



Adaptive golden eagle optimization based multi-objective scientific workflow scheduling on multi-cloud environment

S. Immaculate Shyla¹ · T. Beula Bell² · C. Jaspin Jeba Sheela³

Received: 3 November 2022 / Revised: 15 July 2023 / Accepted: 1 October 2023 /
Published online: 26 October 2023

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract

An exemplary for emerging knowledges and the capacity to provide reliable cloud services, cloud computing. Giving consumers on-demand access to “unlimited” computer resources is one of the key components of cloud computing. Single cloud-holding resources, however, are typically constrained and might not be able to handle the unexpected spike in user demands. In order to support resource sharing amongst clouds, the multi-cloud concept is thus established. These days, offering resources and administrations across numerous clouds is unquestionably amazing. The goal of conventional research on cloud scheduling is to reduce costs or increase speed. However, the major indicator of QoS and a vital problem is the dependability of work process scheduling. As a result, multi-objective scheduling for a logical work process in a multi-cloud environment is suggested in this research with the goal of controlling the work process while also balancing cost and timeliness while satisfying the criterion of reliability. The adaptive golden eagle optimisation (AGEO) algorithm is created to realise this idea. The solution encoding, fitness analysis, and updating functions are used in the proposed algorithm’s validation. Different workflow models are employed for the experimental study, and performance is assessed using various indicators. The projected approach attained 1920 utilization. Similarly, the PSO and GA achieved 1901 and 1900 utilization.

Keywords Makespan · Task scheduling · Multi-cloud environment · Golden eagle optimization · Normalized cost · Resource utilization

✉ S. Immaculate Shyla
immaculatejudit@gmail.com

T. Beula Bell
Beulamaglin@gmail.com

C. Jaspin Jeba Sheela
jaspinjebasheela@gmail.com

¹ Department of Computer Science, Holy Cross College (Autonomous), Nagercoil, India

² Department of Computer Applications, Nesamony Memorial Christian College, Marthandam, India

³ Department of PG Computer Science, Nesamony Memorial Christian College, Marthandam, India

1 Introduction

The development of cloud computing architecture has accelerated due to the enormous surge in knowledge of information on the internet. The Internet-based delivery of innovation administrations is greatly aided by cloud computing. Customers are given access to the PC framework's resources, such as data storage and recording power, without having to deal directly with dynamic management [1]. Three different kinds of help related to configuration, status, and programming are all available from a cloud. An infrastructure as a service (IaaS) foundation called Primary Administration offers [2] configuration administrations like capacity configuration and computing resources. Platform as a Service (PaaS) level second aid enables users to compile their requests on a specific platform. The third aid provides software as a service (SaaS) [3], which enables users to get programmes directly through the cloud without having to install anything locally. Utilizing advancements in virtualization, cloud providers identify assets for clients as Virtual Machines (VMs) [4].

When several customers are requesting administrative tasks from the cloud, a production fault booking is expected to operate as part of distributed computing's regular operations. Effective allocation booking provides both a constrained chance to deliver the required quality of service (QoS) and an improved opportunity to distribute assets between identified faults [5]. With the defined constraints in the VM area taken into account, the work scheduling problem is resolved and effective results are obtained. As a result, one of the fundamental elements of every cloud architecture is the job scheduling algorithm [6]. To help cloud users, numerous cloud organiser models are employed. The multi-cloud is one that stands out. Multi-cloud is a service provided by two businesses that are dispersed among several distinct cloud providers. The multi-cloud level might be entirely private, entirely open, or both. Different cloud possibilities are used by organisations to distribute recording resources [7] and remove the danger of data disasters and individual duration. In distributed computing, the internal and external resource needs are maintained; for examples, movement speed, accumulation, resource costs, and reaction time may vary for each task. Execution of load adjustments, dependability, and the dynamic transfer of assets to the registration centre are often the main issues [8]. As a result, it is anticipated that the assignment for cloud computing may have been scheduled incorrectly.

Standard and dynamic task scheduling are the two different types of task scheduling systems. For specialised businesses, static scheduling makes use of public data and scheduling firms. This is more than the prior booking system, and it runs businesses when the data is murky [9]. Despite the fact that this task is NP-complete, there have been some calculations to date that have satisfied the constraints for planning error predictions. Several upgrade approaches have been used in job planning, including particle swarm optimization, tabu search, simulated annealing, ant colony optimization, electro search, and genetic algorithms. Given the large record power anticipated to carry out a continuous logical work process [10], it is difficult to develop an efficient computation that takes into account every constraint required to carry out the task scheduling [11].

