**DOI**: https://doi.org/10.52756/ijerr.2023.v33spl.005 ***Int. J. Exp. Res. Rev.*, Vol. 33: 41-56 (2023)**

*Original Article* | *Peer Reviewed* | *Open Access*

International Journal of Experimental Research and Review (IJERR)
© Copyright by International Academic Publishing House (IAPH)
ISSN: 2455-4855 (Online)
www.iaph.in

# AGWO: Cost Aware Task Scheduling in Cloud Fog Environment Using Hybrid Metaheuristic Algorithm

**Medishetti Santhosh Kumar* and Ganesh Reddy Karri**

VIT-AP University, Amaravathi, Andhra Pradesh, India

**E-mail/Orcid Id:**

*MSK*, santhosh.21phd7113@vitap.ac.in, https://orcid.org/0000-0001-5936-6582;
*GRK*, guncity11@gmail.com, https://orcid.org/0000-0002-5157-8125

**Article History:**

**Received:** 26th Jul., 2023
**Accepted:** 18th Sep., 2023
**Published:** 30th Sep., 2023

**Keywords:**
Internet of Things, grey wolf optimisation, ant colony optimisation, cloud-fog computing

**Abstract:** In IoT concepts, efficient approaches like cloud-fog computing are emerging, enhancing system benefits. The performance and output of such frameworks can be greatly improved by optimized scheduling of Internet of Things (IoT) task requests. This study presents a novel technique for scheduling Internet of Things requests in a cloud-fog environment, based on an adaption of ant grey wolf optimisation (AGWO). By combining the operators of ant colony optimisation (ACO) and grey wolf optimisation (GWO), AGWO aims to improve the speed and quality of ACO's solution discovery. The suggested AGWO approach is evaluated using numerous datasets of varying sizes, both synthetic and real-world. The effectiveness of AGWO is further investigated by comparing it to standard metaheuristic methods. The experimental results demonstrate that AGWO is superior to competing strategies in terms of makespan time reduced by 42% and we reduced the cost by 36% while resolving the task scheduling problem.

## Introduction

In the past several years, the Internet of Things (IoT) has emerged as a major trend in the information technology (IT) sector (Nguyen et al., 2019; Ghasempour, 2019). Other smart gadgets, security systems, machineries, cameras, sensors, vehicles, and more are all incorporated into the IoT along with laptops, tablets, and smartphones. The primary objective of the Internet of Things is to facilitate a wide range of applications and services, such as those in the transportation and traffic management sectors, the healthcare industry, the energy sector, the healthcare sector, the vehicle network sector, and the industrial sector (Fu et al., 2018; Zuo et al., 2015). The data these programmes produce is enormous, and it takes time and effort to organise, store, and analyse it so that it may be used to serve users and their goals better (Lin et al., 2016). Even the most sophisticated devices lack the processing power to keep up with the growing number and diversity of Internet of Things (IoT) applications (Chen et al., 2018).

An IoT community can be fostered implicitly by using a cloud infrastructure. Cloud computing (CC) is commonly understood to be a massive data centre that provides users with easy virtual access to both receive and supply resources (Liu et al., 2018). The storage, battery life, network capacity, and processing constraints of today's smart gadgets can be overcome with the use of powerful computing environments like cloud and fog computing. This optimisation entails outsourcing time-consuming operations to these computer environments while giving smart devices reasonable workloads. Fog computing (FC) is an important part of IoT ecosystems since it is a distributed computing paradigm (Abualigah and Diabat, 2020). It functions as a separate computer model by relying on fog nodes to facilitate service availability, data preservation, and communication broadcasting. By allocating data centre resources in accordance with where mobile users are, fog computing bolsters cloud computing's utility (Vijayalakshmi et al., 2020). For fog nodes, the major goal of job scheduling is to maximise efficiency and performance. Fog computing allows faster data transfer and less network congestion (Wang et al., 2018). But with these improvements come new difficulties in coordinating activities and allocating resources.

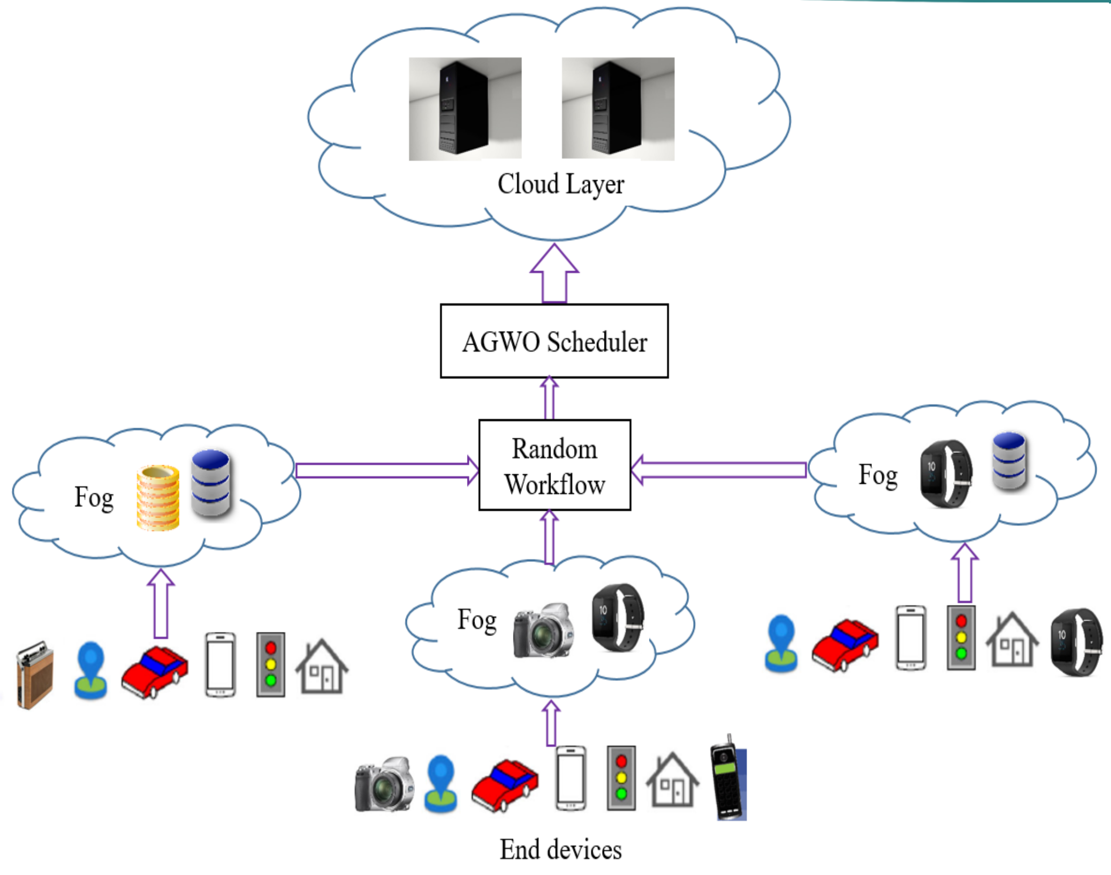

**Figure 1. System architecture**

As shown in Figure 1, the fog computing architecture is a three-tiered network. Cloud servers that store and analyse massive amounts of data are located on the top layer, which represents the core zone of cloud computing. In the middle layer, located there specifically to aid mobility, are fog computing zones filled with fog machines. The next layer is the Internet of Things devices zone (Yang et al., 2020; Ghasempour and Moon, 2016), which includes sensors, laptops, cell phones, automobiles, and personal computers used by end users.

Using a hybrid metaheuristic algorithm like AGWO (Ant Grey Wolf Optimisation) for cost-aware task scheduling in a cloud fog scenario can be difficult for a number of reasons. Some of the main problems with this strategy are as follows:

1. Problem Complexity: Task scheduling in a cloud fog environment is inherently complex due to the dynamic nature of the environment, the heterogeneity of resources, and the presence of multiple objectives and constraints. AGWO needs to address these complexities and find efficient scheduling solutions while considering cost factors. However, modelling and representing the problem accurately within the AGWO framework can be challenging, requiring careful consideration of various factors and trade-offs.

2. Algorithm Parameter Tuning: AGWO involves several parameters that need to be carefully tuned for optimal performance. Determining the appropriate population size, convergence criteria, search operators, and other parameters can be non-trivial. The effectiveness and efficiency of AGWO heavily depend on finding the right parameter settings, which often require extensive experimentation and expertise. Inadequate parameter tuning can lead to suboptimal results or even algorithm failure.

