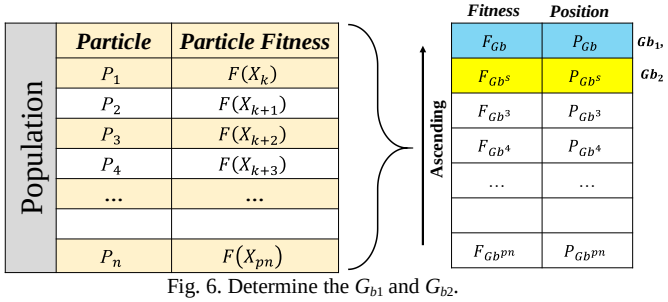

Fig. 6. Determine the G_{b1} and G_{b2}.

D. Proposed Crossover Operator DCW

The crossover is considered one of the important GA operators [25]. The crossover operator aims to generate new particle by changing the position inside two particles. The crossover operator can improve the sharing of information between particles and prevent the early convergence of a swarm. The new crossover mechanism is proposed in this paper to get an optimized leader.

In this work, a Dynamic Crossover Window (DCW) is proposed as shown in Algorithm 3 and Fig. 7 with a full example of the parents that inherit the quality genes in offspring. The crossover operation depends on d_{af} value as in subsection (B) and (30). We need to define the dynamic factor (D_{fa}) as in (34) based value of d_{af} that starts from 0.5 to 1 in the last iteration.

$$D_{\mathrm{fa}} = 1 - d_{\mathrm{af}} P^{\mathrm{dim}},\ \forall\ 0.5 < d_{\mathrm{af}} \leq 1 \qquad (34)$$

The $1 - d_{\mathrm{af}}$ is for starting the crossover segment small in early iteration and increases gradually. This is because in the initial iterations, the value of d_{af} is impact on the distance, and thus there will be less focus on fitness. Therefore, we need to reduce the window so that the genes are not changed significantly and the quality of the particle is maintained, while when the iterations progress, the choice of the F_b particle becomes completely dependent on the fitness value, and in this way a good particle is chosen. That's why we increase the window of change. The limit l_m as in (35) is to find the difference between $P^{\mathrm{dim}}$and D_{fa} to use in (36) to find crossover segment C_s. C_s is determines how much the size of window.

$$l_m = P^{\mathrm{dim}} - D_{\mathrm{fa}} \qquad (35)$$

$$C_s = 2l_m - P^{\mathrm{dim}} \qquad (36)$$

Equation (37) is to determine the range of start C_s, where S_{rp} is represent the start point in range [0, H_r] randomly as in (38):

$$H_r = |\, P^{\mathrm{dim}} - C_s \,| \qquad (37)$$

$$S_{\mathrm{rp}} = \mathrm{rand}\,[0, H_r] \qquad (38)$$

While the end crossover segment E_{cs} in (39) is to determine the end of C_s. The gens between S_{rp} and E_{cs} are selected based uniform crossover with rate (cr) (which is set to 0.8) between two particles while the remaining genes (out of the range) are select from global best particles, the DCW is applied on P_{Fb} and P_{Gb^s} with P_{Gb} in the two stages.

$$E_{\mathrm{cs}} = [S_{\mathrm{rp}} + C_s\,] \qquad (39)$$

So then, C_s in interval $[\mathrm{S_{rp}}, \mathrm{E_{cs}}]$.

After getting a F_b particle, if F_{Fb} is better than F_{Gb}, then P_{Fb} takes the position of P_{Gb}.

However, if it is not a better, the crossover applied between P_{Fb} and P_{Gb} particles to find a new particle we called C_{GbF} as in (40). If the $F(C_{\mathrm{GbF}})$ is better than F_{Gb}, then C_{GbF} takes the position of P_{Gb}.

$$C_{\mathrm{GbF}} = P_{\mathrm{Fb}} \text{Ⓧ} P_{\mathrm{Gb}} \qquad (40)$$

where Ⓧ is a crossover operation.

If $F(C_{\mathrm{GbF}})$ is not better than F_{Gb}, the crossover applied between P_{Gb^s} and P_{Gb} particles as in Eq. (41).

$$C_{\mathrm{Gbs}} = P_{\mathrm{Gb}^s} \text{Ⓧ} P_{\mathrm{Gb}} \qquad (41)$$

If the $F(C_{\mathrm{Gbs}})$ is better than F_{Gb}, then C_{Gbs} takes the position of P_{Gb}. Otherwise it will update velocity based on P_{Gb}.

- *Illustrative Example of DCW*

Consider number of cloudlets is 100, total number of VMs $\{C_n + F_n\}$ is 15 and number of iterations is set to 150, then when $1 - d_{af}$ is 0.4833333333 in iteration 4 where $d_{\mathrm{af}} > 0.5$, the D_{fa} will be 48, l_m is 52, C_s is 4, H_r is 96, S_{rp} is in range [0, 96] random, then S_{rp} is 64 select randomly within range and E_{cs} is 68.

