*International Journal of Advanced Technology and Engineering Exploration, Vol 10(107)*
*ISSN (Print): 2394-5443 ISSN (Online): 2394-7454*
*http://dx.doi.org/10.19101/IJATEE.2023.10101467*

Research Article

# A meta-heuristic clustered grey wolf optimization algorithm for cloud resource scheduling

**Juliet A Murali[1*] and Brindha T[2]**
Research Scholar, Department of Computer Science, Noorul Islam Centre for Higher Education, Tamilnadu, India[1]
Associate Professor, Department of Information Technology, Noorul Islam Centre for Higher Education, Tamilnadu, India[2]

Received: 23-April-2023; Revised: 10-October-2023; Accepted: 12-October-2023


## Abstract
***Cloud computing services refer to the on-demand provision of computer resources and services over the internet. Numerous resources are available from cloud service providers (CSPs). Infrastructure as a service (IaaS) is a cloud computing service that enables the sharing of computer resources over the web. One of the key challenges in cloud scheduling is the efficient allocation of these resources. Recently, several swarm-intelligence (SI) scheduling techniques have been adopted. In this study, a two-phase scheduling model known as the clustered grey wolf optimization (CGWO) algorithm is proposed. During the first phase, the task splitting agglomerative clustering (TSAC) algorithm classifies jobs based on their deadlines, while the advanced grey wolf optimization (AGWO) algorithm handles resource allocation. The CloudSim simulation results demonstrate that the CGWO framework outperforms currently used algorithms, including genetic algorithm (GA), particle swarm optimization (PSO), salp swarm algorithm (SSA), and standard grey wolf optimization (GWO). The evaluation considers factors such as makespan, resource utilization, cost, throughput, convergence speed, and others when comparing various cloud scheduling algorithms. The suggested model incorporates a clustering mechanism to alter the traditional first-in, first-out (FIFO) structure of job execution. This study reveals that GA and SSA are excellent choices, particularly for lower and intermediate task counts, if the primary goal is to reduce makespan. If effective resource utilization and throughput are top priorities, SSA and CGWO appear to be promising options. The improvement rate of SSA over CGWO in terms of makespan is approximately 0.135%. Regarding resource utilization, CGWO has shown an improvement rate of 8.228%, 4.88%, and 0.93% compared to GA, GWO, and PSO, respectively. CGWO's rate of resource utilization improvement is 1.28% lower than that of SSA.***

## Keywords
***Cloud computing, Clustering, Scheduling, Resource allocation, Swam intelligence algorithms.***

## 1.Introduction

### 1.1Background

Cloud computing is an example of distributed computing and is one of the most widespread techniques nowadays. It enables the internet-based sharing of resources such as storage, servers, databases, and numerous application services. With the assistance of service providers, anyone can access these underfunded services [1]. Cloud service providers (CSPs), who run data centers (DCs) with powerful servers and infrastructure, offer these services. Software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS) are the three types of cloud service models [2].

CSPs supply virtualized computing resources over the internet in the IaaS model. Amazon web services (AWS), microsoft azure, and google cloud platform (GCP) are well-known IaaS providers. PaaS offers a platform for creating, deploying, and managing applications that include infrastructure, development tools, and services. While the cloud provider takes care of the maintenance and the underlying infrastructure, users may concentrate on creating and deploying applications. Heroku, google app engine, and microsoft azure app service are a few PaaS platform examples. Email services like Gmail, productivity suites like Microsoft 365, and customer relationship management tools like Sales-force are examples of SaaS applications. The shared use of virtual resources, especially IaaS, is the main focus of this study.

*Author for correspondence

### 1.2Challenges

With the support of IaaS, users could construct, manage, and scale their virtualized infrastructure across the web. The effective distribution of these resources to meet workload demands is called resource scheduling. It might be challenging to assign cloud resources to a user [3, 4]. Cloud scheduling issues are NP-hard because of fluctuating workloads, heavy usage of dynamic cloud resources, and variety of optimization objectives. Depending on the circumstances and the requirements, the cloud scheduling problem can change. Task scheduling and resource scheduling are the typical categories. Jobs or tasks are distributed across the available cloud resources using task scheduling. Allocating virtual resources to activities or jobs is the basic goal of resource allocation [5].

Maximizing utilization should be the goal of resource scheduling, which means resources are used to their maximum potential. Over-provisioning of resources, where businesses allocate more virtual resources than they need, is a common problem. When different users or organizations share resources in multi-tenant cloud environments, resource scheduling must balance tenant resource distribution while preserving security, isolation, and fairness. It can be challenging to plan resources in real-time, to match without wasting or over-provisioning resources. The service level agreements (SLAs) are an essential part of cloud scheduling. Resource availability, virtual machines (VMs) resource needs, and load balancing are just a few considerations in resource scheduling. When VMs are allocated and de-allocated, resource fragmentation can occur over time. This results the inefficient use of resources and a rise in scheduling issues [6]. Application performance depends on optimizing network allocation for data transfer and communication. Cost optimization and data security are major concerns. Resource contention can happen when numerous VMs or workloads fight for the same resources [7].

### 1.3Motivation

Making suitable scheduling algorithms and policy choices can be challenging. Organizations must select or create algorithms that meet their requirements because different workloads may demand various scheduling strategies. Different types of cloud scheduling algorithms exist based on the methods and objectives. A few of them are queue-based scheduling algorithms, genetic and evolutionary algorithms (EAs), heuristic algorithms, swarm algorithms and machine learning-based algorithms. Some other commonly used algorithms are static scheduling algorithms, dynamic scheduling algorithms, priority-based scheduling algorithms, load balancing algorithms, deadline-based scheduling algorithms, and queue-based scheduling algorithms. The selection of a cloud scheduling algorithm is influenced by the nature of the workload, resource limitations, performance objectives, and cost concerns. In complicated cloud environments, the allocation of resources can be balanced and optimized via hybrid approaches that incorporate several scheduling algorithms [8].

When tackling optimization issues in complicated search areas, where conventional search techniques may be ineffective or unfeasible swarm intelligence (SI) algorithms are suitable. They have uses in a number of industries, including engineering, banking, logistics, and telecommunications. The collective activity of social creatures like ants, bees, and birds is the inspiration for a technological optimization method known as SI. These algorithms are used to address complicated optimization issues. The SI algorithms identify optimum or nearly optimal solutions through population-based interactions agents [9].

Virtual resource allocation is an optimization problem. Nearly optimal solutions can be obtained by the use of the meta-heuristic algorithm. Most prominent meta-heuristic algorithms are the ant colony optimization (ACO) [10], grey wolf optimization (GWO) [11, 12 ], particle swarm optimization (PSO) [13,14], genetic algorithm (GA), salp swarm algorithm (SSA) [15, 16, 17] etc. GWO is a nature-based SI algorithm.

### 1.4Objectives

This work, introduced a cloud task scheduling algorithm based on the meta-heuristic GWO algorithm. The following are the shortcomings in present scheduling frameworks: (1) the active state is the sole physical machine (PM) state that most systems consider. PMs can be in many states, such as sleep or idle. It could result in resource over-provisioning, waste of resources, and greater expenses. It also causes a chance to induce dynamic workload variability and SLA violations. (2) All tasks in a job are allocated to a single PM. The best VM placement is not guaranteed, nor is the proper use of resources. Resource consumption is inefficient as a result of resource fragmentation. When numerous VMs seek the same resources, resource contention can happen.

(3) The scheduling policy is first-in, first-out (FIFO) (virtual central processing unit (vCPU) requested in a single PM). The proper utilization and allocation of resources are not maintained in the case of FIFO. The FIFO policy has changed since the introduction of clustering. (4) The number of resources required is not checked during PM allocation [18]. The proposed strategy employing VMs or other resource utilization is examined, which causes the management of resources under and over-provisioning. Here, dynamic workload handling serves to ensure balanced multi-tenancy. The availability of resources is checked to maintain the best possible VM placement. Additionally, the cost optimization remains kept up.

### 1.5Contributions

This paper, come up with a new task scheduling algorithm titled clustered grey wolf optimization algorithm (CGWO). The evaluation is done in CloudSim environment. The following are the major contributions:

- Proposed a two-stage work scheduling technique based on a meta-heuristic termed GWO.
- Task splitting agglomerative clustering (TSAC) is used to cluster the jobs before they are submitted to the cloud.
- The second step involved some slight variations of the "encircling" and "attacking" equations, which applied in the proposal using CloudSim.

The remainder of the paper is evaluated as follows. Section 2 discusses an analysis of resource distribution in a cloud context. The architecture of the suggested technique is described in section 3, along with problem formulations. It also provides the illustration of the algorithms and the mathematical models. Section 4 covers the evaluation of the result. Section 5 and section 6 discusses the discussion, conclusion and upcoming work.

## 2.Literature review

The bio-inspired or biologically motivated are frequently used to describe optimization strategies that draw their inspiration from biological processes and natural events. These methods use the behavior and mechanisms seen in live things and ecosystems to address challenging optimization issues. One of the categories of optimization methods inspired by the group behavior of social creatures like bees, ants, birds, and fish called SI algorithms. The SI algorithms have been used in cloud scheduling to attain work distribution, load balancing, and resource optimization [19].