Section 2 of the added section of the research article provides an overview of work scheduling. Section 3 presents the background investigation of the multi-cloud environment. The proposed strategy is presented in Section 4. In part 5, the article's synopsis is given.

2 Related works

Researchers have developed a number of task-scheduling methods. This section only analyses a few research articles.

The transcendental planning computation aimed at explicit cloud knowledge is evaluated using the Prediction of Tasks Computation Time approach (PTCT). Belal Ali almaytami et al., [12] have presented an intelligent scheduling algorithm employing a Direct Acyclic Graph (DAG). Similar to how the MacBook computation is greatly improved, the suggested calculation also simplifies and minimises the scheming and difficulty of by means of the PCA and ETC networks. In comparison to complex Min-Min, Max-Min, QoS-Guide, and MiM-MaM scheduling computations, reproduction results resolve the unmatched demonstration of computations directed at different conformations in effectiveness, rapidity, and schedule length proportion.

For production cloud management, V.Priya et al., [13] proposed integrated asset booking and load adjustment computations. This method uses the uncertainty of doing so as the foundation for a multi-dimensional resource planning model to provide asset planning capabilities in the cloud architecture. By gradually selecting an appeal from a class using multi-dimensional sequence load optimization computations, it was possible to increase the utilisation of VM through response and acceptable load modification. By reducing abuse and overuse of resources, a load modification intention previously further optimises idle time for each type of request. Reconfigurations were made possible to evaluate the Cloudsim test system's dependability on cloud server farms, and it achieves better results in terms of achievement rates, asset planning efficiency, and reaction time.

Particle swarm optimization (PSO), which was intended to use heuristic computations, has been improved by Seema A. Alsaidy et al., [14]. Calculations for the longest job to fastest processor (LJFP) and minimal completion time (MCT) are utilized to introduce PSO. Projected LJFP-PSO and MCT-PSO computations are presented. The MacBook is rated for managing all power usage measures, randomization, and entire processing time. For assessment reasons, the proposed method is also contrasted with the traditional work scheduling approaches. The LJFP-PSO and MCT-PSO are compared through standard PSO and calculations of a similar nature to determine their reliability and dispersion.

Given that efforts are viewed as premature and free in PSO computation, Fatemeh Ebadifard et al. [15] have developed a error planning. This solution utilised a load-balancing process to produce the fundamental PSO technique. This approach is opposed to the suggested method, Round Robin (RR), the improved PSO job scheduling, and the load-balancing system. The method demonstrates these calculations for the restructuring outcomes with a 22% gain in asset consumption and a 33% decrease in the Macbeth, contrast, and crucial PSO computations. The results show that, compared to the conventional PSO computation, our recommended approach is more efficient with additional jobs and integrates closer to a superior arrangement.

The Energy-Effective Assignment Booking Calculation in the Order Priority Technique by Best-Worst (BWM) and Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) have been introduced by Reihaneh Khorsand et al., [16]. This paper's main objective was to determine the most significant cloud booking configuration. A dynamic collection first distinguishes grading standards. After receiving such advice, a BWM cycle was utilised to lower the critical load aimed at each level, and depending on that, the relevance of the chosen rules changed. These weighted standards were then added by TOPSIS as inputs to assess and gauge each additional option presented. ANOVA and CloudSim Toolkit are used for fact-checking,

and it is predicted that the presentation of the planned and existing computations will include some definitions for these metrics (makepan, energy use, and property value).

The computing time of tasks in the cloud is one of the important considerations. As a platform for dynamic computation, cloud schedulers should be quick and easy to integrate with the actual cloud platform. These scheduling plans ought to be quickly convergent and offer an ideal outcome. Therefore, the goal of this research is to maximise throughput, Average Resource Utilization Ratio (ARUR), and task execution time. A thorough analysis of the relevant literature reveals that the vast majority of task scheduling techniques now in use are tested on relatively small datasets, which is insufficient to demonstrate their scalability. This is due to the fact that scheduling algorithms must be scalable.

3 System model

Models and applications for cloud resources are described in this section. The broad definition and workflow model for the idea of cloud resources and applications are presented in this part. The task processing time is taken into account in order to schedule tasks effectively.