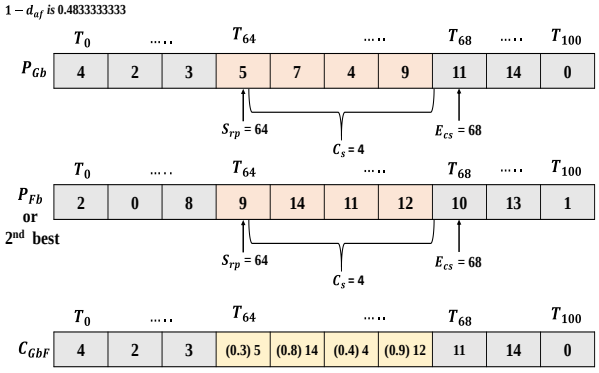

Fig. 7. Illustrative example of proposed DCW.

Algorithm 3: Proposed crossover operator DCW

Input: $G_{b1} = \{P_{\mathrm{Gb}},\ F_{\mathrm{Gb}}\}$, $F_b = \{P_{\mathrm{Fb}},\ F_{\mathrm{Fb}}\}$, $G_{b2} = \{P_{\mathrm{Gb}^s},\ F_{\mathrm{Gb}^s}\}$, cr.
Output: Enhanced leader (particle) C_{GbF} and C_{Gbs}

1: Find dynamic factor D_{fa} based value of d_{af} as in (34)
2: Find The l_m as in (35) is to find the deference between P^{dim} and D_{fa}.
3: Calculate C_s to determine size of window based (36)
4: Determine the range of start C_s, where S_{rp} is represent the start point in range [0, H_r] based (37) and (38)
5: Determine the E_{cs} using (39)
6: C_{GbF} = Array with size P^{dim}
7: C_{Gbs} = Array with size P^{dim}
8: for j = 0 to P^{dim}–1 do
9: if $j \geq S_{\mathrm{rp}}$ and $j < E_{\mathrm{cs}}$ then
10: if random ∈ (0, 1) < cr then
11: $C_{\mathrm{GbF}}[j] = P_{\mathrm{Gb}}[j]$
12: end
13: else
14: $C_{\mathrm{GbF}}[j] = P_{\mathrm{Fb}}[j]$
15: end
16: else
17: $C_{\mathrm{GbF}}[j] = P_{\mathrm{Gb}}[j]$
18: end
Note: in the same way repeat steps from 7 to 18 to get C_{Gbs}

E. Complete Model for PSO-OL Scheduler

The flowchart of the proposed scheduling approach PSO-OL is presented in Fig. 8 and the pseudocode is shown in Algorithm 4. Firstly, the particles are initialized according to the proposed PPIS strategy as in subsection (A). Then, find F_b particle based R_{fd} with d_{af} as in subsection (B), and then find the second-G_{best} as in subsection (C). The proposed crossover DCW is used with these two particles to generate an optimized leader in each iteration.

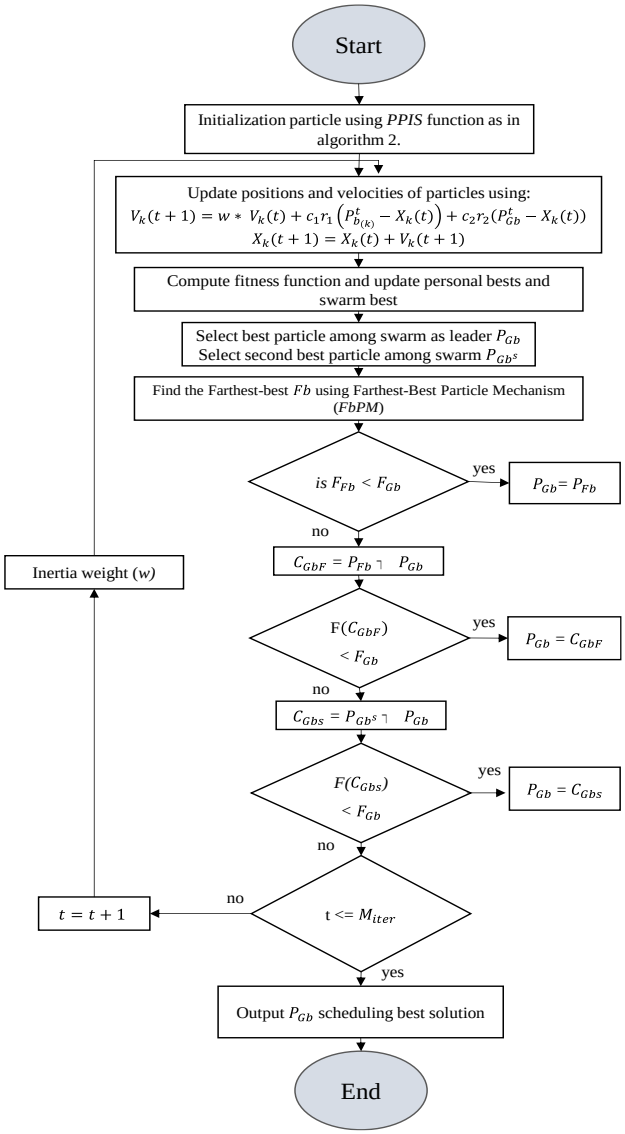

Fig. 8. Flowchart of the proposed PSO-OL.