The EAs are a family of optimization algorithms that draw inspiration from the process of natural selection and evolution observed in biological systems. These algorithms are used to find optimal or near-optimal solutions to complex problems by simulating the process of evolution over a population of candidate solutions. EAs are particularly well-suited for optimization problems where the search space is large, complex, and lacks a gradient for optimization [20, 21].

SI is a field examining artificial and natural systems made up of several individuals that are coordinated via decentralized control and self-organization. Simple autonomous entities are part of emergent collective intelligence system. Several fields, including optimization, robotics, data clustering, and routing, employ SI methods [22].

The biological immune system (BIS) serves as the basis for the computational intelligence system known as the artificial immune system (AIS). AIS are a set of computer algorithms and models that use insights from the human immune system to tackle challenging issues in variety of fields, such as optimization, anomaly detection, pattern recognition, and data mining. The adaptive and self-regulating characteristics of the BIS, which protect the body against diseases and develop the ability to identify and respond to new threats, are mimicked in AIS algorithms [23].

Memetic algorithms (MAs) are methods for solving optimization problems by merging local search techniques with GAs. Although MAs have been used in a variety of fields, including cloud computing, to address optimization problems, their success in these applications depends on the particular issue and the surrounding circumstances. The specific problem, how the optimization stated, and how the answers affect how MA are used to cloud computing [24].

Cloud computing is one of the many areas where biochemical algorithms to handle scheduling, resource allocation, and optimization issues. It is crucial to remember that the use of biochemical algorithms in cloud computing necessitates a thorough comprehension of both the particular issue domain and the biological processes. For best outcomes, it may be required to modify and adapt biochemical algorithms to fit the peculiarities of cloud computing settings [25, 26].
Computer models called plant growth simulation algorithms (PGSAs) are based on how plants grow

and develop. It's vital to remember that PGSAs may need extensive adjustment and development in order to be used to cloud computing settings. In their original environment, these algorithms are often utilized for distinct objectives, therefore optimizing and managing cloud resources may require new methodologies and factors to be taken into account [27]. *Table 1* compares algorithms with biological inspiration. In this section, we'll focus on algorithms.

**Table 1**Comparison of optimization techniques

| S. No. | Optimization techniques | Inspired By | Examples |
|---|---|---|---|
| 1 | EA [20,21] | The process of biological evolution, such as natural selection, mutation, and crossover | GA, genetic programming, evolution strategies |
| 2 | SI Algorithms [22] | Collective behaviour and interactions observed in social organisms, such as ants, bees, birds, and fish. | ACO, PSO, GWO, SSA |
| 3 | AIS [23] | The immune system's ability to recognize and respond to foreign substances | Artificial immune network (AIN), Immune network algorithm (INA), clonal selection algorithm (CLONALG) |
| 4 | MA [24] | Information exchange and cooperation to improve the search process. | Genetic local search (GLS), memetic differential evolution (MDE), memetic PSO (MPSO) |
| 5 | Biochemical Algorithms [25,26] | The chemical reactions and signalling pathways that occur within living cells. | Chemical reaction optimization (CRO), artificial chemical reaction optimization (ACRO) |
| 6 | PGSA [27] | The growth and branching patterns observed in plant development to optimize complex systems, such as communication networks and distribution networks. | Whale optimization algorithm (WOA), artificial plant optimization algorithm (APOA), plant propagation algorithm (PPA) |

A group of optimization and problem-solving methods known as SI are motivated by the group behavior of social insects and other animal swarms. By replicating the interactions and collaboration among people in a swarm, these algorithms utilized to solve complicated problems. In this paper, a GWO base scheduling policy which comes under SI. Some prominent SI algorithms are listed below:

Like PSO, ACO is an optimization system draws inspiration from nature. ACO is usually employed to resolve combinatorial optimization issues when the objective is to select the best answer from a limited number of options. An effective optimization system named ACO emerged after studying how ants forage [28].

The GWO is a meta-heuristic algorithm [29, 30]. The simulation findings indicate that using benchmarked test functions improves the proposed optimizer performance. GWO is often used in containerized clouds. In a cloud context, the mean GWO algorithm is employed for work scheduling. The GWO, an optimization algorithm inspired by nature and based on the social behavior of grey wolves, is introduced in [31]. A population-based algorithm called GWO designs the social structure and hunting actions of grey wolves in nature. A population of different choices is iteratively improved using the method for tackling optimization issues.

It has the ability to resolve a variety of optimization issues, including both single- and multi-objective issues. The performance of GWO is assessed against the results of several recognized optimization techniques, particularly differential evolution (DE), GA, and PSO. These benchmarking tests indicate how GWO performs fairly in solution quality and convergence rate. The authors examine how GWO converges to various benchmark functions. They demonstrate that the method frequently converges to solutions that are nearly optimum [32, 33].

By employing GA, it achieves optimal scheduling by treating user pleasure as an objective function and resource credibility as a component of user satisfaction. Subsequently, it includes this scheduling method into Agent and suggests a multi-agent-based cloud computing system design. The numerical outcomes demonstrate that this scheduling method

increases user satisfaction as well as system running efficiency. In the objective function, the technique addresses both user satisfaction and resource credibility. It takes user satisfaction as the optimization aim and incorporates resource credibility into the function of user satisfaction. Then a GA was used to calculate the solutions. The technique can optimize user interests and computation efficiency by taking the aforementioned steps. In order to complete resource scheduling in cloud computing with multi-great agent's autonomy, learning capacity, and sociality, it offered a multi-agent scheduling framework after participating this scheduling approach into agents [34].

GA and PSO are employed to compare the performance of the GWO-based technique PSO. The GWO's makespan and load variation are both minimized. When simulation results are compared to PSO along with standard GWO, the suggested mean GWO processes demonstrate a 9% improvement on PSO and a 3% improvement on normal GWO. It also demonstrates that it lowers energy consumption for a large number of iterations, improving performance. Ahmad presented a hybrid GA-PSO approach. When comparing to GA as well as PSO, the simulation results depict that it reduces the makespan, cost, and balance load.

SSA is a naturally inspired algorithm that mimics the actions of salp in seas or oceans. Identification of the food source is a collective action that creates a food chain. There is a leader at the front of the chain, and the others are followers. Depending on the leader's inspiration, the followers may adjust their stance. The position of the best solution inside the population is designated as the food source position (F) in the chain signifies the best solution for the problem [35]. A nature-inspired optimization approach called PSO is used to discover approximate solutions to challenging optimization issues. It was first created in 1995 by James Kennedy and Russell Eberhart, who took their cue from the flocking or schooling of birds or fish. PSO has been used in a wide range of domains, including economics, robotics, and machine learning, and engineering design. It is popular for being straightforward to use, which makes it a popular option for resolving challenging optimization issues. There are numerous PSO algorithm variations, including global best PSO, local best PSO, constriction coefficients PSO, adaptive PSO, multi-objective PSO, and discrete PSO [36].

Jain and Sharma [37] suggested a binary SSA that is quality of service (QoS) aware. When it comes to minimizing makespan, the QoS aware binary salp swarm algorithm (QBSSA) outperforms the GWO. It also optimizes resource utilization, increases throughput, and reduces average wait times.

A hybrid work of GWO and teaching learning-based optimization was proposed in [38, 39]. In comparison to GWO, PSO, and biography based optimization (BBO) the simulation results show that it maximizes resource consumption and has good load balancing. This GWO technique can be applied to a various technology, which includes the fog computing, internet of things (IoT), and so on. *Table 2* compares algorithms with biological inspiration. This section, focus on SI algorithms.

**Table 2** Comparison of recent optimization techniques.

| S. No. | Optimization techniques | Technique | Inspired By | Examples |
|---|---|---|---|---|
| 1 | ACO [28,29] | SI | nature-inspired, robustness, solves combinatorial problems | convergence speed, parameter tuning, complexity |
| 2 | GWO [30,31] | SI | nature-inspired, few parameters, convergence | limited flexibility, performance variability, limited research |
| 3 | GA [32,33,34] | evolutionary computation | versatility, population diversity, global search, crossover & mutation | parameter tuning, computational intensity, representation choice |
| 4 | PSO [35,36] | SI | simplicity, convergence, adaptability, parallelization | local optima, lack of diversity, parameter sensitivity, limited exploration |
| 5 | SSA [40,41,42] | SI | novel approach, exploration & exploitation | limited application, parameter sensitivity, computational cost |

The study revealed that SI algorithms were used to solve several optimization issues, like cloud scheduling [43, 44].The following are the primary causes:

- Environments for cloud computing are distributed and dynamic. Because the SI algorithms are distributed and decentralized, they can be used to solve real-time cloud scheduling issues.
- These SI algorithms are adept at sifting through search space and identifying globally optimal or nearly optimal solutions.
- SI algorithms are helpful for cloud scheduling because they aim to allocate resources to meet performance metrics.
- They can deal with situations in which resource availability and task execution timeframes are variable.