3. Scalability: When working with huge quantities of cloud fog, the scalability of AGWO presents considerable difficulty. Finding optimal solutions becomes computationally demanding when the number of activities and resources grows exponentially. AGWO's population-based approach can help distribute the search process, but efficiently handling massive problem sizes and resource constraints remains challenging.

5. Balance between Exploration and Exploitation: AGWO needs to strike a balance between exploration and exploitation to effectively search the solution space. Exploration involves searching for new and potentially better solutions, while exploitation focuses on refining and improving existing solutions. Achieving the right balance is crucial to avoid getting trapped in local optima and to discover high-quality scheduling solutions. Designing mechanisms that dynamically adjust the exploration-exploitation trade-off can be challenging and requires careful consideration.

6. Incorporating Real-Time Factors: In a cloud fog environment, real-time factors such as task arrivals, resource availability, and changing workload patterns need to be considered. AGWO should adapt and respond to these dynamic factors to ensure timely and efficient task scheduling. Incorporating real-time information into the algorithm and developing strategies to handle dynamic environmental changes pose significant challenges.

7. Benchmarking and Comparative Evaluation: It is crucial to compare AGWO's performance with other existing algorithms and approaches to assess its effectiveness. Conducting fair and comprehensive benchmarking studies can be challenging due to the diversity of problem instances, performance metrics, and evaluation criteria. It requires designing appropriate test scenarios, selecting suitable benchmarks, and ensuring consistency in experimental setups.

The AGWO (Ant Grey Wolf Optimization) algorithm is a hybrid metaheuristic approach proposed for cost-aware task scheduling in a cloud fog environment. It combines artificial intelligence techniques and nature-inspired optimization algorithms to address the challenges of task scheduling while considering cost factors. The cloud fog environment refers to a distributed computing paradigm that extends the capabilities of cloud computing by incorporating fog nodes, which are located closer to the edge of the network. Fog nodes provide resources and services to nearby devices, enabling faster response times and reducing network congestion. distributing tasks to appropriate resources, including fog node clusters and cloud servers, is an important part of task scheduling in cloud fog systems. However, cost factors, including energy consumption, resource utilization, and communication overhead, must also be considered to achieve cost efficiency. The AGWO algorithm takes inspiration from the hunting behavior of grey wolves in nature. Grey wolves exhibit collaborative hunting strategies, combining individual exploration with coordinated pack movements. AGWO emulates these behaviors by using a population of ant grey wolves to search for optimal task-scheduling solutions. Each grey wolf in the algorithm's initial population stands for a different possible answer. These solutions are scored using a fitness function that takes into account both efficiency metrics (like makespan) and performance metrics (like energy usage and resource utilization).

AGWO utilizes a hybrid approach, combining global search and local search mechanisms. The global search mechanism allows grey wolves to explore the entire search space, while the local search mechanism focuses on refining solutions in the local neighborhood. This combination of exploration and exploitation enhances the algorithm's ability to find high-quality scheduling solutions. Cost awareness is achieved by incorporating cost factors into the fitness evaluation process. By considering cost metrics alongside performance metrics, AGWO aims to find solutions that minimize the overall cost while meeting the performance requirements. AGWO aims to optimize task scheduling by reducing the makespan (total time taken to complete all tasks) and minimizing the associated costs. It leverages the hybrid metaheuristic approach to handle cloud fog environments' complex and dynamic nature. The motivation behind developing AGWO (Ant Grey Wolf Optimization) for cost-aware task scheduling in a cloud fog environment using a hybrid metaheuristic algorithm stems from several key factors:

1. Cost Efficiency: Cloud fog environments involve the utilization of resources distributed across cloud servers and fog nodes. Efficiently scheduling tasks in such environments requires considering cost factors such as energy consumption and resource utilization. The motivation behind AGWO is to develop a task scheduling algorithm that explicitly incorporates cost awareness, aiming to minimize the overall cost while meeting performance objectives.

2. Heterogeneity and Dynamism: Cloud fog environments are characterized by the presence of heterogeneous resources and dynamic workloads. Fog nodes located at the edge of the network have limited resources compared to cloud servers. Task arrivals, resource availability, and workload patterns can also change rapidly. AGWO's motivation lies in addressing the challenges posed by the heterogeneity and dynamism of the environment, aiming to optimize task scheduling decisions accordingly.

3. Performance Optimization: While cost reduction is a primary objective, AGWO also focuses on optimizing performance metrics such as makespan. Efficient task scheduling ensures that tasks are allocated to suitable resources to minimise the total time taken to complete all tasks. AGWO aims to find scheduling solutions that strike a balance between cost reduction and performance optimization, thus enhancing the overall efficiency of task execution in cloud fog environments.

4. Nature-Inspired Optimization: AGWO takes inspiration from the hunting behavior of grey wolves. Grey wolves exhibit collaborative and adaptive strategies for successful hunting. The motivation behind adopting a nature-inspired optimization approach is to develop an algorithm that emulates the wolves' behaviors to

effectively explore the solution space and find high-quality task scheduling solutions.

5. Hybrid Metaheuristic Approach: AGWO integrates global and local search mechanisms to exploit their strengths. In order to find potential regions, the global search mechanism looks across the entire solution space, while the local search mechanism concentrates on improving solutions that are close to the best ones. Improving the algorithm's exploration and exploitation skills leads to more optimal or near-optimal scheduling solutions, which is why the hybrid metaheuristic approach was developed.

By incorporating cost-awareness, addressing the heterogeneity and dynamism of cloud fog environments, optimizing performance metrics, and employing a hybrid metaheuristic approach, AGWO aims to provide an effective and efficient solution for cost-aware task scheduling. The motivation is to contribute to the development of scheduling algorithms that meet the unique challenges and requirements of cloud fog environments, ultimately improving resource utilization, reducing costs, and enhancing overall system performance (Abualigah et al., 2020; Wang et al., 2020).

IoT contexts have varying user requirements, and fog computing platforms outperform cloud computing in terms of performance. Therefore, a fundamental difficulty needs to be addressed: optimally scheduling activities in fog computing by efficiently distributing resources and aligning them with user demands.

First, the paper's key contributions are discussed, and second, the paper's core evaluation process and comparisons are outlined.

We propose the following improvements to the current state of the art in order to enhance IoT services in a cloud-fog computing environment:

(i) To provide efficient task scheduling in cloud-fog environments, we present a hybrid evolutionary method named AGWO. This algorithm combines the foundational GWO methodology with the adaptive ACO strategy to increase convergence speed and searching explorations.

(ii) Quality of service (QoS) and overall cost are two competing criteria that must be taken into account when designing a cost model for Internet of Things (IoT) tasks transmitted for processing in a cloud-fog framework.

(iii) Comparing the makespan and task execution cost of the proposed scheduling strategy to those of other approaches applied to realistic workloads.

The outline of this paper looks like this:

The article continues with the following structure. Section 2 examines various methods currently used for creating task scheduling. In Section 3, we discuss the system model and problem formulation for TS in detail, and in Section 4, we detail the AGWO task scheduling results that we believe to be the most effective. Section 5 presents a discussion of the AGWO methodology and their limitations. Finally, the work is summed up in Section 6.

## Literature Survey

Task scheduling in the cloud is addressed by the hybrid method proposed by Zhang et al. (2018), which combines the Biogeography-Based Optimisation (BBO) algorithm and the grey wolf optimisation (GWO). This method was designed to maximize resource utilization and minimize makespan time. The suggested HBBOG algorithm has been shown to perform better than conventional BBO and GWO algorithms in experiments regarding solution quality and convergence speed.

An MGWO-based task scheduling technique is presented for fog computing environments by Saif et al. (2019). The algorithm considers limitations on resources and task dependencies to maximise resource utilization, reducing energy consumption and optimising makespan time. The suggested MGWO algorithm is shown to be useful in delivering efficient task scheduling in fog computing settings by experimental results.

An enhanced GWO method is given for task scheduling in cloud-fog computing systems (Bacanin et al., 2019), as suggested by Bacanin, Nebojsa, et al. To improve the effectiveness of the solutions and convergence speed, the algorithm uses an adaptive search mechanism and a dynamic pheromone updating strategy. The enhanced GWO algorithm outperforms standard optimization techniques in terms of makespan, load balancing, and resource utilization, as determined by experimental evaluations.