Algorithm 4: Proposed PSO-OL scheduler algorithm

Input: T: List of tasks, $T = \{T_1, T_2, T_3, \dots, T_n\}$ with its $\{L(\mathrm{T}_i)$ in MI, input/output file sizes$\}$, R: List of cloud-fog nodes, $R = \{R_1, R_2, R_3, \dots, R_m\}$ with its $\{\mathrm{R}_{\mathrm{MIPS}_j}$, $N_{PE\ j}$ RAM, BW$\}$, LB, UB, P_n, M_{iter}, $\omega, c1, c2$.

Output: Best mapping of T on R.

```
for k = 1 to P_n do
    Initialize a population P with velocity V_k and position X_k of the kth particle within the bounds using PPIS function as in Algorithm 2.
    Compute F(x) based (20) and update personal bests and swarm best.
    Set best position of particle as P^t_{b_k}
    Select best particle among swarm as leader P_Gb
    Select second best particle among swarm P_Gb^s
end
t=1
while (t <= M_iter) do
    for k = 1 to P_n do
        Compute new velocity using V_k^{t+1} = wV_k^t + c_1 r_1 (P^t_{b_k} - X_k^t) + c_2 r_2 (P^t_Gb - X_k^t)
        if V_k^{t+1} not in range then
            Put it in range according to LB and UB
        end
        Compute new position using X_k^{t+1} = X_k^t + V_k^{t+1}
        Evaluate fitness of all particles using F(x) as per (20)
        if F(X_k^{t+1}) < F^t_{b_k} then
            P^{t+1}_{b_k} = X_k^{t+1}
            F^{t+1}_{b_k} = F(X_k^{t+1})
        end
        else
            P^{t+1}_{b_k} = P^t_{b_k}
        end
        if F^{t+1}_{b_k} < F_Gb then
            P_Gb = P^{t+1}_{b_k}
            F_Gb = F^{t+1}_{b_k}
        end
        Find farthest-best F_b using farthest-best particle mechanism (FbPM)
        if F_Fb < F_Gb then
            P_Gb = P_Fb
        else
            Apply crossover using DCW as in Algorithm 3
            C_GbF = P_Fb ⓧ P_Gb
        end
        if F(C_GbF) < F_Gb then
            P_Gb = C_GbF        /*become C_GbF as leader*/
        else
            C_Gbs = P_{Gb^s} ⓧ P_Gb
            if F(C_Gbs) < F_Gb then
                P_Gb = C_Gbs    /*become C_Gbs as leader*/
            end
        end
    end
    t++
end
Continue the process until the termination condition is met, which can occur when either the maximum number of iterations is reached or the desired accuracy level is achieved.
```

VII. PERFORMANCE METRICS

Four evaluation metrics that used to test and evaluate the performance of the proposed MODEL-PSO in a cloud-fog system. These metrics include makespan, cost, throughput, and the Performance Improvement Ratio (PIR). It is important to highlight that in the existing literature, most researchers commonly rely on makespan or in some cases

cost as the primary measure to evaluate the effectiveness of their algorithms. However, in the case of the PSO-OL approach, the evaluation encompasses the following metrics:

A. Makespan

Makespan is the primary objective and widely used metric for evaluating the efficiency of scheduling in a cloud-fog for IoT/MCS environment. It can be defined as the completion time of the last executed task. A smaller makespan indicates that the broker has effectively allocated tasks to the appropriate VMs. The makespan is mathematically represented as provided in Eq. (10).

B. Cost

The cost in cloud-fog task scheduling refers to the expenses associated with processing a received task from IoT/MCS users to the cloud is determined by factors such as the demand cost for processing the incoming task, which can be calculated based on the cost of utilizing the CPU, memory expenses, and the cost incurred by bandwidth consumption. Essentially, it reflects the financial aspects of executing tasks efficiently within the cloud-fog system. The cost of task execution when task T_i runs on virtual machine R_j can be computed using Eq. (11), and the overall cost can be calculated using Eq. (15).

C. Throughput

The throughput in the context of cloud-fog task scheduling refers to the rate at which tasks are processed and completed within the Cloud-Fog. It measures the efficiency of the scheduling system in terms of how many tasks it can successfully execute in a given makespan. Higher throughput indicates that the system can handle a larger volume of tasks, improving overall performance and responsiveness. Essentially, throughput is a critical metric for evaluating the efficiency of task scheduling algorithms in cloud-fog systems and is calculated as follows [26].

$$\text{Throughput} = \frac{\sum S_T}{\text{Makespan}} \tag{42}$$

where S_T represents the count of tasks that have been completed successfully.

D. Performance Improvement Ratio (PIR%)

The PIR metric is utilized to evaluate the effectiveness of a PSO-OL approach by considering the reduction in execution time. Consequently, it is regarded as an important measure for evaluating the algorithm's effectiveness. The mathematical calculation of the PIR metric is presented as follows [27].

$$\text{PIR\%} = \frac{\text{Makespan}_x - \text{Makespan}_P}{\text{Makespan}_x} \times 100\% \tag{43}$$

where Makespan_x is the makespan obtained from xth algorithm and Makespan_P is the makespan obtained from the proposed algorithm.