## 3.Method

Because virtual resources are distributed via the internet, customers can a QBSSA ess them through cloud computing. The PMs are used to build up the virtual resources. A significant issue is allocating suitable virtual resources to clients. The distribution of resources is done in two stages. The first step is to determine which resources are available. The work is completed in the second stage using the resources that have been given.

### 3.1Basic architecture

The customer's request is a group of heterogeneous tasks $T = \{T_1, T_2, T_3, \dots T_n\}$. These tasks execution is done by the allocating VMs say $VM = \{VM_1, VM_2, VM_3, \dots VM_m\}$. The proposed resource allocation algorithm, CGWO provides a near-optimal solution for task allocation problems. The architectural model is demonstrated in the *Figure 1*.

The cloud task manager manages the tasks submitted by the user. The functions of the cloud pre-processing manager comprise the clustering of all enrolled pieces of work and collection of availability, capacity, and cost value of virtual resources present in the DC. This information is at hand to the scheduler. The scheduler makes a study about this information and finds out the appropriate VM allocation to tasks.

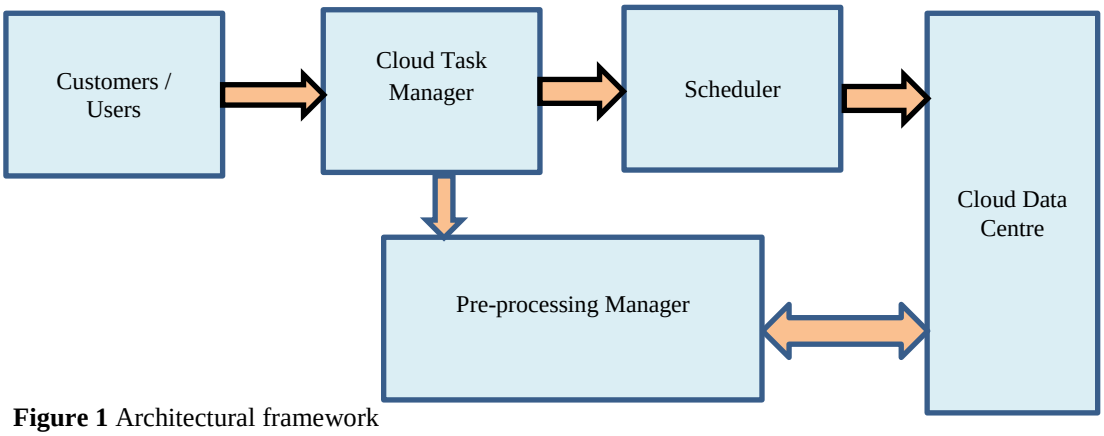



**Figure 1** Architectural framework

This block diagram in *Figure 2* represents the suggested model. Components explained:

User Request: This reflects the requests made for cloud resources like VM, containers, or server-less computing operations by users or applications. These requests may include details regarding the resources needed, performance standards, etc.

Clustering: The clustering divides the request into two categories. The first group of jobs has a short deadline, whereas the second group has a long deadline. For this, TSAC is employed.

The scheduling method advanced grey wolf optimization (AGWO) receives the result from clustering. The scheduling method employed initial solution identification, fitness calculation, hunting phase, exploration phase, termination checking, and final solution identification steps.

Initial population identification: Initial population identification is one of the important tasks in AGWO, and based on this final solution varies. The number of options depends on how many jobs need to be planned.

Fitness calculation: According to objective functions, the fitness value for each schedule is determined. To evaluate fitness value for the initial population λp value is identified, which represents the number of VMs available for the task scheduling. This variable is an entirely novel variable that AGWO has added. The fact that the makespan decreases as tasks are spread across more VMs means that this parameter is inversely proportional to the fitness value.

Hunting phase: Based on the fitness value, this component changes the alpha, beta, and delta solutions. The beta solution comes in second place, the delta comes in third, and the alpha is currently the top solution. These solutions lead the hunting pack.

Exploration phase: The positions of the alpha, beta, and delta solutions are used to update all the scheduling solutions in the pack throughout this phase. This facilitates the search space's exploration and promotes convergence toward optimal solutions.
Termination checking: A maximum number of iterations, a predetermined fitness threshold, or a time limit can all be used as termination conditions. The stopping criterion is also based on the variable A, which falls under 1.

Final solution identification: The algorithm's best solution is often the position of the alpha solution, returned as the best answer to the optimization problem.

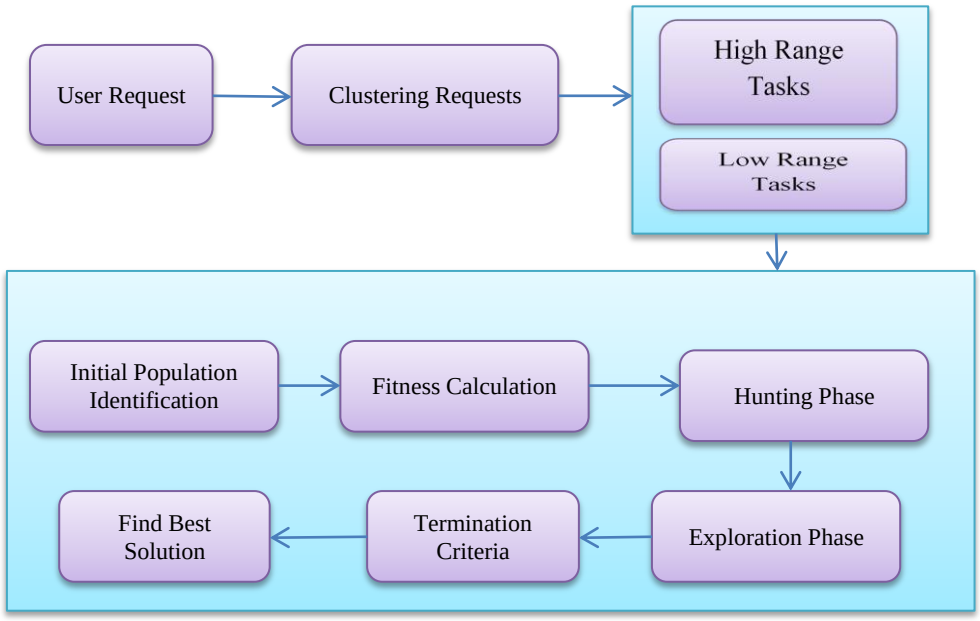



**Figure 2** Complete block diagram

### 3.2Problem formulation

The main objective of the proposed framework is to create a resource allocation algorithm in the cloud environment. The performance metrics under consideration during the implementation of the framework are makespan, cost, etc. The schedule created by the resource allocation algorithm should produce a schedule having minimum Makespan $Obj_1(S)$ and provide efficient utilization of resources, in turn, reduce the cost $Obj_2(S)$. Assume $S$ is the optimal schedule for the problem.

$$Obj(S) = \mu \times Obj_1(S) + (1-\mu) \times Obj_2(S), Where\ 0 < \mu < 1 \quad (1)$$

$$Obj_1(S) = min\ (Makespan\ (S)) \quad (2)$$

$$Obj_2(S) = min\ (Totalcost) \quad (3)$$

The fitness function $Obj(S)$ is given in Equation 1, where μ is a constant its value is 1≥ μ≥0.

#### 3.2.1Makespan

Makespan is defined as the amount of time from start to finish for completing a set of job, that is the maximum completion time of all jobs. It also depends on the CPU computing power. There are m number of VMs and n number of tasks. Assume that 10 tasks are split among 3 separates VMs.VM1 is assigned tasks T1, T2, and T3, VM2 is allocated tasks T4, T5, T6, and T7, and VM3 is allocated the remaining tasks T8, T9, and T10.The finish time represents the fastest these three VMs may possibly finish. The VM 3 (slot

5) finish time is listed below. Here is the start time of VM2 (slot 1), which is the minimum of 10 jobs split among the three VMs. It is shown in *Figure 3*.

| | Slot 1 | Slot 2 | Slot 3 | Slot 4 | Slot 5 |
|---|---|---|---|---|---|
| VM1 | | T1 | T2 | T3 | |
| VM2 | T4 | T5 | T6 | T7 | |
| VM3 | | | T8 | T9 | T10 |

**Figure 3** Task Distributions in VMs

$$Makespan(S) = \sum_{i=0}^{m} \sum_{j=0}^{n} (CPTij - STTij) \quad (4)$$

Where $CPT_j$represents the task completion time,$T_j$represents the task as well as $STT_j$represents the start-time of task $T_j$.The cost value is a function of CPU computing power and load capacity of VM and is calculated using Equation 5. It is a function of processing power and memory needed for the exaction of task.