3.1 Cloud resource model

The IaaS platform provides computing power in the form of virtual machines to process large-scale scientific procedures. The processing VM is commonly referred to as an instance. This gives the instances access to numerous QoS factors, including network bandwidth, storage capacity, CPU type, and different prices. Figure 1 shows the prototype cloud resource model.

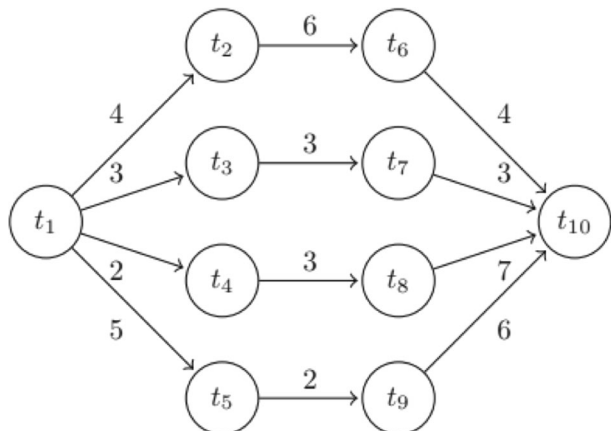
This instance type's description reads,

$$I = \{I_1, I_2, \dots, I_K\} \quad (1)$$

Where K is described as the number of instance kinds. The instance kind I_K is described as [17],

$$\{IC_K, C_K^{BW}, C_K^{IC}, C_K^S, BW_K, S_K\} \quad (2)$$

Fig. 1 Directed acyclic graph



Where, BW_K is represent as the bandwidth, S_K is denoted as the storage capacity and IC_K is described as processing power. C_K^{BW} is represented as the cost of bandwidth, C_K^{IC} is represented as the cost of processing power and C_K^S is denoted as the cost of storage capacity. The task processing time in the instant I_K is denoted as follows,

$$t_{I,K}^{process} = \frac{l(T_I)}{IC_K} \tag{3}$$

Here, (T_I) denotes the period’s length with a million instructions, and $l(T_I)$ denotes the execution time with all of the information received from them. Based on the parent task, $T_j \in Parent(T_I)$, the data transmission time is assessed which is constructed as follows,

$$t_{I,J}^{transmission} = \frac{d_{I,J}}{Min(BW_K, BW_L)} \tag{4}$$

Here, BW_L is described as the bandwidths of the instances to T_j scheduled, BW_K is described as the bandwidth of the instances to T_I scheduled, $d_{I,J}$ is described as the data of the task. At the same time, two tasks are processed after that data transmission time moves to zero.

3.2 Application model

The application model is developed based on the DAG graph which is formulated as follows,

$$g = (t, e) \tag{5}$$

Here, $t = \{t_1, t_2, \dots, t_n\}$ is described as the computational tasks and $e = \{E_{t_i,t_j} | t_i \neq t_j \text{ and } t_i, t_j \in t\}$ is control sets or information dependencies among tasks described as directed edges. The precedence constraint among two tasks is denoted as the edges e_{t_i,t_j} . Normally, the task t_i is a parent of $t_j (t_i \in Parent(t_j))$ and, the child of t_i is denoted as t_j . This restriction stipulates that a task process like t_j is expected individual after the procedure of determining its paternities and full data collected from the parents has been completed. Add two dummy tasks to the beginning and end conditions of the DAG in order to increase the number of t_{exit} and t_{entry} presented in the task operation. Figure 1 illustrates a simple DAG instance. The weight assigned to each edge in the graph is indicated by the estimated data transmission time among the relevant activities. The workflow condition is processed using three virtual machines (V_{m1}, V_{m2} and V_{m3}) in this instance [18].

3.3 Scheduling architecture

Create a multi-objective scheduling algorithm for this study with the goal of enhancing workflow in a multi-cloud environment. Utilising this anticipated technique, the workflow cost and timeline are optimised while taking dependability limits into account. Thus, when network scheduling occurs, three different types of difficulties are taken into account, including the type of task instructions for data transmission, the type of VM picked for the task, and the cloud IaaS platform chosen. The MOS approach, which is based on the adaptive golden eagle optimisation algorithm, is used to resolve these issues. Figure 2 presents the suggested architecture.