VIII. Experimental Results and Analysis

In this section, a set of experiments was performed in order to evaluate the effectiveness of the PSO-OL scheduling algorithm in comparison to state-of-the-art algorithms, both heuristics and metaheuristics. The assessment covered several aspects, including makespan, total execution cost, throughput, and PIR.

A. Experimental Setup

In order to evaluate the effectiveness of the proposed approach and analyze the results, CloudSim is an open-source framework tool that was developed in Java [28, 29] and is used for the purpose of establishing and simulation an integrated cloud-fog environment, encompassing cloud and fog nodes, along with a constrained number of service requests pertaining to IoT applications. All experiments were conducted on a single machine with the characteristics displayed in Table IV.

TABLE IV: Simulation Environment Attributes

Variable	Characteristics
Simulator version	CloudSim V-5.0
JDK	11
Machine characteristics	Intel(R)-Core i7-10750 H, CPU (2.6 GHz), (2.59) GHz. 16 GB Memory and HD one TB

The cloud-fog system is accountable for executing all requests from IoT/MCS users. Cloud and fog nodes vary in terms of their processing capabilities and resource usage costs. We assumed regarding the processing capabilities of each node which is represented by their processing rate measured in MIPS. Additionally, we considered factors like CPU, RAM, and bandwidth usage costs.

In the fog layer, nodes like routers, gateways, workstations, or personal computers have limited processing capacity. On the other hand, in the cloud layer, servers or virtual machines located in high-performance data centers handle tasks. Consequently, cloud nodes have significantly higher processing speeds compared to fog nodes. In contrast, utilizing resources in the cloud is more expensive compared to the fog. These expenses are quantified using Grid Dollars (G$), a virtual currency used in the simulation to represent real-world monetary costs.

B. Analysis of the Experimental Results

In this subsection, we present the experiments performed to evaluate the PSO-OL scheduling method and provide an analysis of the obtained results in two scenarios.

- *Scenario 1: Effective of PSO-OL in Optimize Leader*

To evaluate the effectiveness of the proposed method PSO-OL in finding the best leader due to the fact that all particles are attracted toward the swarm leader. So, having a high-quality leader can make the search process more efficient, and this is reflected in improving the fitness function and thus improving the overall scheduling system. In this case, we tried 400 tasks and 25 VMs. The number of particles is 100 and the number of total iterations is 250 as shown in Table V. The results for 10 runs show the effectiveness of the proposed method in finding the best leader in each iteration compared to the traditional PSO algorithm as shown in Fig. 9, where in each run the number of optimized leaders is illustrated in the y-axis. It is clear that the proposed method for finding the best leader is very effective, as it overcame the traditional PSO algorithm in

finding a leader with a better fitness function, as in Fig. 10, which shows the percentage of improvement.

TABLE V: EFFECTIVENESS TEST ATTRIBUTES

Variable	Parameters
No. of tasks	400
No. of VMs	25
No. of particles	100
Total iterations	250
No. of run	10

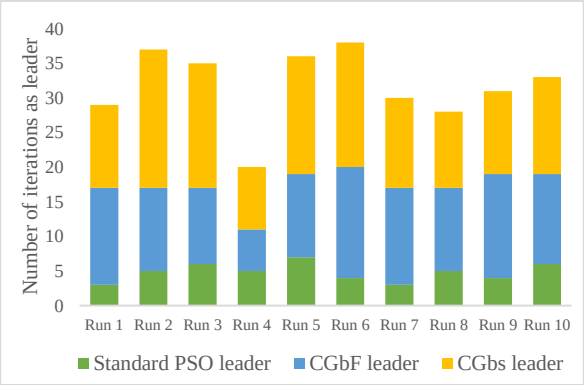

Fig. 9. The optimized leaders.

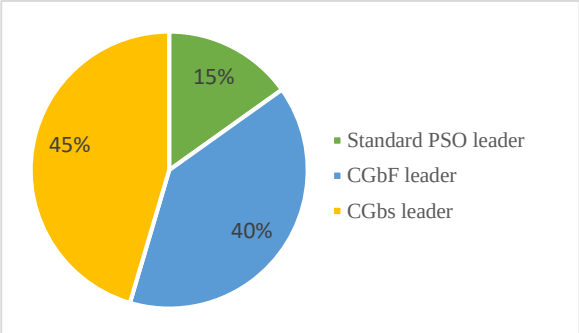

Fig. 10. The percentage of leaders improvement.

- *Scenario 2: Cloud-fog task scheduling*

To evaluate the effectiveness of the PSO-OL algorithm, we conducted comparisons with existing methods from the literature such as Evolutionary-Task Scheduling algorithm (ETS) [17], A Time Cost aware Scheduling (TCaS), Modified Particle Swarm Optimization (MPSO) and Round-Robin (RR) [13] and Bee Life Algorithm (BLA) [30]. Table VI and Table VII illustrate the cloud and fog properties respectively.