For scheduling tasks in cloud-fog computing, (Kaur and Aron, 2021) present a hybrid algorithm that incorporates ACO and GWO. Considering limited resources, the program seeks to optimize makespan and energy consumption. The hybrid Tabu-GWO-ACO algorithm improves the basic techniques in terms of solution quality and convergence speed, as shown by experimental results.

Dynamic task scheduling in cloud-fog computing environments utilising the MGWO method is the subject of (Najafizadeh et al., 2022). To schedule effectively, the algorithm takes into account dynamic changes in workload and the availability of resources. The MGWO algorithm performs in terms of make time and resource utilization, and experimental results suggest that it operates well in dynamic circumstances.

Yin et al. (2022) proposed a method for scheduling tasks in cloud-fog computing environments based on combining the monarch butterfly optimization algorithm and the improved ant colony optimization algorithm (HMA). To identify near-optimal solutions, the authors framed the scheduling problem as an optimization task and used HMA to solve it. Extensive experiments were run to assess the effectiveness of the suggested method to minimize service delay and task energy consumption. The outcomes proved that HMA performed better than competing algorithms when it came to scheduling tasks.

By combining Grey Wolf Optimisation (GWO) and the Modified Moth Flame algorithm (MMFA), the unique hybrid algorithm proposed by (Gupta and Singh, 2022) enhances Deep reinforcement learning (DRL) by furnishing a local search mechanism for scheduling tasks in fog computing scenarios. The authors offer an alternative fitness function that factors in task dependencies, resource availability, and communication costs. Throughput, latency, makespan time, and energy usage were all assessed experimentally to determine the algorithm's efficacy. Results demonstrated the hybrid GMFA algorithm's effectiveness in scheduling jobs.

(Abualigah et al., 2020) introduced an enhanced version of TS-GWO for task scheduling in cloud and fog computing. The authors incorporated a dynamic parameter adaptation mechanism to improve the algorithm's convergence speed and search efficiency. Extensive simulations were performed to evaluate the proposed approach's performance in terms of makespan reduction, load balancing, and resource utilization. The results demonstrated the superiority of the enhanced TS-GWO algorithm over traditional TS-GWO and other existing algorithms.

For cloud-fog computing, (Subramoney et al., 2022) suggested an MS-PSO based task scheduling method. To effectively assign tasks to available resources, the authors framed the scheduling problem as an optimisation task and used the MS-PSO algorithm. The effectiveness of the algorithm in terms of makespan, resource utilisation, and energy usage was measured by experimental assessments. The outcomes confirmed the MS-PSO algorithm's superiority over the conventional methods in attaining effective task scheduling.

For the purpose of load balancing in cloud-fog computing environments, (Kakkottakath et al., 2022) created a multi-objective hybrid particle search optimisation algorithm (MOHPSO). The authors used a modified PSO algorithm to balance competing demands on available resources and schedule tasks accordingly. The algorithm's effectiveness in load balancing, makespan, and resource utilisation was determined by running a series of simulations. The findings proved the efficiency of the hybrid MOHPSO algorithm in satisfying load-balancing requirements.

An updated version of the improved differential evolution (IDE) algorithm was suggested by (Li et al., 2020) to schedule tasks in fog computing environments energy-efficiently. The algorithm's convergence rate and power consumption were optimised by implementing a dynamic parameter adaption method. Extensive studies were conducted to measure the algorithm's effectiveness in energy savings, maximising available resources, and making the most of available time. The findings proved that the improved IDE method delivered energy-efficient task scheduling.

Task scheduling in cloud-fog computing environments: a comparison of the GA-PSO method and the genetic algorithm (GA) by (Subramoney and Nyirenda, 2020). The algorithms' makespan, resource utilisation, and load balancing were analysed, and a variety of fitness functions were presented. The results demonstrated that the GA-PSO method was superior than GA in optimising makespan reduction and load balancing targets.

In order to better schedule tasks in the cloud, (Ahmed et al., 2019) suggested a Moth Search Algorithm (MSA) based on a Differential Evolution (DE) technique. The authors used DE to assign jobs to available resources effectively by casting the scheduling problem as an optimisation challenge. They compared DE's efficiency, effectiveness, and energy usage to that of several different metaheuristic algorithms. The findings proved the efficiency of DE in cloud computing systems for scheduling tasks.

An adaptive version of Differential Evolution was described by (Jing et al., 2021) for use in load balancing in fog computing settings. In order to tailor DE's mutation and crossover techniques to the peculiarities of a fog computing system, the scientists suggested a methodology called dynamic Moth-Flame Optimisation Differential Evolution (DMFO-DE). The effectiveness of the algorithm in terms of load distribution, turnaround time, and resource consumption was measured by extensive simulations. The findings show that adaptive Differential Evolution successfully accomplished the load balancing goals in fog computing.

Table 1 shows that previous researchers classified cloud computing paradigms based on parameters including makespan, energy usage, and SLA-based trust characteristics. We present an efficient and grey wolf optimization algorithm (AGWO), which uses ACO and GWO techniques to plan tasks in a fog environment while taking into account make time and total cost. The proposed method is especially useful for scheduling latency-sensitive applications, such as automotive networks, and for real-time applications, such as smart cities.

**Table 1. Comparison study of existing scheduling parameters**

| Authors | Technique Used | Parameters Addressed |
|---|---|---|
| Zhang et al., 2018 | HBBOG | Resource utilization and makespan time |
| Saif et al., 2023 | MGWO | Resource utilization, energy consumption and makespan time |
| Bacanin et al., 2019 | GWO | Makespan, load balancing, and resource utilization |
| Kaur and Aron, 2021 | Tabu-GWO-ACO | Makespan and energy consumption |
| Wang et al., 2022 | MGWO | Maketime and resource utilisation |
| Yin et al., 2022 | HMA | Energy consumption |
| Gupta and Singh, 2022 | MMFA | Throughput, latency, maketime and energy consumption |
| Abualigah et al., 2020 | TS-GWO | Makespan, load balancing, and resource utilization |
| Jing et al., 2021 | QoS-DPSO | Time, reliability, cost |
| Chen et al., 2022 | DRL | Response time, success rate, cost |
| Elaziz et al., 2021 | AOAM | Makespan, energy consumption |
| Hussain et al., 2022 | DVFS | Energy consumption, electricity price |
| Medishetti and Karri, 2023 | IDOA | Makespan time, VM failure rate, degree of imbalance |
| Medara et al., 2021 | EASVMC | energy consumption, resource utilization, VM migration |
| Mohammadzadeh et al., 2021 | HGALO-GOA | Cost, makespan, energy consumption |
| Kumar et al., 2023 | EEOA | Cost, energy consumption, makespan |
| Proposed algorithm | AGWO | Makespan and cost |

## System Model and Problem Formulation

### System Model

In order to better understand the proposed system, this article will outline the system model and the interaction between the various components involved in task scheduling (TS). The TS problem is also formulated for the audience to understand.

Based on our findings, we suggest a three-tiered architecture that includes cloud nodes, fog nodes, and Internet of Things (IoT) smart devices. In Figure 1 we see a simplified diagram of the entire proposal. This initial tier comprises smart sensors, wearable tech, and medical equipment. Nodes in the cloud or fog system can process service requests from these gadgets.

Personal computers, mini-servers, and smart gateways are all examples of nodes found in the middle layer, referred to as the fog nodes layer. With limited resources, each fog node serves as a smart server. Our architecture concludes in a cloud node layer comprising powerful servers with scalable computing resources capable of simultaneously processing many separate tasks. If you need to minimise delays and boost processing efficiency, scheduling time-sensitive tasks on a nearby fog node is the way to go. However, due to the superior computational power of cloud nodes, these types of intensive computations should be routed there rather than to fog nodes.

The suggested architecture depends significantly on the Fog Broker, which will live in the fog nodes layer. The Fog Broker incorporates the Resource Monitoring Service, the Task Scheduler, and the Task Manager. Requests for tasks come into the Task Manager from users and devices all over the IoT. It stores information about the tasks and resources they need, then sends that information to the Task Scheduler. The status of all available resources is tracked and recorded by the Resource Monitoring Service. The Task Scheduler benefits from its cooperation with both fog nodes and cloud nodes, which it uses to make more informed scheduling decisions.