The cloud user initially launches a workflow application using any design as its foundation. After that, dependability, cost, and makespan are taken into account when designing the multi-objective scheduling technique. Finally, the workflow tasks are associated with the efficient cloud platforms that fit the necessary VM type. Each cloud service provider includes a task line and a process order for these jobs in their workflow architecture. In this case, a task is being completed while an endless variety of virtual computers continuously work with cloud users and parallel tasks in the queue. The tasks connected to dependence relations should be moving forward in accordance with the requirements. Each cloud in this multi-cloud scenario has unique performance metrics and pricing.

3.3.1 Multi-objective functions

A multi-objective optimization problem has three suitability functions, each of which is designated as $f(x)$. These fitness functions are $f_1(x), f_2(x), \dots, f_k(x)$, and they all need to be optimized simultaneously. Here, x stands for the decision space and X is an element. Pareto dominance is often used to compare the $f(x)$ solution (i.e., $X_1, X_2 \in X$).

$$X_1 > X_2 \iff \forall_i : f_i(X_1) \geq f_i(X_2) \wedge \exists J : f_i(X_1) < f_i(X_2) \tag{6}$$

Here, $(X_1) < (X_2)$ is described as dominates X_2 , $f(x)$ required to be reduced. The outcome is known as Pareto optimum since it does not get better with more solutions. The multi-objective function of this proposed strategy is thought to be connected to cost, time, and resource utilisation. The following is an explanation of these objective functions:

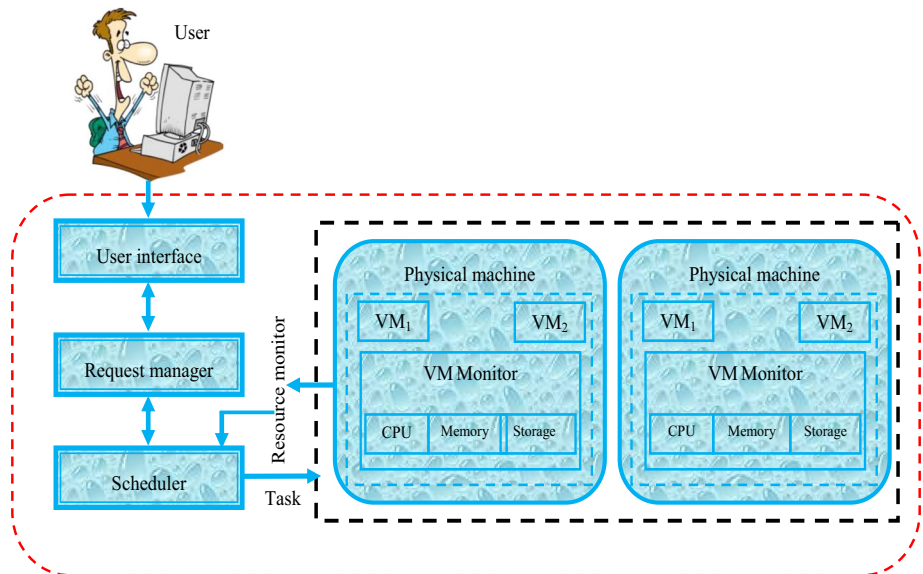


Fig. 2 Proposed architecture

3.4 Resource utilization

The number of commands used to operate the virtual machine up until its completion time is similar to the virtual machine resource utilisation. The VM’s resource use is calculated as follows:

$$u_k = \frac{\sum_{t_i \in A} mi(t_i)}{\sum_{t_i \in A} \left(t_{I,K}^{process} + \max_{t_i \in Parent(t_i)} t_{I,J}^{transmission} \right)} \tag{7}$$

Where, $mi(t_i)$ is denoted as the task size and A is described as the set of tasks. The average utilization is represented as follows,

$$Utilization = \frac{\sum_{Vm \in v} u_k}{|v|} \tag{8}$$

Where, $|v|$ is denoted as the cardinality of it and v is represented as the set of instances.

Cost function

The number of commands used to operate the virtual machine up until its completion time is similar to the virtual machine resource utilisation. The VM’s resource use is calculated as follows:

$$Cost_i^{PC} = t_{I,K}^{process} \times C_K^{IC} \tag{9}$$

Here, C_K^{IC} is described as the processing cost and $t_{I,K}^{process}$ is described as the processing time. The following equation is used to calculate the cost of data transfer between tasks and to their offspring.

$$Cost_i^{BW} = \sum_{t_j \in (t_i, \epsilon Parent(t_j))} t_{I,J}^{transmission} \times C_K^{BW} \tag{10}$$