TABLE VI: CLOUD NODES ATTRIBUTES

Variable	Cloud parameters	Unit
Nodes number	3	node
CPU speed	[3000, 5000]	MIPS
The cost of CPU utilization	[0.7–1.0]	G$/s
cost for Memory utilization	[0.02–0.05]	G$/MB
cost for Bandwidth utilization	[0.05–0.1]	G$/MB

TABLE VII: FOG NODES ATTRIBUTES

Variable	Fog parameters	Unit
Nodes number	10	node
CPU speed	[500–1500]	MIPS
The cost of CPU utilization	[0.1–0.4]	G$ /s
The cost of Memory utilization	[0.01–0.02]	G$ /MB
The cost of Bandwidth utilization	[0.01–0.02]	G$MB

For simulation, a total of 11 datasets were generated, with varying number of tasks varying from 40 to 500 tasks. These tasks within the datasets were created using random generation, and their attributes were determined in accordance with the specifications shown in Table VIII. Table IX presents algorithm attributes.

TABLE VIII: IOT/MCS TASKS ATTRIBUTES

Variable	Parameters	Unit
Length	[1000–100000]	MI
Memory required	[50–200]	MB
Size of input file	[10–100]	MB
Size of output file	[10–100]	MB

TABLE IX: ALGORITHMS ATTRIBUTES.

Variable		BLA	TCaS	MPSO	ETC	PSO-OL
Run count		30	30	30	10	30
Population size (ℕ)	queen	1	100	100	100	100
	drones (D)	30				
	Workers (W)	69				
Crossover rate (α)		90%	90%	-	0.5	0.8
Mutation-rate (γ)		0.01	0.01	-	0.1, 0.3	-
c_1, c_2, w		-	-	c1 = c2 = 1.5 w = 0.9–0.1	-	c1 = c2 = 1.5 w = 0.9–0.1
Iterations number		500	500	500	500	500

In this scenario, we have preserved a fixed number of VMs,10 as cloud, and 3 as fog), while the processing capacity required to complete each user-assigned task may vary. We generated a set of tasks randomly, with the numbers of tasks ranging from 40 to 200, increasing in intervals of 40, and from 200 to 500 increasing in intervals of 50 using a random dataset.

For the purpose of evaluating the performance of the proposed PSO-OL approach in this scenario, we have taken multiple metrics into consideration. These metrics are makespan, Cost, throughput, and performance improvement ratio.

Fig. 11 illustrates the result of the comparison of the proposed approach PSO-OL in the makespan measure. There was a total of 13 VMs utilized in this experiment, and the system was received 40, 80, 120, 160, 200, 250, 300, 350, 400, 450, and 500 of IoT/MCS tasks. After testing the random datasets over 500 generations and in order to mitigate the influence of uncertain factors on the experimental results, every experiment is carried out 30 times, and take the average to provide a more reliable assessment. The experimental results demonstrate that the PSO-OL approach achieved a more notable reduction in the average makespan when compared to the RR, BLA, MPSO, ETS, and TCaS.

In our experiments, we set a value of α and β to 0.5, meaning that makespan and resource utilization have the same importance in the fitness function. The PSO-OL approach achieves the shortest makespan by optimizing

the time needed for task completion time. This, in turn, leads to a higher level of performance for the PSO-OL approach, which is achieved by efficiently mapping tasks onto VMs. We conclude, that the PSO-OL demonstrates a significant advantage over all other algorithms when applied to large datasets. As a result, the algorithm's superiority in terms of makespan becomes evident, especially in scenarios involving limited resources and large task datasets.

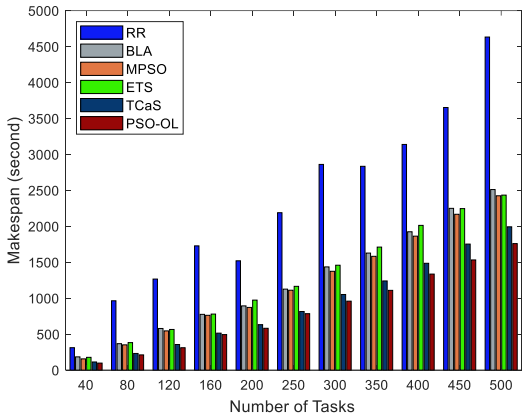

Fig. 11. Comparison results of makespan in scenario 2.

From Fig. 12, it can be concluded that the PSO-OL approach showed remarkable performance from the makespan perspective on each dataset indicating less execution time, therefore, this leads to the whole cost being significantly less in the PSO-OL approach than in comparison algorithms. For instance, the cost of RR, BLA, MPSO, and TCaS was 63026 G$, 62468 G$, 61090 G$, and 63750 G$, while that of the PSO-OL approach was 53987 G$ for the 500 tasks. As the task count decreased, also there was a reduction in the overall cost of the algorithms. For instance, the overall cost of the RR, BLA, MPSO, TCaS, and PSO-OL was 18664 G$, 18324 G$, 17838 G$, 19043 G$, 16562 G$, respectively, for the 250 tasks. The primary factor contributing to cost reduction is the increased utilization of fog nodes for task execution, as opposed to relying heavily on cloud nodes. This cost-saving advantage stems from the inherently lower cost of using fog resources compared to cloud resources. Consequently, this approach lowers the expenses associated with crowdsensing users requesting cloud-fog services.