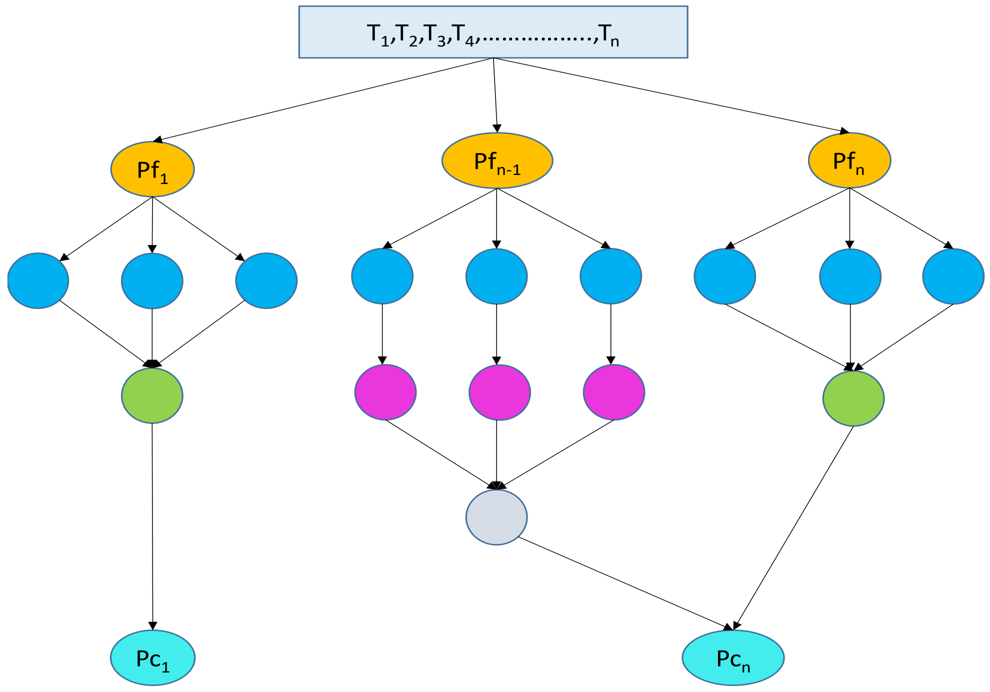

**Figure 2. Random workflow for cloud-fog scheduling**

The created methods for scheduling tasks are executed by the Task Scheduler, which forms the backbone of the Fog Broker. The Task Scheduler distributes work orders to computers based on their specific requirements, in addition to the capabilities of the available resources and the characteristics of the submitted tasks. Therefore, task requests are processed quickly and communicated back to the Fog Broker for distribution.

## Problem Formulation

Task scheduling in a cloud-fog setting is formulated mathematically and described in the next section. Let's pretend the suggested cloud-fog computing infrastructure includes a set of n separate tasks $T = T1, T2, T3, \ldots Tn$ that have been submitted to the Fog Broker for execution. The time limit, file size of input and output, memory requirements, and other parameters are all unique to each Tk work. Millions of Instructions (MI) are used to quantify how long a process takes to complete. There are a total of m nodes in the cloud-fog system, which includes both cloud and fog nodes. This set can be written as $CN = CN1, CN2, CN3, \ldots, CNm$ where CNj is the jth computer in the network. CPU speed, RAM size, network throughput, and storage space are only some of the characteristics unique to each $CN_j$. MIPS (Millions of Instructions Per Second) is the standard measure of a CN's processing speed.

$$\left(CN = N_{cloud} U\ N_{Fog}\right) \ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

Figure 2 depicts the scheduling system as a direct acyclic graph (DAG), with T representing the set of n tasks *(t1, t2,..., tn)* and E representing the set of directed edges (dependence or priority limitations) between jobs in the workflow. G = (P, EG) is a complete graph that can be used as an alternate representation for the nodes in a cloud-fog network.

Figure 2 depicts a DAG with eight tasks *(T1, T2,..., Tn)* and their respective processors (Pf1, Pf2,..., Pfn) in fog computing and cloud computing, respectively. The cloud nodes and the results from the fog nodes are integrated after the input tasks have been handled. The main objective is to allocate tasks to processors in such a way as to optimise system performance while minimising resource consumption and other related costs. In this case, our task scheduler thinks about these things while deciding how to divide up processor time.

Since there are n task requests and m computing nodes, the *ECT* matrix has nm dimensions to indicate the expected computation time for each task request for every computing node. The Task Scheduler uses the *ECT* matrix to schedule tasks. The following formula can be used to determine the estimated execution time, denoted by the symbol $ECT_{k,j}$, for task Tk on compute node $CN_j$.

The expected execution time (ECT) of task Tk on computing node CNj can be expressed as:

$$ECT_{k,j} = Task\_Length\_k \ / \ Computing\_Power\_j \ \ldots. (2)$$

Here, Computing Power_j denotes the Millions of Instructions Per Second (MIPS) level processing power of computing node $CN_j$, and Task length_k is the MIPS-level length of the task $T_k$.

It is known that the problem of allocating jobs to available compute nodes is NP-complete. The primary objective is to find a timetable that reduces the makespan (the time it takes to complete the task). By keeping the makespan as small as possible, we can ensure that no single operation takes too long to complete. In this research, we focus on reducing the makespan of a cloud-fog system by solving the task scheduling (TS) problem. The following formula can be used to determine the makespan (MK) for a given schedule Z:

A schedule Z's makespan (MK) is determined by the maximum total expected execution time (ECT) across all computing nodes $CN_j$, where j ranges from 1 to m, and considering all task requests Tk, where k ranges from 1 to n. The equation to calculate the makespan is:

$$MK(Z) = max\ \{\ \Sigma\ ECT_{k,j}\ |\ j \in \{1,\ 2,\ ...,\ m\}\ \} .. (3)$$

At this stage, the task scheduling (TS) problem can be mathematically formulated as follows: To minimize the makespan of a schedule Z, represented by the function f(Z), which can be defined as:

$$f(Z) = min\ \{\ max\ \{\ \Sigma\ ECT_{k,j} |\ j \in \{1,2,...,m\}\ \}\ |\ k \in \{1,2,...,n\}\ \} \ldots\ldots\ldots\ldots. (4)$$

In this formulation, f(Z) seeks to minimize the maximum total expected execution time (ECT) across all computing nodes *CNj*, considering all task requests $T_k$.

## Execution time

Minimize the overall execution time (makespan) of task scheduling in a cloud and fog computing system.

Let's consider the following variables and parameters:

- Task set: $T = \{T1,\ T2,\ T3,\ ...,\ Tn\}$ where each task Tk represents an individual task with attributes such as task length, memory requirements, input/output file sizes, and deadline.
- Computing nodes: $(CN = CN1,\ CN2,\ CN3,...,\ CNm)$ where $CN_j$ is a computational node with its own CPU speed, memory size, network bandwidth, and storage capacity.

The execution time of task $T_k$ on computing node $CN_j$, denoted as $ECT_{k,j}$, can be calculated using a mathematical equation that considers the characteristics of the task and the computing node. This equation can be customized based on the specific system and requirements but generally involves factors such as task length, computing power, communication latency, and resource constraints. An example equation for calculating $ECT_{k,j}$ is:

$$ECT_{k,\ j} = f\left(TL\_k,\ CP\_j,\ CL\_k,j,\ RC\_j\right)\ \ldots\ldots(5)$$

TL represents the task length, CP represents the computation power, CL represents the communication latency, and RC represents the Resource constraints. Here, f() represents the mathematical function that considers the relevant parameters. The equation can be further refined and customized based on the specific considerations and optimization goals of the task scheduling problem in cloud and fog computing. The objective is to find an optimal scheduling solution that minimizes the overall execution time by assigning tasks to suitable computing nodes while considering factors such as task dependencies, resource availability, and communication overhead.

## Response time

minimize the overall response time of task scheduling in a cloud and fog computing system.

Let's consider the following variables and parameters:

- Task set: $T = \{T1,\ T2,\ T3,\ ...,\ Tn\}$ where each task Tk represents an individual task with attributes such as task length, memory requirements, input/output file sizes, and deadline.
- Computing nodes: $(CN = CN1,\ CN2,\ CN3,...,\ CNm)$ where $CN_j$ is a collection of computing nodes, each with its own parameters including CPU speed, memory size, network throughput, and available disc space.