### 3.2.2Cost

The cost value is a function of CPU computing power and load capacity of VM and is calculated using Equation 5. It is a function of processing power and memory needed for the exaction of task.

$$Total_{cost} = f(vCPU_{Need}, Mem_{Need}) \quad (5)$$

$$TotvCPU_{Need} = ReqvCPU \times vCPU_{cost} \quad (6)$$

$$TotMem_{cost} = Req_{Mem} \times Mem_{cost} \quad (7)$$

### 3.2.3Resource utilization

It is calculated using a VM's task completion time, makespan, and total number of VMs. It is calculated by using Equation 8.

$$Resource\ utilization = \sum_{i=1}^{m} \left( \frac{time\ taken\ by\ VM_i\ to\ complete\ tasks}{Makespan * m} \right) \quad (8)$$

### 3.2.4Throughput

The number of tasks completed in a particular amount of time is referred to as throughput. Here it is calculated using Equation 9.

$$T\square roug\square put = No\ of\ tasks\ completed\ successfully/Time\ taken \quad (9)$$

### 3.2.5Convergence speed

The term "convergence speed" describes the rate at which a scheduling algorithm reaches an ideal or nearly ideal solution. It is frequently quantified in terms of convergence time or iteration count. Faster convergence is indicated by a lower value.

$$Convergence\ speed = Time\ taken\ to\ reac\square\ convergence \quad (10)$$

## 3.3Task scheduling model

Task entry, clustering, and optimization are all part of the proposed paradigm. Customer requests are entered into a task entry queue during the first phase. These duties will be distributed across virtual resources. The clustering strategy in the proposed framework is TSAC, which is a version of hierarchical agglomerative clustering that is TSAC. The optimization step employs AGWO, a modified version of GWO. *Figure 4* depicts the situation.

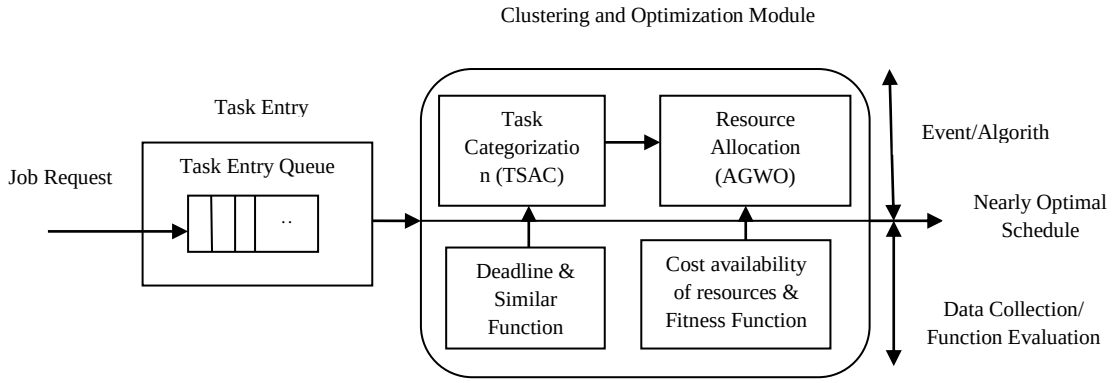



**Figure 4** Architecture of CGWO framework

## 3.4Task splitting agglomerative clustering

One of the key components of the proposed system is clustering. The jobs that will be scheduled are divided into two categories in this clustering section. Tasks with a long deadline are placed in one of two categories: high range or low range. For example, imagine there are six tasks T1, T2, T3, T4, T5, T6 to be scheduled, each with a deadline of 2, 5, 10-, 20-,

15-, or 3-time units. According to the clustering method, T1, T2, and T6 are in the low range category, and T3, T4, and T5 are in the high range category.

$$Sim(T_i, T_j) = MaxSim(D_{tx}, D_{ty}) \quad (11)$$

Where $tx \in T_i$ and $ty \in T_j$

The TSAC algorithm initially considers each scheduling tasks as a single cluster and find out the similarity function using Equation 8. $D_{tx}$ and $D_{ty}$ are the deadlines for tasks $tx$ and $ty$, respectively, in Equation 8. Examine how the deadlines for tasks $tx$ and $ty$ are comparable. Repeat the process until there are only two clusters left. Algorithm 1 depicts the clustering algorithm.

Algorithm 1: Task Splitting Agglomerative Clustering
Input: Tasks to be scheduled.
Output: Two clusters, high range and low range
Initially each task makes a cluster search space.
Set CL as total number of clusters.
Evaluate $MaxSim(D_{tx}, D_{ty})$ using Equation 11
While ($C_L \leq 2$):
    Merge the two closest clusters
    Update $Sim(T_i, T_j)$
    Decrement $C_L$ by 1
End while

## 3.5Advanced grey wolf optimization

The encircling and attacking formulae are slightly adjusted in this proposed strategy. Based on the distribution of jobs across cloud DC, a priority value is introduced. The rest of the methods appear to be normal GWO. *Table 3* lists the parameters utilised in the technique. The calculation of the fitness value is an important aspect of this technique. It shows how far the agent has come towards aim.

**Table 3** Variables used

| Parameters | Meaning |
|---|---|
| r1, r2 | random values in [0,1] |
| $\vec{a}$ | random value decreases from 2 to 0 |
| $\vec{P}_p$ ( t ) | Position of prey |
| $\vec{P}$( t ) | Position of Wolf |
| P(t+1) | Fitness value |
| $P_\alpha$ | Best search Agent |
| $P_\beta$ | Second Best Search Agent |
| $P_\delta$ | Third Best Search Agent |
| $\lambda_p, \lambda_\alpha, \lambda_\beta, \lambda_\delta$ | Number of VMs used for task distribution |
| $CP_{Tij}$ | Completion Time of Task Tj in $VM_i$ |
| $ST_{Tj}$ | Start-Time of Task $T_j$in$VM_i$ |
| $D_{tx}$ | Deadlines for tasks $t_x$ |
| $D_{ty}$ | Deadlines for tasks $t_y$ |
| $vCPU_{Need}$ | Processing Power |
| $Mem_{Need}$ | Memory Needed |

### 3.5.1 Encircling the prey

Encircling the prey entails determining the fitness value that is closest to the target. When a project or task is divided among available VMs in a cloud scheduling problem, utilization efficiency improves. The following equations suggest a mathematical model for encircling behaviour. The suggested model includes a variableλ_p, which represents the number of VMs utilized for job/task distribution data. The aim is inversely proportional to it. The agent is getting closer to the goal as its values rise.

$$\vec{D} = \left|\vec{C}.\vec{P}_p(t) - \left[\vec{P}_p(t)/\lambda_p\right]\right| \quad (12)$$

$$\vec{P}(t+1) = \left|\vec{P}_p(t) - \vec{A}\vec{D}\right| \quad (13)$$

Where $\lambda_p$ is the number of VMs used for task allocation over the cloud. $\vec{P}_p(t)$is the prey's position, $\vec{P}(t)$) signifies the wolf position and P(t+1) Equation 13 is fitness value. The vector $\vec{A}$ as well as$\vec{D}$ Equation 12 are estimated as equated in 14 and 15.

$$\vec{A} = 2\,\vec{a}\,.r_1 - \vec{a} \quad (14)$$

$$\vec{C} = 2.r_2 \quad (15)$$

The best three groups, alpha (α), beta (β), as well as delta (δ), finish this phase. The best groups are determined by assessing the fitness value of each schedule used in the encircling phase. The alpha(α) emerges as the finest among them. The remaining answers are placed in the omega (ω) category.

### 3.5.2 Hunting and attacking

During the hunting phase, activity is rescheduled rooted on the mean of alpha, beta, as well as delta. This will continue until the vector A falls below one, which is the attacking condition. The alpha has now become the best option. The following are the equations (Equation 16 to Equation 22) utilized in the hunting phase:

$$\vec{D}_\alpha = \left|C_1\vec{P}_\alpha - \left[\vec{P}(t)/\lambda_\alpha\right]\right| \quad (16)$$

$$\vec{D}_\beta = \left|C_1\vec{P}_\beta - \left[\vec{P}(t)/\lambda_\beta\right]\right| \quad (17)$$

$$\vec{D}_\delta = \left|C_1\vec{P}_\delta - \left[\vec{P}(t)/\lambda_\delta\right]\right| \quad (18)$$

$$\vec{P}_1 = \vec{P}_\alpha - \vec{A}_1\vec{D}_\alpha \quad (19)$$

$$\vec{P}_2 = \vec{P}_\beta - \vec{A}_1\vec{D}_\beta \quad (20)$$

$$\vec{P}_3 = \vec{P}_\delta - \vec{A}_1\vec{D}_\delta \quad (21)$$

$$\vec{P}(t+1) = \left(\vec{P}_1 + \vec{P}_2 + \vec{P}_3\right)/3 \quad (22)$$

Algorithm 2 depicts the AGWO Algorithm's pseudo-code form.

Algorithm 2: Advanced Grey Wolf Optimization Algorithm

Input: Tasks to be scheduled.

Output: nearly optimal Schedule.

Initialize the population based on number of task n

Initialize *a, A and C*

Categorize the tasks using TSAC Algorithm

Availability of resource is checked and select task category.

Set the initial iteration t=1

for k=1 to n do

Generate Initial Populations $P_i$

End for

Estimate the fitness function for each schedule using fitness function Equation 1.

Identify the best solutions $P_{\alpha,} P_{\beta ,} P_{\delta.}$

While (t<max(t))

for each schedule

Update the position of current schedule using $\vec{P}$ (t+1) Equation 22.