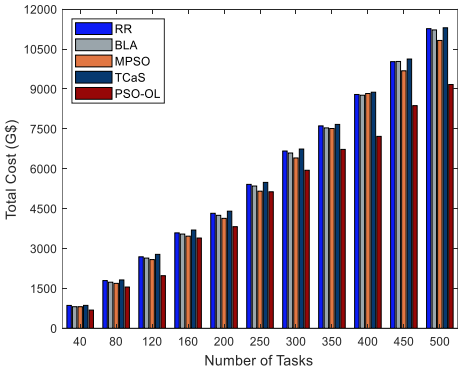

Fig. 12. Comparison results of cost in Scenario 2.

Fig. 13 illustrates a comparison of throughput achieved by the PSO-OL algorithm, alongside RR, BLA, MPSO, ETS, and TCaS algorithms, using a synthetic dataset. The horizontal axis represents the number of IoT/MCS tasks, while the vertical axis signifies the throughput parameter. The simulation results demonstrate that the PSO-OL algorithm outperformed all other algorithms in terms of throughput. The experimental results displayed that the PSO-OL algorithm achieves a better throughput 163.07%, 42.71%, 37.76%, 38.35%, and 13.29% when compared with other algorithms (i.e., RR, BLA, MPSO, ETS, and TCaS) in number of tasks 500. These results confirm the PSO-OL algorithm is effectiveness and this indicates the stability of the proposed method and concluded that the proposed approach effectively balances the workload, ensuring that no single server becomes overloaded.

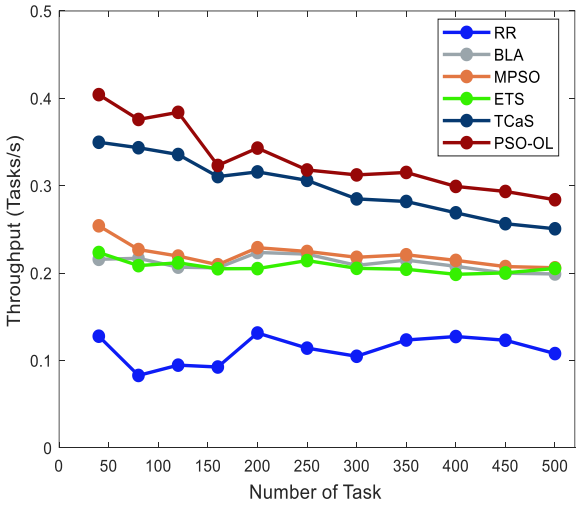

Fig. 13. Comparison of throughput in Scenario 2.

The percentage of PIR for our approach PSO-OL depending on makespan as it relates to the RR, BLA, MPSO, ETS, and TCaS algorithm is presented in Fig. 14 for the random workload, the results demonstrate that the PSO-OL approach produces 68.38–62.00, 46.62–29.92, 37.14–27.42, 44.69–27.69 and 13.48–11.71 makespan time improvements over the RR, BLA, MPSO, ETS, and TCaS respectively in case of task a 40 and 500.

In terms of the number of tasks that have been executed, we find that the fog nodes execute a larger number of the tasks received by them compared to the cloud nodes, and this is good to think about introducing the concept of fog with the cloud, which is reflected in a significant reduction in the cost, because the costs of using fog VMs are cheaper than the costs of using cloud VMs as shown in the Fig. 15.

The superior performance of PSO-OL in comparison to other algorithms can be attributed to its ability to avoid getting trapped in local optima. This is achieved through the mentioned modifications which strengthened the exploration and exploitation phases, enabling it to find optimal solutions within a reasonable timeframe for task scheduling.

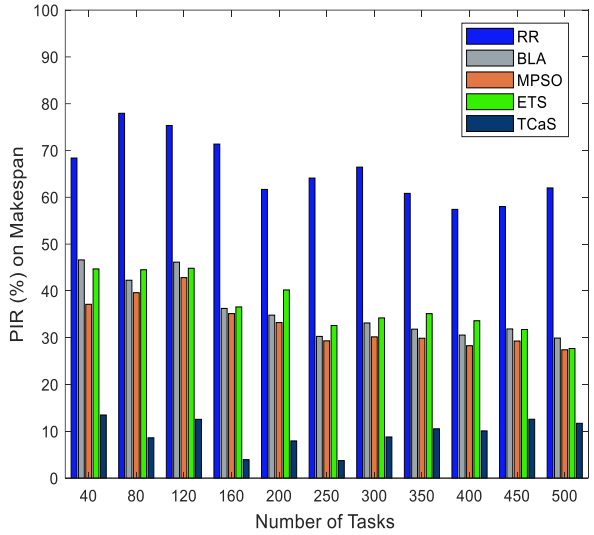

Fig. 14. PIR (%) on makespan comparison in Scenario 2.

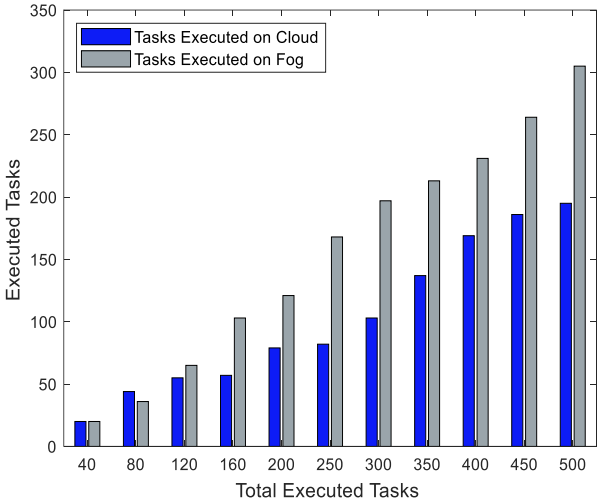

Fig. 15. Tasks executed in each layer for PSO-OL.