The response time of task Tk on computing node $CN_j$, denoted as $RT_{k,j,}$ can be calculated using a mathematical equation that considers various factors such as task length, communication latency, resource availability, and scheduling overhead. This equation can be customized based on the specific system and requirements. An example equation for calculating $RT_{k,j,}$ is:

$$RT_{k,\ j} = g\left(TL\_k,\ CL\_k,j,\ RA\_j\right)\ \ldots\ldots..(6)$$

TL represents the task length, CL represents the communication latency, and RA represents the Resource availability. The g() notation here denotes the particular mathematical function that makes use of all of the appropriate variables. The equation can be modified and improved upon in light of the unique constraints and optimisation targets of the cloud and fog computing work scheduling problem.

## Makespan

The makespan is the overall amount of time needed to finish all of the tasks in a particular schedule when using cloud computing for task scheduling. When comparing different task scheduling algorithms for use in the cloud, the makespan is a crucial indicator of how well each one performs.

The makespan, denoted as MK, can be calculated using the following equation:

$$MK = max\ \{\ CompletionTime(Tk)\ \ |\ Tk \in T\ \} .. (7)$$

Completion Time(Tk) is the time it took to complete the task Tk. The schedule's total time to completion is represented by the maximum completion time selected from the list of tasks using the equation.

## Objective function

The objective function in task scheduling for cloud computing represents the optimization goal that a scheduling algorithm aims to achieve. It quantifies a metric that needs to be minimized or maximized to find an optimal task schedule.

Minimising the makespan, or the total amount of time needed to finish all the activities in a given schedule, is a common objective function in cloud computing task scheduling. Other goals, such as energy usage, resource utilisation, or cost, may also be taken into account, however, depending on the specific needs and aims.

Let's denote the objective function as f(Z), where Z represents a schedule of task assignments to computing nodes. The equation for the objective function in task scheduling can be formulated as follows:

$$f(Z) = g(Z) \qquad \text{.............}(8)$$

Here, g() is the mathematical function that quantifies the objective. The specific form of the objective function equation depends on the optimization goal and the constraints considered in the task scheduling problem. For example, if the objective is to minimize the makespan, the objective function equation can be:

$$f(Z) = min\left\{ MK(Z) \right\} \qquad \text{.............}(9)$$

Where MK(Z) represents the makespan of the schedule Z, which can be calculated using appropriate equations as discussed earlier. Other objective functions can be formulated accordingly, incorporating different metrics or constraints relevant to the specific task scheduling problem in cloud computing.

## Initialization

The initialization step in the proposed Ant Grey Wolf Optimization (AGWO) algorithm for task scheduling in cloud computing is a process that sets up the initial population of grey wolves and ants in the search space. While this step is primarily implemented using randomization, it is not explicitly defined by a mathematical equation. The initialization procedure can be described as follows:

1. Initialize Grey Wolves:
   - o Set the population size of grey wolves as NP.
   - o Generate NP random positions in the search space for the grey wolves.
   - o Each position represents a potential solution for task scheduling.
2. Initialize Ants:
   - o Determine the number of ants as NA.
   - o Generate NA random positions in the search space for the ants.
   - o Each position represents a candidate solution for a task assignment.

The initialization step aims to create an initial diverse set of solutions by randomly assigning positions to the grey wolves and ants in the search space. These positions represent potential task assignment configurations on the available computing nodes. It's important to note that the specific details of the initialization process, such as the population size and position assignment, can be tailored based on the problem requirements and characteristics. The main objective is to ensure a varied set of initial solutions that can be further optimized during the subsequent steps of the AGWO algorithm.

Following the initialization, the AGWO algorithm progresses to the Ant Colony Optimization (ACO) and Grey Wolf Optimization (GWO) phases, along with other operations, to refine the task scheduling solutions and achieve optimal or near-optimal results.

## Updating stage

In the proposed Ant Grey Wolf Optimization (AGWO) algorithm for task scheduling in cloud computing, the updating stage refers to the process of updating the positions of grey wolves and ants based on their search behavior and the quality of solutions. While this stage involves iterative updates, it is not represented by a single mathematical equation. However, I can provide an overview of the updating process in AGWO:

1. Ant Colony Optimization (ACO) Phase:
   - o o Based on trial pheromones and heuristic information, each ant builds a proposed solution by assigning tasks to compute nodes probabilistically.
   - o The effectiveness of the ant-built solutions informs the maintenance of the pheromone trails.
2. Grey Wolf Optimization (GWO) Phase:
   - o The grey wolves use their hunting behavior to explore the search space and refine the solutions.
   - o The positions of grey wolves are updated based on the fitness of the solutions and the social hierarchy among the grey wolves (alpha, beta, delta).
3. Local Search:
   - o A local search operation may be performed to further refine the solutions obtained from the ACO and GWO phases.
   - o This can involve neighborhood exploration or local optimization techniques.

The updating stage involves iterative updates of the positions of grey wolves and ants based on the evaluation of their solutions and searches behavior. The specific equations and update rules used in AGWO can vary based on the task scheduling problem and the design choices of the algorithm.

It's important to note that the AGWO algorithm combines the principles of ACO and GWO, and the updating stage integrates their respective mechanisms to refine the task scheduling solutions. The algorithm iteratively improves the solutions through multiple iterations until convergence or a termination condition is met, aiming to find optimal or near-optimal task assignment schedules in cloud computing environments.

## Proposed AGWO Algorithm

Pseudo code for AGWO Task Scheduling (**Input:** tmax, m, n, N):

```
1: Set the initial value for the population X.
2: t = 1.
3: while t <= tmax do
4:     Compute the fitness value (Fi) for each solution Xi.
5:     Determine the best solution Xb.
6:     Update X1 using Eq. (2).
7:     for i = 2 to N do
8:         rand = random value between 0 and 1.
9:         if rand < 1/3 then
10:            Enhance Xi using Eq. (4).
11:        else if 1/3 < rand < 2/3 then
12:            Enhance Xi using Eq. (6).
13:        else
14:            Enhance Xi using Eq. (7).
15:        Compute the fitness of each solution Xi.
16:        Update Xb.
17:        Compute the probability Pri using Eq. (8).
18:        Improve Xi using Eq. (9).
19:    t = t + 1.
20: End while
21: Return Xb.
```

Based on the probabilities (Pri) of each solution, either Ant Colony Optimisation (ACO) or Grey Wolf Optimisation (GWO) operators are used to update the feasible regions. The probabilities (Pri) are calculated as follows:

$$Pri = Fiti \;/\; \Sigma\big(Fiti\big)\; for\ i = 1\ to\ N \ldots\ldots\ldots\ (10)$$

where Fiti represents the fitness value of solution i, and N is the total number of solutions."

## Results

To implement the proposed algorithm, we utilized the CloudSim (Abualigah, Laith, et al. 2020) toolkit for modelling the cloud environment. Extensive experiments were conducted using a simulation strategy. In order to validate the effectiveness of the AGWO algorithm, we provide evaluation comparisons with the widely-used MOTSAO, CSDEO, QoS-DPSO and BLEMO algorithms.

We conducted simulations using a machine with an Intel Core i5-3373U CPU running at 1.8 GHz and 6 GB of RAM to get AGWO's results. CloudSim Toolkit was used to implement the AGWO algorithm, and its performance was measured in terms of both cost and makespan. In the study, we only investigated tasks that were not dependent on results from other tasks. It was believed that the links' data-transfer speeds would be distributed normally, within the range of 40 Mbps to 10,000 Mbps. Parameters used in the simulation analysis are outlined in Table 2. These parameters include the pheromone value, the local evaporation rate (p), and the global evaporation rate (τ). The table shows how many ants, tasks, virtual machines, and data centres were used to run the simulation.

**Table 2. Scheduling based on ACO parameters**

| α | β | p | τ | ζ | Imax | ants | tasks | VM |
|---|---|---|---|---|---|---|---|---|
| **2** | 10 | 0.8 | 1.0 | 0.2 | 4 | 15 | 35-150 | 8-24 |

**Table 3. Cloudsim toolkit parameters**

| Parameter | Cloud | Fog |
|---|---|---|
| Number of VMs | [10,15,20] | [15,20,35] |
| Computing power (MIPS) | [3000:5000] | [1000:2000] |
| RAM (MB) | [5000:20000] | [250:5000] |
| Bandwidth (Mbps) | [512:4096] | [128:1024] |
| Cost (G$) | [0.6:1.0] | [0.2:0.5] |

## Makespan

The proposed method consistently obtains the best makespan of any known heuristic. Compared to other comparable algorithms such as MOTSWAO, CSDEO, QoS-DPSO, BLEMO, the proposed AGWO consistently generated better makespan results across all scenarios, as represented by the graph displaying in the figure 3 best makespan results for real-time workloads with varying task sizes. The conclusion is that AGWO is an excellent method for addressing important challenges in cloud fog task allocation scheduling.