End for

Update a, A and C.

Estimate the schedule fitness

Update best solutions $P_{\alpha,,} P_{\beta ,} P_{\delta.}$

t=t+1

End while

return best solution $P_{\alpha}$.

## 4.Results

This section details the tests that were conducted to assess the suggested method's performance.

### 4.1Experimental setup

The simulation results were analyzed on HP PC with an Intel CORE i5 8th Gen x64 processor and 8GB of RAM. The CloudSim simulator is used to examine the proposed framework's performance. The CloudSim 4.0 toolkit is compatible with the jdk 1.8.0_111 package and operates on the Windows 10 platform using NetBeans IDE 8.2.

CloudSim allows the dynamic creation of tasks using the concept of Cloudlets. It also supports the dynamic creation of heterogeneous VM instances. In our proposed model the capabilities of VMs such as available memory, the processing speed of VCPU are variant. To evaluate CGWO several tasks are managed with various heterogeneous sets of tasks and VMs.

### 4.2Evaluation of results

The performance results in comparison of the proposed CGWO are with the GA, PSO [10], SSA, and Standard GWO [9] concerning makespan for task scheduling problem. The simulations parameters are given in *Table 4*. The datasets considered for the evaluation amount to the count of tasks vary from 5 to 1000. The simulation result is obtained by executing the tasks 20 times, and by taking the average value obtained from the execution. The simulation environment has 30 PM, 86 VMs distributed over this PMs and a totally of 166 vCPUs. The task length parameter specifies the computational load of each task in terms of millions of instructions per second (MIPS). The job length might range from 100 to 1000 MIPS, reflecting various computational needs. Each virtual computer has access to random access memory (RAM). Here, the range is defined as 256 MB to 1024 MB. The computing power of the virtual CPU is measured in MIPS by the capacity of vCPU. The vCPUs have a 500–1000 MIPS performance range. The VM policy indicated above is time-shared. The method by which numerous VMs divide up the same physical resources (such CPU, memory, etc.) for use is known as time-sharing.

**Table 4** Simulation parameters

| **Parameters** | **Values** |
|---|---|
| Count of Tasks | 5-1000 |
| Count of VMs | 86 |
| Count of PMs | 30 |
| Task length | 100-1000 MIPS |
| RAM | 256-1024 (MB) |
| Capacity of vCPU | 500-1000MIPS |
| Number of Iterations | 20 |
| VM Policy | Time Shared |

**Makespan**-The makespan value would be a specified amount of time expressed in seconds, minutes, or any other suitable unit, and would reflect the entire amount of time needed to accomplish all tasks. It estimates the total duration for execution. Better performance is symbolised by a lower makespan.

*Figure 4* shows the makespan value obtained by applying CGWO, GA, PSO [10], SSA, and GWO. *Table 5* represents the tabular representation of makespan. The graph *Figure 5* shows the number of tasks on the x-axis and the makespan value in seconds on the y-axis. The makespan should be kept to a minimum. Resource Utilization- The percentage of computing resources (such as CPU, memory, or network) that are used during task execution is often used to describe resource utilization. When the percentages are higher, resource utilization is higher.

*Figure 6* and *Table 6* shows the comparison of GA, PSO, Standard GWO, SSA, with the proposed

method, in terms of resource utilization. The x-axis represented the number of tasks, while the y-axis parameter varied as resource utilization. The VMs limited to 50. It can be seen from the comparison that SSA uses its resources more efficiently. Comparing the CGWO to the normal GWO, improvements can be seen.

**Table 5** Makespan in second

| Number of tasks | PSO | GWO | GA | SSA | CGWO |
|---|---|---|---|---|---|
| **5** | 24.05 | 22.11 | 20.12 | 19.35 | 20.54 |
| **100** | 54.31 | 53.35 | 52.31 | 51.87 | 54.67 |
| **200** | 130.12 | 120.24 | 110.42 | 114.33 | 115.53 |
| **500** | 419.33 | 410.38 | 388.63 | 385.56 | 386.73 |
| **1000** | 785.42 | 774.46 | 761.54 | 755.43 | 750.86 |

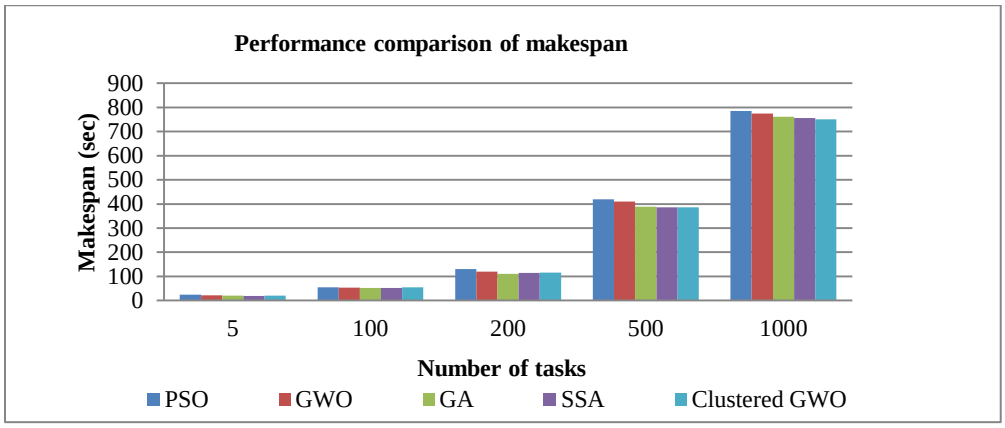



**Figure 5** Comparison of Makespan

**Table 6** Resource utilization in %

| Number of tasks | GA | Standard GWO | PSO | SSA | CGWO |
|---|---|---|---|---|---|
| **5** | 63.64 | 66.51 | 70.56 | 72.86 | 72.16 |
| **100** | 69.38 | 72.32 | 75.55 | 77.75 | 72.75 |
| **200** | 71.27 | 74.67 | 77.71 | 79.79 | 80.76 |
| **500** | 76.56 | 80.54 | 82.66 | 82.74 | 82.44 |
| **1000** | 79.76 | 80.56 | 82.81 | 84.83 | 84.83 |

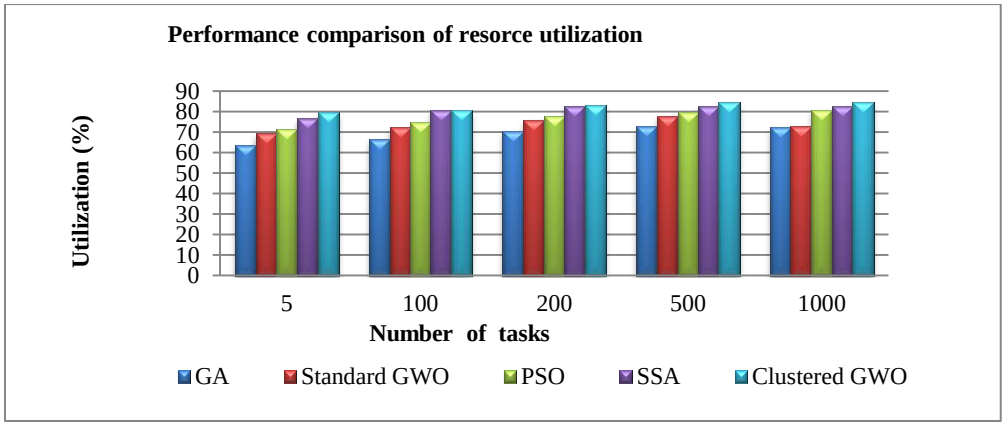



**Figure 6** Comparison of resource utilization

**Cost**-The cost function of CGWO is measured against varying a number of tasks. The cost value is calculated on the dynamic availability of resources and the individual cost of resources. As the count of

task increases the cost value decreases means the good resource utilization and is depicted in *Figure 7*. In terms of total cost, *Figure 8* compares GA, PSO, standard GWO, as well as SSA to the proposed approach. The number of tasks was displayed on the x-axis, while the total cost was indicated on the y-axis. The simulation values for the total cost of GA, PSO, SSA, Standard GWO, and CGWO when there are 500 tasks are 54, 46, 44, 50.5, and 44.1, respectively. In comparison to PSO and traditional GWO, *Figure 6* indicates that the proposed technique has the lowest cost.