IX. Conclusions and Future Works

This paper proposed a new variant of PSO to address task scheduling challenges in cloud-fog computing for IoT/MCS applications. The cloud-fog broker in this work was responsible for analyzing, estimating, and scheduling all transmission requests generated from edge devices to be executed within the cloud-fog system. The proposed PSO-OL algorithm performed in the scheduling of tasks, and the results indicate that the proposed algorithm exhibits improved global search capability compared to the traditional PSO algorithm. This enhancement helps prevent the traditional PSO algorithm from getting trapped in local optima. PSO-OL was proposed with the objective of minimizing makespan, improve resource utilization and increasing throughput to enhance QoS goals. To evaluate the effectiveness of the proposed PSO-OL algorithm in achieving these objectives, a simulation was conducted, comparing its performance with that of existing state-of-the-art algorithms. The results from the simulation demonstrated that the proposed PSO-OL outperformed the competing algorithms. This demonstrates its capability to effectively manage the significant increases in request generation from IoT devices and effectively allocate tasks to resources. The proposed approach can also be applied in areas other than task scheduling. As a future work, this work can be enhanced by introducing machine learning techniques in addition to using other optimization techniques with PSO, also applied to real-world data sets.

Conflict Of Interest

The authors Abbas M. Ali Al-muqarm and Dr. Naseer Ali Hussien declare there are no conflicts of interest.

Author Contributions

Conceptualization, Naseer Ali Hussien, and Abbas M. Ali Al-muqarm; the methodology, Abbas M. Ali Al-muqarm; software, Abbas M. Ali Al-muqarm; wrote the manuscript, Abbas M. Ali Al-muqarm; writing review and editing, Abbas M. Ali Al-muqarm and Naseer Ali Hussien; proofread the paper, Naseer Ali Hussien.

References

[1] S. P. C. Rao and M. Sushama, “An IoT-Based Sensor Technology for Improving Reliability and Power Quality in Smart Grid Systems,” Int. J. Electr. Electron. Eng. Telecommun, vol. 12, no. 4, July 2023. doi: 10.18178/ijeetc.12.4.264-271

[2] S. H. Supangkat, R. Ragajaya, and A. B. Setyadji, “Implementation of digital geotwin-based mobile crowdsensing to support monitoring system in smart city,” *Sustainability*, vol. 15, no. 5, #3942, 2023.

[3] W. Liu, C. Li, A. Zheng, Z. Zheng, Z. Zhang, and Y. Xiao, “Fog computing resource-scheduling strategy in IoT based on artificial bee colony algorithm,” *Electronics*, vol. 12, no. 7, #1511, 2023.

[4] A. R. Kadhim and F. Rabee, “Deadline and cost aware dynamic task scheduling in cloud computing based on stackelberg game.,” *Int. J. Intell. Eng. Syst.*, vol. 16, no.3, 2023 doi: 10.22266/ijies2023.0630.14.

[5] A. Khan, A. Abbas, H. A. Khattak, F. Rehman, I. U. Din, and S. Ali, “Effective task scheduling in critical fog applications,” *Sci. Program.*, vol. 2022, 2022, doi.org/10.1155/2022/9208066.

[6] A. M. Ali Al-muqarm and N. Ali Hussien, “Resource management techniques in cloud-fog for iot and mobile crowdsensing environments,” *Int. J. Electron. Telecommunications*, 2023, vol. 69, no. 2, PP. 341-352.

[7] Hosseinzadeh, M., Azhir, E., Lansky, J. *et al.*, “Task scheduling mechanisms for fog computing: A systematic survey,” *IEEE Access*, vol. 11, pp. 50994-51017, 2023.

[8] M. Abdel-Basset, N. Moustafa, R. Mohamed, O. M. Elkomy, and M. Abouhawwash, “Multi-objective task scheduling approach for fog computing,” *IEEE Access*, vol. 9, pp. 126988–127009, 2021.

[9] F. A. Saif, R. Latip, Z. M. Hanapi, and K. Shafinah, “Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing,” *IEEE Access*, vol. 11, pp. 20635–20646, 2023.

[10] S. Nabi, M. Ahmad, M. Ibrahim, and H. Hamam, “AdPSO: adaptive PSO-based task scheduling approach for cloud computing,” *Sensors*, vol. 22, no. 3, #920, 2022.

[11] A. S. Abohamama, A. El-Ghamry, and E. Hamouda, “Real-time task scheduling algorithm for IoT-based applications in the cloud–fog environment,” *J. Netw. Syst. Manag.*, vol. 30, no. 4, #54, 2022.

[12] K. Dubey, S. C. Sharma, and M. Kumar, “A secure IoT applications allocation framework for integrated fog-cloud environment,” *J. Grid Comput.*, vol. 20, no. 1, #5, 2022.