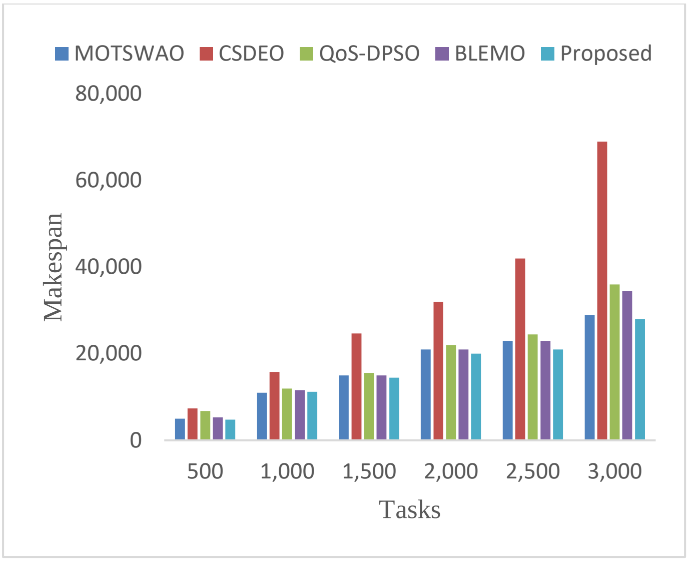

**Figure 3. Makespan calculation of proposed method**

### Throughput

Using realistic workloads, the below figure 4 compares the throughput attained by the MOTSWAO, CSDEO, QoS-DPSO, BLEMO and the proposed AGWO algorithm. The throughput measurement is the number of IoT tasks on the horizontal axis of these diagrams and on the vertical axis. These results demonstrate that AGWO outperforms all other evolutionary algorithm techniques tested in terms of throughput across all task dimensions and workloads. It demonstrates that the AGWO algorithm can discover near-optimal solutions to real-world problems with reliability and consistency. In many test cases and workloads, traditional MOTSWAO and CSDEO techniques produce a higher throughput than QoS-DPSO and BLEMO. In terms of throughput, the results indicate that the MOTSWAO method outperforms the CSDEO method across all task dimensions and demands. h- DEWOA converges faster compared to BLEMO and other metaheuristic algorithms. This advantage makes the hybridization of the proposed AGWO superior to other methods in terms of throughput.

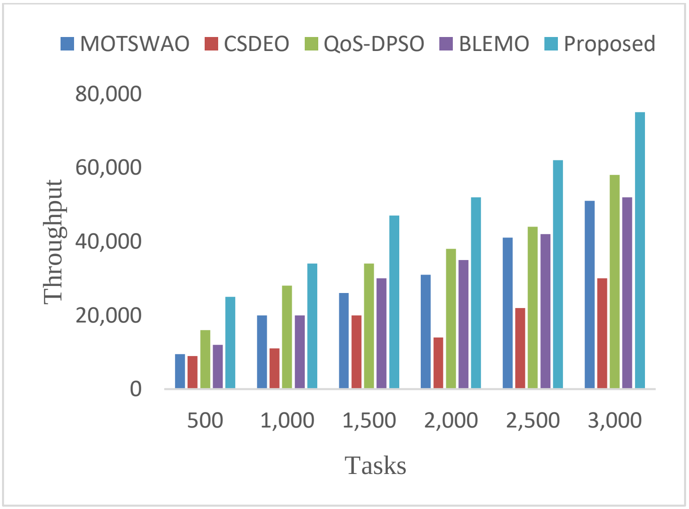

**Figure 4. Throughput calculation of proposed method**

### Execution time

It demonstrates that the proposed method (AGWO) reduces overall execution time, particularly for numerous activities as shown in figure 5. The AGWO scheduler uses a hybrid of the ACO and GWO algorithms to assign resources; as a result, less time is wasted searching and all jobs receive near-perfect VM. AGWO is approximately 28.3 percent more efficient than GWO. When there are few requests, MOTSWAO and CSDEO can complete the task rapidly. However, as the number of tasks increases, the execution time of these algorithms increases substantially. Compared to MOTSWAO and CSDEO, QoS-DPSO, BLEMO processing delays for improvement are 8.6 and 9.12 percent, respectively.

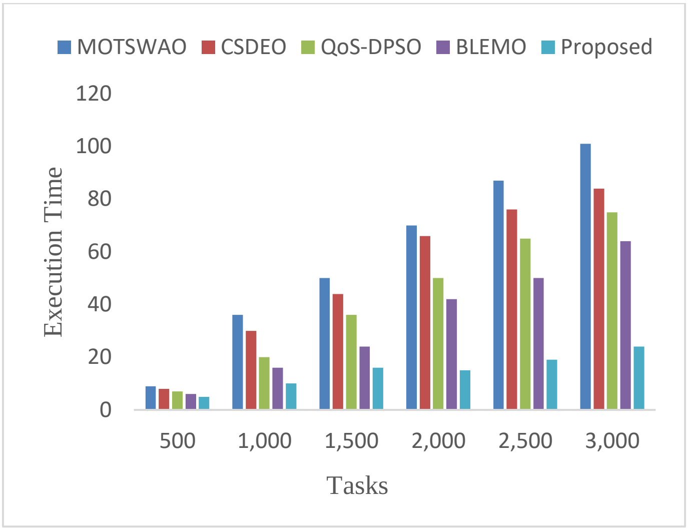

**Figure 5. Execution time of proposed method**

### Response time

Below figure 6 depict the typical response times for Internet of Things-related tasks. Response time is the time required for an IoT device to submit a request and then receive a response. Compared to other methodologies, the MOTSWAO's Internet of Things (IoT) response time is the quickest (58.12 seconds). MOTSWAO and CSDEO outperform QoS-DPSO by 35.32 and 19.56 seconds, respectively. It depicts the typical response periods for the real time workloads, broken down by specific tasks. MOTSWAO requires a lot of time, and the recommended AGWO method requires the shortest turnaround time among the real-time tasks. Due to their slow response periods, the MOTSWAO and CSDEO methods break the SLA in a significant way. Taking into account actual demands, the previous data indicate that throughput time and makespan have improved significantly. The success of AGWO can largely be attributed to the incorporation of CSDEO and QoS-DPSO into the proposed algorithm. We conclude that the AGWO we created is the most effective method to address cloud-fog Task Scheduling optimization concerns because it consistently generates the best solutions.

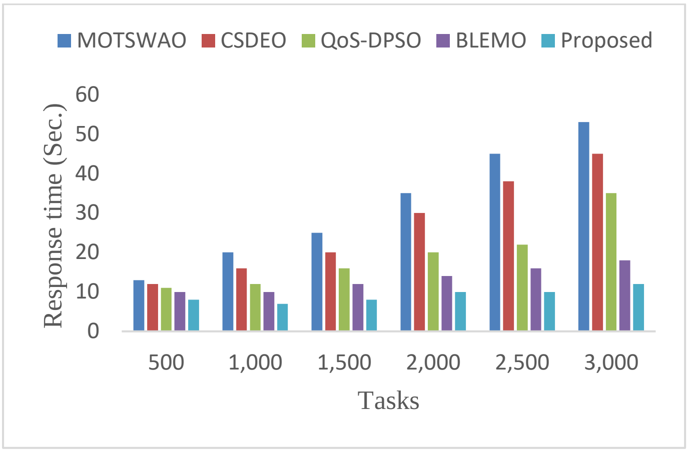

**Figure 6. Response time of proposed method**

## Cost

It has been observed that among the algorithms, MOTSWAO has the most expense and AGWO has the lowest. As seen in Figure 7, both CSDEO and BLEMO are reasonably priced when it comes to using cloud and fog resources. Remember that MOTSWAO will charge you more because it prioritises reducing the time it takes to complete a task over other factors. In contrast, the average cost of CSDEO is reduced by 3.5% while the average cost of MOTSWAO is reduced by 25.38&percnt; with QoS-DPSO. This is mostly because the proposed method reduces costs by distributing tasks to the most cost-effective resources, regardless of factors like additional charges and energy use.