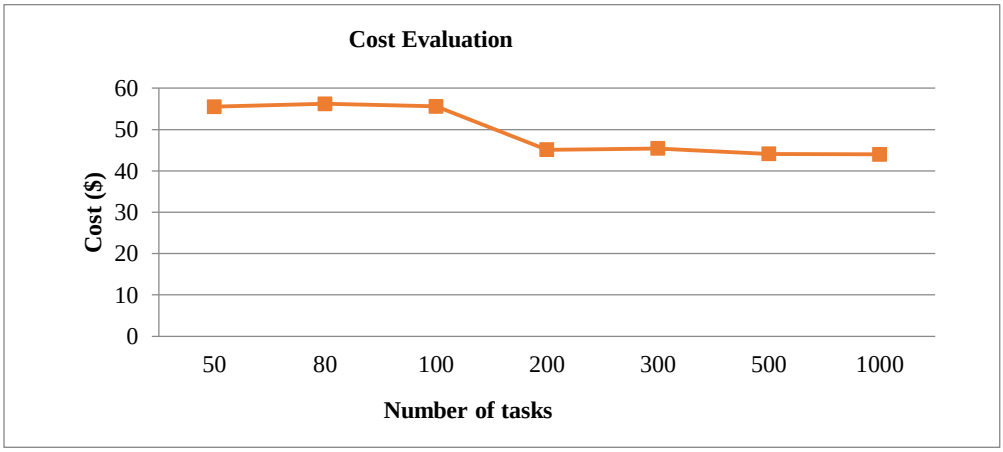



**Figure 7** Comparison of resource utilization

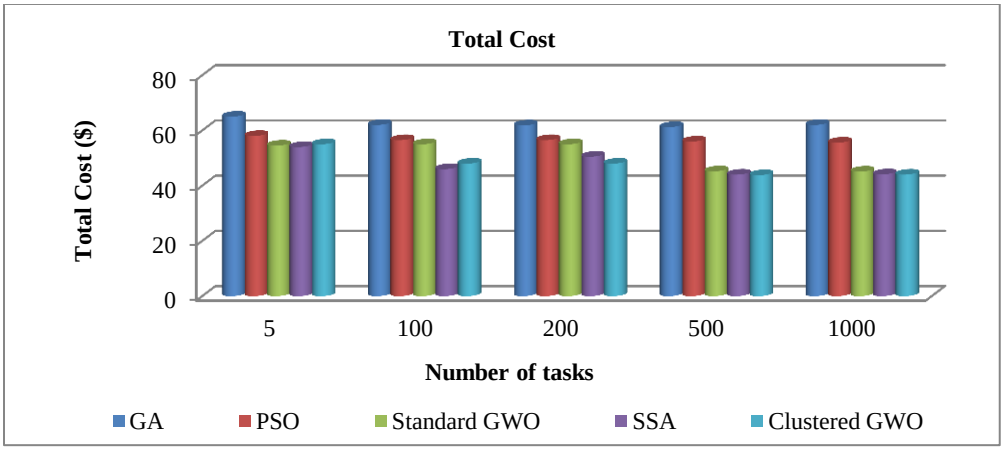



**Figure 8** Cost evaluation

Throughput - The quantity of jobs completed in a certain amount of time, such as seconds or minutes, is known as throughput. Higher values suggest a processing capacity that is greater.

The CGWO method appears to consistently obtain the maximum throughput values across various numbers of tasks, according to the presented data. *Figure 9* displays the results of various heuristic methods in terms of throughput. When there are 500 jobs to schedule, GA, GWO, PSO, SSA, and CGWO have throughput values of 180,170,170,200,200 correspondingly. The table representations of throughput values are shown in *Table 7*.

Here, *Table 7* exposes the performance of CGWO over other traditional models like GA, PSO, Standard GWO, and SSA in terms of throughput. Besides, the experimentation confirmed the proposed CGWO outperformed the other conventional models and proved its efficacy.

**Table 7** Throughput

| Number of tasks | GA | Standard GWO | PSO | SSA | CGWO |
|---|---|---|---|---|---|
| **5** | 5 | 5 | 5 | 5 | 5 |
| **100** | 80 | 85 | 90 | 90 | 90 |
| **200** | 100 | 120 | 150 | 150 | 148 |
| **500** | 180 | 170 | 170 | 200 | 200 |
| **1000** | 200 | 220 | 235 | 252 | 250 |

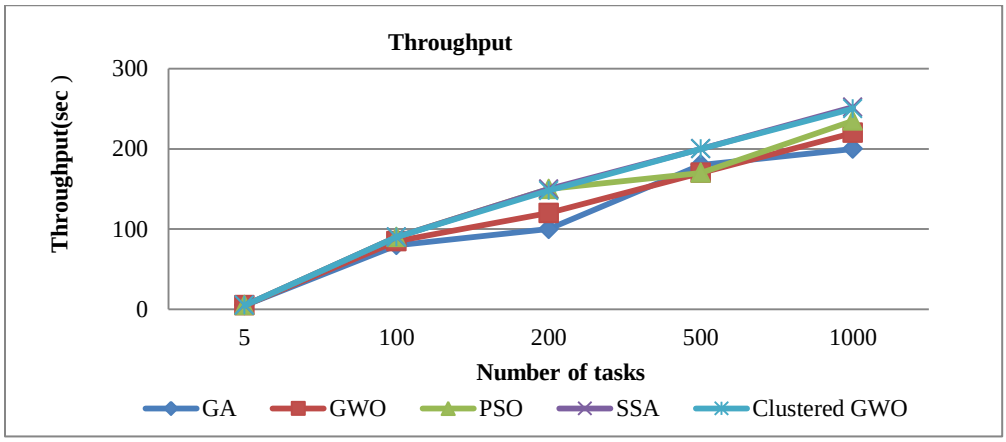
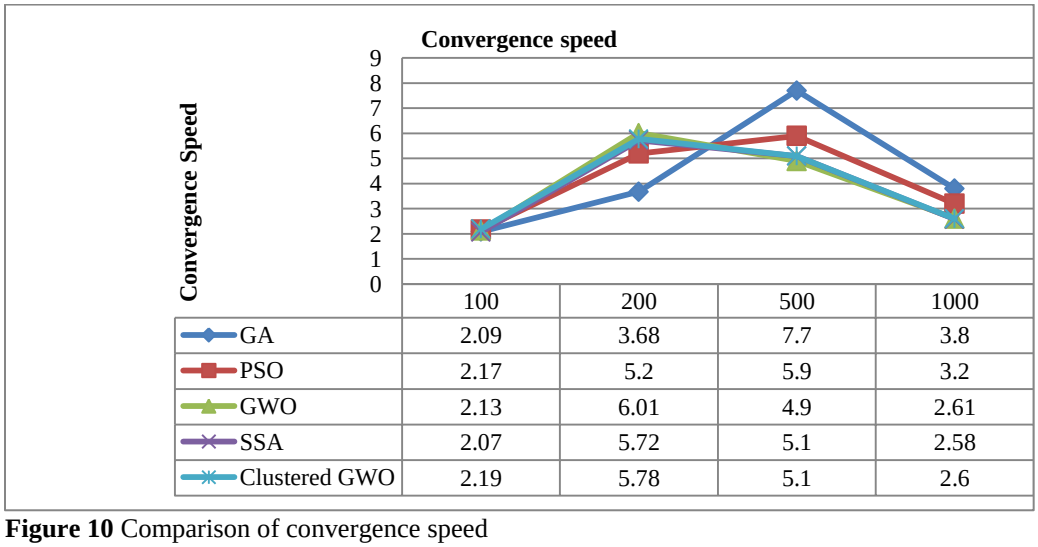



**Figure 9** Comparison of throughput

Convergence Speed: The rate at which a scheduling algorithm achieves an ideal or very close to ideal solution is referred to as convergence speed. The faster a problem converges to its optimal or nearly optimal state, the lower its Convergence Speed value. For GA, PSO, GWO, SSA, and CGWO, the corresponding numbers of iterations are 30, 25, 20, and 20, respectively. We can compare the convergence speeds of various algorithms like GA, PSO, GWO, SSA, and CGWO based on the information provided for their values at various issue sizes. Based on the given data representing convergence speed values for different algorithms (GA, PSO, GWO, SSA, and CGWO) at different problem sizes, we can compare their convergence speeds.




| | 100 | 200 | 500 | 1000 |
|---|---|---|---|---|
| GA | 2.09 | 3.68 | 7.7 | 3.8 |
| PSO | 2.17 | 5.2 | 5.9 | 3.2 |
| GWO | 2.13 | 6.01 | 4.9 | 2.61 |
| SSA | 2.07 | 5.72 | 5.1 | 2.58 |
| Clustered GWO | 2.19 | 5.78 | 5.1 | 2.6 |

**Figure 10** Comparison of convergence speed

The convergence speed representations are displayed in the *Figure 10* for various issue sizes, and each algorithm is represented by a different column. The algorithm converges to an optimal or nearly ideal solution more quickly the lower the convergence speed set. These conclusions can be drawn from the

presented data. In comparison to the other methods, GA and SSA show considerably faster convergence speeds for issue sizes of 100 and 200.GWO and CGWO show relatively higher convergence speeds than the other methods for issue sizes of 500 and 1000. The PIR %for the makespan of the proposed CGWO, GA, PSO, SSA, and Standard GWO are shown in *Table 8*. It could be observed that the makespan values obtained through Clustered-GWO are better than compared to other methods. The PIR of makespan sketches the proposed CGWO produces 0.135%,0.488%,3.565%, and 2.368% over SSA, GA, GWO, and PSO.