Here, bandwidth cost is denoted by C_K^{BW} . Consider here the data communication and analysis times among its parents to calculate the cost of storage.

$$Cost_i^S = \left(t_{I,K}^{process} + \max_{t_i \in Parent(t_j)} t_{I,J}^{transmission} \right) \times C_K^S \tag{11}$$

Here, C_K^S is described as the storage cost. Finally, the total cost of handling finished jobs is defined as,

$$Cost = \sum_{t_j \in t} Cost_i^{PC} + Cost_i^S + Cost_i^{BW} \tag{12}$$

Makespan

A workflow’s whole completion time is known as the makespan. The makespan of a parameter is determined using the entire time of the departure job due to the DAG’s construction. Additionally, the highest path of the workflow is used to calculate a workflow’s processing time in a risky manner. Therefore, the makespan is defined as the time it takes

for the last operation of a risky route to complete. To control the task's start time and completion time, two operations are defined. The task's start-up time is calculated as follows based on the completion times of its parents and the time required for data communication between its parents:

$$t^{str}(t_i) = \begin{cases} 0 & t_i = t_{entry} \\ \max_{t_j \in \text{Parent}(t_i)} \{t^{com}(t_j) + t_{ij}^{trn}\} & \text{otherwise} \end{cases} \quad (13)$$

Here, $t^{str}(t_i)$ is described as the starting time, $t^{com}(t_j)$ is described as the complete time computed based on the below formulation,

$$t^{com}(t_j) = t^{str}(t_j) + t_{I,K}^{process} \quad (14)$$

After that, the makespan is computed based on the below formulation.

$$\text{Makespan} = t^{com}(t_{exit}) \quad (15)$$

4 Proposed approach

The proposed approach helps to solve the workflow scheduling problem by taking into account the discrete multi-objective optimization problem that links workflow tasks to necessary instances. The adaptive golden eagle optimization approach was created in order to resolve the multi-objective function and make it possible for an efficient workflow scheduling process in a multi-cloud environment.

4.1 Golden eagle optimization algorithm

The golden eagle, sometimes known as the patchwork chrysaetos, preys on hawk and eagle prey. These eagles are closely related to people. The spiral trajectory used in the golden eagles' hunting and cruising is one of its distinctive features. The prey is present for the most of the spiral trajectory on the eagle's single side. Through this procedure, they are able to observe their hounded prey and surround stones with bushes to determine the best angle of attack. The characteristics of the golden eagle are moving forward in every instance of flight with two forces: a tendency to cruise and a tendency to attack. The golden eagle attacks quickly because it is aware that it only catches little prey. The chasing's primary behaviour.

- It searches for other eagles to gather data about prey.
- They continue to have a predisposition for both attack and cruise for the entirety of the trip.
- Its current high propensity for cruising in the early stages of a smooth transition to hunting to an added propensity for attacking in the later stages.
- The eagle pursues a spiral trajectory designed to detect the traditional attack route.

The golden eagle generates intelligent balance, assault, and cruising between the two stages, such as exploration and exploitation. The algorithm's accurate modelling is displayed in the manner as follows. The measured construction aims to mimic the variations in golden eagles' hunting behaviour. The presentation of the formulation's spiral motion is followed by the analysis of its division into assault and cruise vectors that emphasise exploration and exploitation.

4.1.1 Golden eagle spiral motion

This algorithm is based on the spiral motion of golden eagles. Every golden eagle keeps a record of its most favourable lodging location. The eagle is motivated to pursue the best prey and go on the attack. Every time, the golden eagle circled the best position that was too far away for it to pick the circle connected with a memory comparable to its own and randomly selected the prey of the other golden eagles. As a result, the golden eagle's original population is described as follows:

$$F \in \{1, 2, \dots, \text{population size}\} \quad (16)$$

4.1.2 Selection of prey

Each golden eagle has the option to select a prey to use for its attack and travel functions in every iteration. The prey is created as the best solution in the golden eagle algorithm so far taking into account a flock of golden eagles. Each golden eagle is capable of holding the ideal answer. Based on the selected prey, each golden eagle's cruise and attack vectors are calculated. The memory is then modified so that the new location now comes before the old location. The prey-chosen strategy in the golden eagle optimisation is crucial to getting effective results. Every lone golden eagle selected the prey from its storage as part of a broad selection process.