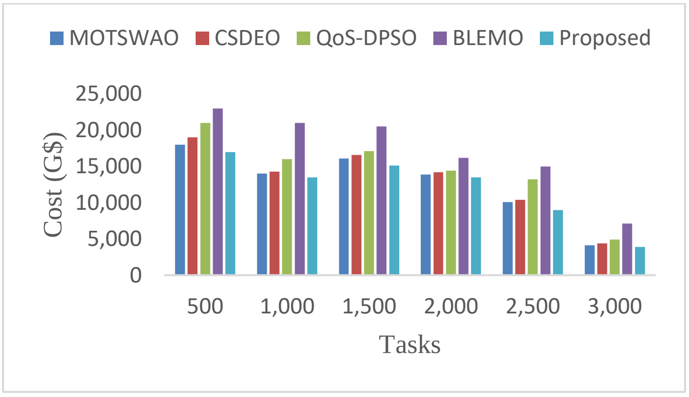

**Figure 7. Total Cost calculation of proposed method**

## Discussion

When evaluating AGWO's (Ant Grey Wolf Optimization) performance for cost-aware task scheduling in a cloud fog environment, two important metrics to consider are makespan and cost. The discussion on the results of AGWO concerning these metrics:

1. Makespan: Makespan refers to the total time taken to complete all tasks in the scheduling process. AGWO aims to minimize the makespan by optimizing task allocation and resource utilization. By leveraging the hybrid metaheuristic approach, AGWO explores different scheduling strategies and adjusts the positions of the ant grey wolves in the search space to find better solutions.

The effectiveness of AGWO in minimizing the makespan would depend on the problem instance, the characteristics of the tasks and resources, and the algorithm parameters. In general, AGWO's exploration capability, combining global and local search strategies, can help in finding scheduling solutions that reduce the makespan compared to traditional methods. However, the degree of improvement would vary depending on the problem's complexity and the algorithm's efficiency in exploring the search space.

2. Cost: AGWO incorporates cost factors into the task scheduling process to optimize the overall cost. This includes considerations such as energy consumption, resource utilization, and communication overhead. The cost-awareness of AGWO allows it to find scheduling solutions that minimize the cost while meeting the performance requirements.

The impact of AGWO on cost reduction would depend on the specific cost factors considered, the problem instance, and the weights assigned to these factors in the fitness evaluation. By integrating cost metrics into the fitness function, AGWO can guide the search process towards cost savings solutions. The algorithm's ability to balance cost and performance can help in achieving cost reduction compared to traditional scheduling methods that do not explicitly consider cost factors.

It's important to note that the trade-off between makespan and cost exists in AGWO. Optimizing one metric may lead to compromises in the other. The algorithm's parameter settings and the problem requirements can influence the extent of this trade-off. To assess the performance of AGWO in terms of makespan and cost, it is essential to compare it with other existing task scheduling algorithms and approaches. Benchmarking experiments can be conducted on various problem instances to evaluate AGWO's effectiveness and competitiveness in achieving lower makespan and cost compared to alternative methods.

## Limitations

While AGWO (Ant Grey Wolf Optimization) for cost-aware task scheduling in a cloud fog environment using a hybrid metaheuristic algorithm has its merits, there are also some limitations to consider. Here are a few limitations of AGWO:

1. Parameter Sensitivity: AGWO requires parameter tuning to achieve optimal performance like many metaheuristic algorithms. The performance of AGWO

can be sensitive to the choice of parameters such as population size, convergence criteria, and search operators. Finding the right parameter settings can be a challenging and time-consuming task, requiring expertise and extensive experimentation.

2. Scalability: The scalability of AGWO can be a limitation when dealing with large-scale cloud fog environments with a high number of tasks and resources. As the problem size increases, the complexity of the search space grows, which can impact the algorithm's performance. AGWO's population-based approach and distributed computation capability may help mitigate scalability issues to some extent, but there can still be limitations in handling extremely large problem instances.

3. Convergence to Suboptimal Solutions: AGWO, like other metaheuristic algorithms, is not guaranteed to find the global optimal solution. Depending on the problem instance and algorithm parameters, AGWO may converge to suboptimal solutions that do not provide the best possible cost-aware task scheduling. The exploration and exploitation balance of the algorithm can impact its ability to escape local optima and find better solutions.

4. Limited Problem-Specific Adaptability: AGWO, being a general-purpose metaheuristic algorithm, may lack problem-specific adaptability. It may not exploit specific characteristics of the cost-aware task scheduling problem in a cloud fog environment. As a result, AGWO might not fully utilize domain-specific knowledge and constraints, potentially leading to suboptimal solutions.

5. Lack of Comparative Performance Evaluation: While AGWO shows promise in cost-aware task scheduling, its performance should be compared against other existing algorithms and approaches. Without proper benchmarking and comparison, it is challenging to determine the true effectiveness and competitiveness of AGWO against state-of-the-art methods. Comparative evaluation of different problem instances and performance metrics is necessary to comprehensively understand AGWO's strengths and weaknesses.

## Conclusions and Future Work

In a cloud-fog computing environment, it has been demonstrated that a cost-aware task scheduling system may effectively manage the allocation and execution of IoT tasks. By combining the ideas of ant colony optimisation (ACO) and grey wolf optimisation (GWO), we developed a novel adaptive method to address the task scheduling problem. The goal of this AGWO strategy was to boost the efficiency of the ACO algorithm. The suggested algorithm was evaluated using a number of tests, with results compared to those of other existing metaheuristics and parameters The makespan time was reduced by 42% and we reduced the cost by 36% while resolving the task scheduling problem. Across a variety of scientific workflows and performance metrics, the given scheduling solution was found to be superior to competing approaches. It outperforms competing approaches while producing superior results in a less amount of time. To further improve the stability and reliability of the suggested technique in the cloud-fog environment, future studies will take into account other factors such as workload redistribution and VM failure. By taking them into account, the proposed method can achieve even higher performance levels and more reliability in real-world situations.

## Acknowledgement

I would like to express my sincere gratitude to my Ph.D. guide Dr. K Ganesh Reddy, for his valuable suggestions during the development of this research work.

## Conflicts of interest

The authors declare no conflict of interest.

## References

Abualigah, L., & Diabat, A. (2020). A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments, *Cluster Comput.*, *24*(1), 205-223. http://dx.doi.org/10.1007/s10586-020-03075-5

Abualigah, L., Elaziz, M.A., Hussien, A.G., Alsalibi, B., Jalali, S.M.J., Gandomi, A.H. (2020). Ts-gwo: Iot tasks scheduling in cloud computing using grey wolf optimizer. Swarm Intelligence for Cloud Computing. Chapman and Hall/CRC, 127-152. https://link.springer.com/article/10.1007/s10489-020-01947-2

Abualigah, L., Shehab, M., Alshinwan, M., Alabool, H., Abuaddous, H.Y., Khasawneh, A.M., & Al Diabat,M. (2020). TS-GWO: IoT tasks scheduling in cloud computing using Grey Wolf optimizer, in: Swarm Intelligence for Cloud Computing, *Chapman and Hall/CRC*, pp. 127–152. https://doi.org/10.1201/9780429020582-5

Ahmed, O.H., Lu, J., Xu, Q., Ahmed, A.M., Rahmani, A. M., & Hosseinzadeh, M. (2021).

Using differential evolution and Moth–Flame optimization for scientific workflow scheduling in fog computing. *Applied Soft Computing*, *112*, 107744. http://dx.doi.org/10.1016/j.asoc.2021.107744

Bacanin, N., Bezdan, T., Tuba, E., Strumberger, I., Tuba, M., Zivkovic, M. (2019). Task scheduling in cloud computing environment by grey wolf optimizer. *IEEE, 2019, 27th telecommunications forum* (TELFOR), Belgrade, Serbia, pp. 1-4. http://dx.doi.org/10.1109/TELFOR48224.2019.8971223

Chen, Z.G., Zhan, Z.H., Lin, Y., Gong, Y.J., Gu, T.L., Zhao, F., Yuan, H.Q., Chen, X., Li, Q., & Zhang, J. (2018). Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach, *IEEE Trans. Cybern.*, 49(8), 2912–2926. http://dx.doi.org/10.1109/TCYB.2018.2832640

Cheng, F., Huang, Y., Tanpure, B., Sawalani, P., Cheng, L., & Liu, C. (2022). Cost-aware job scheduling for cloud instances using deep reinforcement learning. *Cluster Computing*, 25(1), 619-631. https://doi.org/10.1007/s10586-021-03436-8

Elaziz, M.A., Xiong, S., Jayasena, K.P.N., & Li, L. (2019). Abd Elaziz, & Mohamed, et al. (2019) Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution. *Knowledge-Based Systems*, *169*, 39-52. https://doi.org/10.1016/j.knosys.2019.01.023