**Table 8** Performance improvement rate (%)-makespan

| | **PSO** | **GWO** | **GA** | **SSA** | **CGWO** |
|---|---|---|---|---|---|
| **Total Makespan** | 1413.23 | 1380.54 | 1333.02 | 1326.54 | 1328.33 |
| **PIR % over PSO** | NA | 2.368 | 6.017 | 6.535 | 6.391 |
| **PIR % over Std. GWO** | NA | NA | 3.565 | 4.071 | 3.930 |
| **PIR % over GA** | NA | NA | NA | 0.488 | 0.353 |
| **PIR % over SSA** | NA | NA | NA | NA | 0.135 |

The performance improvement rate (PIR %) for the utilization of resources of the proposed CGWO, GA, PSO, SSA, and Standard GWO are shown in *Table 9*. The PIR of resource utilization illustrates the proposed CGWO produces -1.28%,2.181%,3.774%, and 3.735% over SSA, PSO, standard GWO, and GA.

**Table 9** Performance improvement rate (%) - resource utilization

| | **GA** | **Standard GWO** | **PSO** | **SSA** | **CGWO** |
|---|---|---|---|---|---|
| **Total** | 360.61 | 374.6 | 389.29 | 397.97 | 392.94 |
| **PIR % over GA** | NA | 3.735 | 7.367 | 9.388 | 8.228 |
| **PIR % over Std. GWO** | NA | NA | 3.774 | 5.872 | 4.667 |
| **PIR % over PSO** | NA | NA | NA | 2.181 | 0.929 |
| **PIR % over SSA** | NA | NA | NA | NA | -1.28 |

Though the proposed CGWO framework shows some improvements with the existing models it has some limitations. The main limitations are that this framework is not tested with real-time environments. Also, this work considers only the factors such as the makespan, cost, resource utilization and throughput. It does not consider the factors such as the load balance, time etc.

## 5.Discussion

To analyze the effectiveness of the suggested cloud resource scheduling model, assessment matrices such as cost, resource usage, makespan with time, throughput, and convergence speed were taken into consideration.

All optimization strategies tend to have an increasing makespan as the number of jobs rises. The PSO algorithm, followed by GWO, GA, and SSA, has the longest makespan under most conditions. When additional jobs taken into account, CGWO consistently has the smallest makespan among the algorithms. SA typically has the smallest makespan values across all problem sizes, which indicates faster work completion. For the utilization of resources, the following observations are made. Higher resource utilization levels are typically seen in SSA, CGWO, and PSO, showing superior performance in terms of effective resource allocation and utilization. When compared to the other algorithms, GA and standard GWO show substantially lower resource utilization numbers. SA and CGWO consistently exhibit the greatest resource utilization rates across all task amounts, with CGWO reaching 84.83% at 1000 tasks. In comparison to other algorithms, GA often uses fewer resources, especially when the number of tasks is small.

The cost value is determined by factoring the resources' individual costs and their dynamic availability. It shows that when the number of tasks rises, the cost value falls, indicating efficient use of resources. The cost comparison of CGWO seems to be an initial increasing trend, indicating that some setup costs are associated with a smaller number of tasks. The cost begins to drop after a certain number of jobs around 200 tasks, possibly economies of scale or resource optimization.

The throughput statistics provided allow us to draw the following conclusions: All algorithms attain the same throughput number for small problem sizes (5

and 100), indicating equivalent performance in terms of task/job completion rate. SSA, CGWO, and PSO typically provide greater throughput values for bigger problem sizes (200, 500, and 1000), indicating superior performance in terms of processing tasks/jobs at a higher pace. For bigger problem sizes, GA and GWO show relatively lower throughput values than the other techniques. Based on the required level of throughput and the number of jobs in a particular optimization issue, this data can be useful for choosing the most appropriate method.

GA shows the lowest convergence speed number for problems of size 100, indicating a faster convergence rate than the other algorithms. GWO shows the maximum convergence speed value for issue sizes 200 and 500, indicating slower convergence than the other algorithms. All algorithms have relatively low values for convergence speed for problems of size 1000, indicating similar convergence rates.

The observed weaknesses of SI algorithms are covered in the paragraph below. SI algorithms can take longer to decide on other approaches. These algorithms frequently need a variety of parameters, including population size, convergence criteria, and equilibrium between exploration and exploitation. Finding the ideal parameter settings might be challenging. The best answer may not always be found by SI systems, which only provide heuristic solutions. Despite being parallelized; SI approaches may not be able to scale in cloud systems due to the abundance of resources and tasks. The performance of SI algorithms may vary based on the initial conditions. SI methods can be difficult to implement and comprehend, necessitating knowledge of the two algorithms and the issue domain. A complete list of abbreviations is shown in *Appendix I.*

## 6.Conclusion and future work

This study described CGWO as a resource allocation approach for cloud resource scheduling. In the proposed model, a clustering mechanism was introduced to modify the FIFO structure of task execution. The statuses of the VMs were also considered, as resource availability played a significant role in job allocation. The workload was distributed across DCs. The analysis of the proposed model was based on performance metrics, including makespan, resource utilization, cost, throughput, and convergence speed. In summary, the choice of the most suitable optimization algorithm depended on the specific objectives and trade-offs within the optimization problem. If the primary goal was to minimize makespan and resource utilization was not a critical concern, GA or SSA may be appropriate choices. On the other hand, if resource utilization was a primary concern, GA might be preferred despite a slightly longer makespan. Future research projects will focus on essential factors such as flow time and load balance during the scheduling of tasks and jobs. Since SSA demonstrated superiority over GWO in the experimental study, a shift to SSA instead of GWO is planned. Additionally, the results obtained were based solely on CloudSim, but further validation will be performed in real-world environments.

### Acknowledgment

None.

### Conflicts of interest

The authors have no conflicts of interest to declare.

### Author's contribution statement

**Juliet A Murali:** Study conception, data collection, investigation on challenges design, analysis and interpretation of results, draft manuscript preparation, original draft, writing – review and editing. **Brindha T:** Study conception, data collection, investigation on challenges, supervision, manuscript correction.

### References

[1] Chang X, Xia R, Muppala JK, Trivedi KS, Liu J. Effective modeling approach for IaaS data center performance analysis under heterogeneous workload. IEEE Transactions on Cloud Computing. 2016; 6(4):991-1003.

[2] Khazaei H, Mišić J, Mišić VB, Rashwand S. Analysis of a pool management scheme for cloud computing centers. IEEE Transactions on Parallel and Distributed Systems. 2012; 24(5):849-61.

[3] Khazaei H, Miic J, Miic VB, Mohammadi NB. Modeling the performance of heterogeneous IaaS cloud centers. In 33rd international conference on distributed computing systems workshops 2013 (pp. 232-7). IEEE.

[4] Wang B, Chang X, Liu J. Modeling heterogeneous virtual machines on IaaS data centers. IEEE Communications Letters. 2015; 19(4):537-40.

[5] Ghosh R, Longo F, Naik VK, Trivedi KS. Modeling and performance analysis of large scale IaaS clouds. Future Generation Computer Systems. 2013; 29(5):1216-34.

[6] Guo P, Bu LL. The hierarchical resource management model based on cloud computing. In IEEE symposium on electrical & electronics engineering 2012 (pp. 471-4). IEEE.

[7] Wadhonkar A, Theng D. A survey on different scheduling algorithms in cloud computing. In 2nd international conference on advances in electrical,