[13] B. M. Nguyen, H. Thi Thanh Binh, T. The Anh, and D. Bao Son, “Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud–fog computing environment,” *Appl. Sci.*, vol. 9, no. 9, #1730, 2019.

[14] S. M. Hussain and G. R. Begh, “Hybrid heuristic algorithm for cost-efficient QoS aware task scheduling in fog–cloud environment,” *J. Comput. Sci.*, vol. 64, #101828, 2022.

[15] Q. Liu, H. Kosarirad, S. Meisami, K. A. Alnowibet, and A. N. Hoshyar, “An optimal scheduling method in iot-fog-cloud network

using combination of Aquila optimizer and African vultures optimization," *Processes*, vol. 11, no. 4, #1162, 2023.
[16] R. Sing, S. K. Bhoi, N. Panigrahi, K. S. Sahoo, M. Bilal, and S. C. Shah, "EMCS: An energy-efficient makespan cost-aware scheduling algorithm using evolutionary learning approach for cloud-fog-based IoT applications," *Sustainability*, vol. 14, no. 22, #15096, 2022.
[17] M. N. Abdulredha, A. A. Bara'a, and A. J. Jabir, "An evolutionary algorithm for task scheduling problem in the cloud-fog environment," *Journal of Physics: Conference Series*, vol. 1963, no. 1, #12044, 2021.
[18] A. Tsegaye and B. G. Assefa, "HSSIW: Hybrid squirrel search and invasive weed based cost-makespan task scheduling for fog-cloud environment," in *Proc. of 2021 Int. Conf. on Information and Communication Technology for Development for Africa*, 2021, pp. 160–165.
[19] R. M. Singh, L. K. Awasthi, and G. Sikka, "Techniques for task scheduling in cloud and fog environment: a survey," in *Proc. of Second Int. Conf.*, Chandigarh, India, 2019, pp. 673–685.
[20] V. K. Singh, A. S. Jasti, S. K. Singh, and S. Mishra, "Quad: A quality aware multi-unit double auction framework for iot-based mobile crowdsensing in strategic setting," *arXiv Prepr, arXiv2203.06647*, 2022.
[21] E. Alkayal, "Optimizing resource allocation using multi-objective particle swarm optimization in cloud computing systems." University of Southampton, 2018.
[22] S. H. Anbarkhan and M. A. Rakrouki, "An enhanced PSO algorithm for scheduling workflow tasks in cloud computing," *Electronics*, vol. 12, no. 12, #2580, 2023.
[23] A. M. A. Al-muqarm and N. A. Hussien, "Dynamic cost-optimized resources management and task scheduling with deadline constraint for mobile crowd sensing environment.," *Int. J. Intell. Eng. Syst.*, vol. 16, no. 3, 2023, doi: 10.22266/ijies2023.0630.16.
[24] I. Z. Yakubu and M. Murali, "An efficient meta-heuristic resource allocation with load balancing in IoT-Fog-cloud computing environment," *J. Ambient Intell. Humaniz. Comput.*, vol. 14, no. 3, pp. 2981–2992, 2023.
[25] F. Rabee and Z. M. Hussain, "Oriented crossover in genetic algorithms for computer networks optimization," *Information*, vol. 14, no. 5, #276, 2023.
[26] M. Agarwal and G. M. S. Srivastava, "Opposition-based learning inspired particle swarm optimization (OPSO) scheme for task scheduling problem in cloud computing," *J. Ambient Intell. Humaniz. Comput.*, vol. 12, no. 10, pp. 9855–9875, 2021.
[27] M. Agarwal and S. Gupta, "An adaptive genetic algorithm-based load balancing-aware task scheduling technique for cloud computing.," *Comput. Mater. Contin.*, vol. 73, no. 3, 2022, doi: 10.32604/cmc.2022.030778.
[28] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. Pract. Exp.*, vol. 41, no. 1, pp. 23–50, 2011.
[29] A. Malav, S. K. Gupta, S. K. Mahariya, K. Joshi, R. Bhauguna, and S. Verma, "Optimal resource management in cloud computing," in AIP Conference Proceedings, 2023, vol. 2771, no. 1. doi.org/10.1063/5.0152298
[30] S. Bitam, S. Zeadally, and A. Mellouk, "Fog computing job scheduling optimization based on bees swarm," *Enterp. Inf. Syst.*, vol. 12, no. 4, pp. 373–397, 2018.

Abbas M. Ali Al-muqarm received a B.Sc. degree in computer science from the University of Kufa, Najaf, Iraq in 2013 and an M.Sc. degree in computer science from the Faculty of Computer Sciences and Mathematics, University of Kufa, Najaf, Iraq in 2020. He is currently a Ph.D. student. His current research interests include networking, Internet of Things (IoT), wireless sensor networks, Mobile Crowdsensing, Cloud computing, and Fog computing.

Naseer Ali Hussien is currently working as faculty member in Alayen University, has completed his Ph.D. degree in mobile ad hoc network area from University Utara Malaysia in 2013. His current research interest in wireless networks, Internet of Things, Network performance, cloud computing, Network Applications.