Elaziz, M.A., Abualigah, L., Ibrahim, R. A., & Attiya, I. (2021). IoT workflow scheduling using intelligent arithmetic optimization algorithm in fog computing. *Computational Intelligence and Neuroscience*, *2021*, 1-4. https://doi.org/10.1155/2021/9114113

Fu, J.S., Liu, Y., Chao, H.C., Bhargava, B.K., & Zhang, Z.J. (2018). Secure data storage and searching for industrial IoT by integrating fog computing and cloud computing, *IEEE Trans. Ind. Inf.*, 14(10), 4519–4528. https://doi.org/10.1109/TII.2018.2793350

Ghasempour, A. (2019). Internet of things in smart grid: Architecture, applications, services, key technologies, and challenges, *Inventions*, *4*(1), 22. https://doi.org/10.3390/inventions4010022

Gupta, S., & Singh, N. (2022). Fog-GMFA-DRL: Enhanced deep reinforcement learning with hybrid grey wolf and modified moth flame optimization to enhance the load balancing in the fog-IoT environment. *Advances in Engineering Software*, *174*, 103295. https://doi.org/10.1016/j.advengsoft.2022.103295

Ghasempour, A., & Moon, T.K. (2016). Optimizing the number of collectors in machine-to-machine advanced metering infrastructure architecture for internet of things-based smart grid, *IEEE Green Technologies Conference, GreenTech*, pp. 51–55. http://dx.doi.org/10.1109/GreenTech.2016.17

Hussain, M., Wei, L. F., Rehman, A., Abbas, F., Hussain, A., & Ali, M. (2022). Deadline-constrained energy-aware workflow scheduling in geographically distributed cloud data centers. *Future Generation Computer Systems*, *132*, 211-222. http://dx.doi.org/10.1016/j.future.2022.02.018

Jing, W., Zhao, C., Miao, Q., Song, H., & Chen, G. (2021). QoS-DPSO: QoS-aware task scheduling for the cloud computing system. *Journal of Network and Systems Management*, *29*(1), 5. https://doi.org/10.1007/s10922-020-09573-6

Kaur, M., & Aron, R. (2021). Focalb: Fog computing architecture of load balancing for scientific workflow applications. *Journal of Grid Computing*, *19*(4), 40. https://doi.org/10.1007/s10723-021-09584-w

Kumar, M. S., & Karri, G.R. (2023). EEOA: Cost and Energy Efficient Task Scheduling in a Cloud-Fog Framework. *Sensors*, *23*(5), 2445. http://dx.doi.org/10.3390/s23052445

Li, X., Zhang, G., Zheng, X., & Hua, S. (2020). Delay optimization based on improved differential evolutionary algorithm for task offloading in fog computing networks. *IEEE 2020, International Conference on Wireless Communications and Signal Processing* (WCSP).

http://dx.doi.org/10.1109/WCSP49889.2020.9299850

Lin, B., Guo, W., Xiong, N., G. Chen, G., Vasilakos, A.V., & Zhang, H. (2016). A pretreatment workflow scheduling approach for big data applications in multicloud environments, *IEEE Trans. Netw. Serv. Manag.*, *13*(3), 581–594. http://dx.doi.org/10.1109/TNSM.2016.2554143

Liu, X.F., Zhan, Z.H., Deng, J.D., Li, Y., Gu, T., & Zhang, J. (2016). An energy efficient ant colony system for virtual machine placement in cloud computing, *IEEE Trans. Evol. Comput.*, 22(1), 113–128. http://dx.doi.org/10.1109/TEVC.2016.2623803

Medishetti, S. K., & Karri, G. (2023). An Improved Dingo Optimization for Resource Aware Scheduling in Cloud Fog Computing Environment. *Majlesi Journal of Electrical Engineering*, *17*(3), http://dx.doi.org/10.30486/MJEE.2023.1989335.1165

Medara, R., Singh, R. S., & Amit. (2021). Energy-aware workflow task scheduling in clouds with virtual machine consolidation using discrete water wave optimization. *Simulation Modelling Practice and Theory*, *110*, 102323. http://dx.doi.org/10.1016/j.simpat.2021.102323

Mohammadzadeh, A., Masdari, M., & Gharehchopogh, F. S. (2021). Energy and cost-aware workflow scheduling in cloud computing data centers using a multi-objective optimization algorithm. *Journal of Network and Systems Management*, *29*(3), 1-34. https://link.springer.com/article/10.1007/s10922-021-09599-4

Najafizadeh, A., Salajegheh, A., Rahmani, A.M., & Sahafi, A. (2022). Multi-objective Task Scheduling in cloud-fog computing using goal programming approach. Cluster Computing, 25(1), 141-165. https://doi.org/10.1007/s10586-021-03371-8

Nguyen, B.M., Binh, H.T.T., Anh, T.T., & Son, D.B. (2019). Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud-fog computing environment, *Appl. Sci.*, *9*(9), 1730. https://doi.org/10.3390/app9091730

Saif, F.A., Latip, R., Hanapi, Z.M., Shafinah, K. (2023). Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing. *IEEE Access, 11*, 20635-20646. http://dx.doi.org/10.1109/ACCESS.2023.3241240

Subramoney, D., & Nyirenda, C.N. (2020). A comparative evaluation of population-based optimization algorithms for workflow scheduling in cloud-fog environments. *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, http://dx.doi.org/10.1109/SSCI47803.2020.9308549

Subramoney, D., & Nyirenda, C.N. (2022). Multi-Swarm PSO Algorithm for Static Workflow Scheduling in Cloud-Fog Environments. *IEEE Access, 10*, 117199-117214. https://doi.org/10.1109/ACCESS.2022.3220239

Thekkepurayil, J.K.V., Suseelan, D.P., & Keerikkattil, P.M. (2022). Multi-objective Scheduling Policy for Workflow Applications in Cloud Using Hybrid Particle Search and Rescue Algorithm. *Service Oriented Computing and Applications*, *16*(1), 45-65. https://doi.org/10.1007/s11761-021-00330-4

Vijayalakshmi, R., Vasudevan, V., Kadry, S., & Lakshmana Kumar, R. (2020). Optimization of makespan and resource utilization in the fog computing environment through task scheduling algorithm, *Intl. J. Wavelets Multiresolut. Inform. Process.*, *18*(01), 1941025. http://dx.doi.org/10.1142/S021969131941025X

Wang, S., Zhao, T., & Pang, S. (2020). Task scheduling algorithm based on improved firework algorithm in fog computing, *IEEE Access*, *8*, 32385–32394. http://dx.doi.org/10.1109/ACCESS.2020.2973758

Wang, Y., Guo, C., & Yu, J. (2018). Immune scheduling network-based method for task scheduling in decentralized fog computing, Wirel. Commun. *Mobile Comput.*, *33*(16), e4583. http://dx.doi.org/10.1002/dac.4583

Yang, M., Ma, H., Wei, S., Zeng, Y., Chen, Y., & Hu, Y. (2020). A multi-objective task scheduling method for fog computing in cyber-physical-social services, *IEEE Access*, *8*, 65085–65095. http://dx.doi.org/10.1109/ACCESS.2020.2983742

Yin, Z., Xu, F., Li, Y., Fan, C., Zhang, F., Han, G., Bi, Y. (2022). A Multi-Objective Task Scheduling Strategy for Intelligent Production Line Based on Cloud-Fog Computing. *Sensors*, *22*(4), 1555. https://doi.org/10.3390/s22041555

Zhang, X., Kang, Q., Cheng, J., & Wang, X. (2018). A novel hybrid algorithm based on biogeography-based optimization and grey wolf optimizer. *Applied Soft Computing 67*, 197-214. https://doi.org/10.1016/j.asoc.2018.02.049

Zuo, L., Shu, L., Dong, S., Zhu, C., & Hara, T. (2015). A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing, *IEEE Access*, *3*, 2687–2699. http://dx.doi.org/10.1109/ACCESS.2015.2508940.

**How to cite this Article:**

M. Santhosh Kumar & Ganesh Reddy Karri (2023). AGWO: Cost Aware Task Scheduling in Cloud Fog Environment Using Hybrid Metaheuristic Algorithm. *International Journal of Experimental Research and Review*. 33, 41-56.

**DOI:** https://doi.org/10.52756/ ijerr.2023.v33spl.005