electronics, information, communication and bio-informatics 2016 (pp. 665-9). IEEE.
[8] Bansal N, Singh AK. Valuable survey on scheduling algorithms in the cloud with various publications. International Journal of System Assurance Engineering and Management. 2022; 13(5):2132-50.
[9] Manasrah AM, Ba AH. Workflow scheduling using hybrid GA-PSO algorithm in cloud computing. Wireless Communications and Mobile Computing. 2018; 2018:1-6.
[10] Tomás L, Tordsson J. An autonomic approach to risk-aware data center overbooking. IEEE Transactions on Cloud Computing. 2014; 2(3):292-305.
[11] Natesan G, Chokkalingam A. Optimal task scheduling in the cloud environment using a mean grey wolf optimization algorithm. International Journal of Technology. 2019; 10(1):126-36.
[12] Natesan G, Chokkalingam A. Task scheduling in heterogeneous cloud environment using mean grey wolf optimization algorithm. ICT Express. 2019; 5(2):110-4.
[13] Attiya I, Zhang X. A simplified particle swarm optimization for job scheduling in cloud computing. International Journal of Computer Applications. 2017; 163(9):34-41.
[14] Huang CL, Yeh WC. A new SSO-based algorithm for the bi-objective time-constrained task scheduling problem in cloud computing services. arXiv preprint arXiv:1905.04855. 2019.
[15] Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM. Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Advances in Engineering Software. 2017; 114:163-91.
[16] Tolba M, Rezk H, Diab AA, Al-dhaifallah M. A novel robust methodology based salp swarm algorithm for allocation and capacity of renewable distributed generators on distribution grids. Energies. 2018; 11(10):1-34.
[17] Abusnaina AA, Ahmad S, Jarrar R, Mafarja M. Training neural networks using salp swarm algorithm for pattern classification. In proceedings of the 2nd international conference on future networks and distributed systems 2018 (pp. 1-6). ACM.
[18] Bruneo D. A stochastic model to investigate data center performance and QoS in IaaS cloud computing systems. IEEE Transactions on Parallel and Distributed Systems. 2013; 25(3):560-9.
[19] Nssibi M, Manita G, Korbaa O. Advances in nature-inspired metaheuristic optimization for feature selection problem: a comprehensive survey. Computer Science Review. 2023; 49:100559.
[20] Mashwani WK. Comprehensive survey of the hybrid evolutionary algorithms. International Journal of Applied Evolutionary Computation. 2013; 4(2):1-9.
[21] Hua Y, Liu Q, Hao K, Jin Y. A survey of evolutionary algorithms for multi-objective optimization problems with irregular Pareto fronts. IEEE/CAA Journal of Automatica Sinica. 2021; 8(2):303-18.
[22] Kaur K, Kumar Y. Swarm intelligence and its applications towards various computing: a systematic review. In international conference on intelligent engineering and management 2020 (pp. 57-62). IEEE.
[23] Yang H, Liang YW, Chen J. Definition of danger signal in artificial immune system with cloud method. In fourth international conference on natural computation 2008 (pp. 644-7). IEEE.
[24] Alsmady A, Al-khraishi T, Mardini W, Alazzam H, Khamayseh Y. Workflow scheduling in cloud computing using memetic algorithm. In Jordan international joint conference on electrical engineering and information technology 2019 (pp. 302-6). IEEE.
[25] Kapur R. Review of nature inspired algorithms in cloud computing. In international conference on computing, communication & automation 2015 (pp. 589-94). IEEE.
[26] Yang T, Zhao Y. Application of cloud computing in biomedicine big data analysis cloud computing in big data. In international conference on algorithms, methodology, models and applications in emerging technologies 2017 (pp. 1-3). IEEE.
[27] Rao MS, Modi S, Singh R, Prasanna KL, Khan S, Ushapriya C. Integration of cloud computing, IoT, and big data for the development of a novel smart agriculture model. In international conference on advance computing and innovative technologies in engineering 2023 (pp. 2779-83). IEEE.
[28] Liu CY, Zou CM, Wu P. A task scheduling algorithm based on genetic algorithm and ant colony optimization in cloud computing. In 13th international symposium on distributed computing and applications to business, engineering and science 2014 (pp. 68-72). IEEE.
[29] Abdel-basset M, Abdle-fatah L, Sangaiah AK. An improved lévy based whale optimization algorithm for bandwidth-efficient virtual machine placement in cloud computing environment. Cluster Computing. 2019; 22:8319-34.
[30] Nadimi-shahraki MH, Taghian S, Mirjalili S. An improved grey wolf optimizer for solving engineering problems. Expert Systems with Applications. 2021; 166:113917.
[31] Patel D, Patra MK, Sahoo B. Gwo based task allocation for load balancing in containerized cloud. In international conference on inventive computation technologies 2020 (pp. 655-9). IEEE.
[32] Kumar A, Bawa S. Generalized ant colony optimizer: swarm-based meta-heuristic algorithm for cloud services execution. Computing. 2019; 101(11):1609-32.
[33] Malekloo MH, Kara N, El BM. An energy efficient and SLA compliant approach for resource allocation and consolidation in cloud computing environments. Sustainable Computing: Informatics and Systems. 2018; 17:9-24.
[34] Tseng FH, Wang X, Chou LD, Chao HC, Leung VC. Dynamic resource prediction and allocation for cloud data center using the multiobjective genetic algorithm. IEEE Systems Journal. 2017; 12(2):1688-99.
[35] Shadravan S, Naji HR, Bardsiri VK. The sailfish optimizer: a novel nature-inspired metaheuristic

algorithm for solving constrained engineering optimization problems. Engineering Applications of Artificial Intelligence. 2019; 80:20-34.
[36] Del VY, Venayagamoorthy GK, Mohagheghi S, Hernandez JC, Harley RG. Particle swarm optimization: basic concepts, variants and applications in power systems. IEEE Transactions on evolutionary computation. 2008; 12(2):171-95.
[37] Jain R, Sharma N. A QoS aware binary salp swarm algorithm for effective task scheduling in cloud computing. In proceedings of progress in advanced computing and intelligent engineering 2021 (pp. 462-73). Springer Singapore.
[38] Mousavi SM, Moghadasi M, Fazekas G. Dynamic resource allocation using combinatorial methods in cloud: a case study. In 8th international conference on cognitive infocommunications 2017 (pp.73-8). IEEE.
[39] Mavrovouniotis M, Li C, Yang S. A survey of swarm intelligence for dynamic optimization: algorithms and applications. Swarm and Evolutionary Computation. 2017; 33:1-7.
[40] Huang CL, Jiang YZ, Yin Y, Yeh WC, Chung VY, Lai CM. Multi objective scheduling in cloud computing using MOSSO. In congress on evolutionary computation 2018 (pp. 1-8). IEEE.
[41] Ibrahim RA, Ewees AA, Oliva D, Abd EM, Lu S. Improved salp swarm algorithm based on particle swarm optimization for feature selection. Journal of Ambient Intelligence and Humanized Computing. 2019; 10:3155-69.
[42] Hegazy AE, Makhlouf MA, El-tawel GS. Improved salp swarm algorithm for feature selection. Journal of King Saud University-Computer and Information Sciences. 2020; 32(3):335-44.
[43] Supreeth S, Patil K, Patil SD, Rohith S, Vishwanath Y, Prasad KS. An efficient policy-based scheduling and allocation of virtual machines in cloud computing environment. Journal of Electrical and Computer Engineering. 2022; 2022:1-12.
[44] Lipsa S, Dash RK, Ivković N, Cengiz K. Task scheduling in cloud computing: a priority-based heuristic approach. IEEE Access. 2023; 11:27111-26.

**Juliet A. Murali**, received her M.Tech in Computer Science and Engineering from Karunya Institute of Technology and Sciences. She is a research scholar at Noorul Islam Centre for Higher Education, Kumarakovil, Thuckalay, Kanyakumari District, Tamil Nadu, India. She completed her B.Tech in Information Technology from Mahatma Gandhi University, India. Her current areas of interest include Cloud Computing, Parallel Processing, and distributed computing.
Email: jullietjulli123@gmail.com

**Dr. Brindha T** received her Ph.D. degree in Cloud Computing from Noorul Islam Centre for Higher Education, India. She completed her M.E. degree in Software Engineering from Periyar Manniammai College of Engineering, Anna University, India, and her B.Tech degree from Jayamatha Engineering College, Anna University, India. Currently, she is working as an Associate Professor in the Department of Information Technology at Noorul Islam Centre for Higher Education. Her areas of interest include Cloud Computing, Service-Oriented Computing, Parallel Processing, and Distributed Computing.
Email: brindha@niuniv.com

## Appendix I

| S. No. | Abbreviation | Description |
|---|---|---|
| 1 | ACO | Ant Colony Optimization |
| 2 | ACRO | Artificial Chemical Reaction Optimization |
| 3 | AGWO | Advanced Grey Wolf Optimization |
| 4 | AIN | Artificial Immune Network |
| 5 | AIS | Artificial Immune System |
| 6 | APOA | Artificial Plant Optimization Algorithm |
| 7 | AWS | Amazon Web Services |
| 8 | BBO | Biography Based Optimization |
| 9 | BIS | Biological Immune System |
| 10 | CGWO | Clustered Grey Wolf Optimization Algorithm |
| 11 | CLONALG | Clonal Selection Algorithm |
| 12 | CRO | Chemical Reaction Optimization |
| 13 | CSP | Cloud Service Provider |
| 14 | DC | Data Center |
| 15 | DE | Differential Evolution |
| 16 | EA | Evolutionary Algorithm |
| 17 | FIFO | First-In, First-Out |
| 18 | GA | Genetic Algorithm |
| 19 | GCP | Google Cloud Platform |
| 20 | GLS | Genetic Local Search |
| 21 | GWO | Grey Wolf Optimization |
| 22 | IaaS | Infrastructure as a Service |
| 23 | INA | Immune Network Algorithm |
| 24 | IoT | Internet of Things |
| 25 | MA | Memetic Algorithms |
| 26 | MDE | Memetic Differential Evolution |
| 27 | MIPS | Millions of Instructions Per Second |
| 28 | MPSO | Memetic PSO |
| 29 | PaaS | Platform as a Service |
| 30 | PGSA | Plant Growth Simulation Algorithms |
| 31 | PM | Physical Machine |
| 32 | PPA | Plant Propagation Algorithm |
| 33 | PSO | Particle Swarm Optimization |
| 34 | QABSSA | QoS Aware Binary Salp Swarm Algorithm |
| 35 | QoS | Quality of Service |
| 36 | RAM | Random Access Memory |
| 37 | SaaS | Software as a Service |
| 38 | SI | Swarm Intelligence |
| 39 | SLA | Service Level Agreements |
| 40 | SSA | Salp Swarm Algorithm |
| 41 | TSAC | Task Splitting Agglomerative Clustering |
| 42 | vCPU | virtual Central Processing Unit |
| 43 | VM | Virtual Machine |
| 44 | WOA | Whale Optimization Algorithm |