4.1.3 Attack (exploitation)

The attack is planned with a vector that starts with the eagle's current position in the attack strategy. The assault vector offers the population of golden eagles the most popular sites, which is an advantage of the exploiting process in golden eagle optimization. The attack vector in the golden eagle optimization is calculated using the equation below,

$$\vec{a}_I = \vec{x}_F^* - \vec{x}_I \quad (17)$$

Here, the eagle's current position is denoted by \vec{x}_I , its preferred prey location is denoted by \vec{x}_F^* , and the eagle's attack vector is denoted by \vec{a}_I . An benefit of the exploitation stage in golden eagle optimisation is that the assault vector often controls the golden eagle population to the best-identified places.

4.1.4 Cruise exploration phase

In connection with the attack vector, the cruise vector is calculated. The cruise vector, which is perpendicular to the circle of the attack vector, is referred to as a tangent vector. The golden eagle's cruise can also be at a linear speed that is connected to its prey. In order to compute the

cruise vector, first solve the equation for the tangent hyperplane, which is an N-dimensional vector that is expressed in confidence to the circle's tangent hyperplane. The hyperplane equation is stated in its scalar form as follows:

$$h_1x_1 + h_2x_2 + \dots + h_nx_n = d \Rightarrow \sum_{j=1}^N h_jx_j = d \quad (18)$$

Here, $X = [x_1, x_2, \dots, x_n]$ is described as a variables vector. $h = [h_1, h_2, \dots, h_n]$. The hyperplane random point is described as, $p = [p_1, p_2, \dots, p_n]$ and $d = \overline{H} \cdot \overline{P} = \sum_{j=1}^N h_j p_j$. The eagle's position is thought to be a random spot on the hyperplane, and its assault path is thought to be typical for the hyperplane. This is how its cruise vector is displayed,

$$\sum_{j=1}^N A_j X_j = \sum_{j=1}^N A_j^t X_j^* \quad (19)$$

Here, $X^* = [X_1^*, X_2^*, \dots, X_N^*]$ is described as the location of the chosen prey, $\overline{X} = [X_1, X_2, \dots, X_N]$ is described as the design or decision variables vector and $\overline{A}_I = [A_1, A_2, \dots, A_N]$ is described as the attack vector.

4.1.5 Moving to new positions

The displacement of the golden eagle is made up of vectors and an attack. The following is how the golden eagle step vector is presented:

$$\Delta X_I = \overline{R}_1 \rho_a \frac{\overline{A}_I}{\|\overline{A}_I\|} + \overline{R}_2 \rho_c \frac{\overline{C}_I}{\|\overline{C}_I\|} \quad (20)$$

Here, $\|\overline{A}_I\|$ is described as the Euclidean norm of the attack vectors, $\|\overline{C}_I\|$ is described as the Euclidean norm of the cruise vectors, \overline{R}_1 and \overline{R}_2 is described as the random vectors which are presented in the period $[0,1]$, ρ_c is described as the cruise coefficient, ρ_a is described as the attack coefficient. The Euclidean norm of the cruise and attack vectors are formulated as follows,

$$\|\overline{A}_I\| = \sqrt{\sum_{j=1}^N A_j^2} \quad (21)$$

$$\|\overline{C}_I\| = \sqrt{\sum_{j=1}^N C_j^2} \quad (22)$$

The golden eagle position in iteration $t + 1$ is computed based on the addition of the step vector.

$$X^{t+1} = X^t + \Delta x^t \quad (23)$$

4.1.6 Movement from exploration to exploitation

This is the first round of the hunting battle, and it shows a strong propensity for attack in the later rounds due to increased exploration in earlier rounds and high exploitation in later rounds. As shown below [19], intermediate parameters are calculated taking into account transition.

$$\begin{cases} p_a = p_a^0 + \frac{t}{T} |p_a^T - p_a^0| \\ p_c = p_c^0 + \frac{t}{T} |p_c^T - p_c^0| \end{cases} \quad (24)$$

Here, p_c^T and p_c^0 is described as the final and initial parameters of propensity to cruise, p_a^0 and p_a^T is described as the initial and final parameters of propensity to attack, T is defined as maximum iterations and t is defined as the present iteration.

4.2 Levy flight distribution

Levy flight is used to calculate jump size and to update the position of the golden eagle, which employs the levy probability distribution function, which is a power law operation [20]. The following is the formulation of the levy distribution function:

$$l(s, \gamma, \mu) = \begin{cases} \sqrt{\frac{\gamma}{2\pi}} \exp\left[-\frac{\gamma}{2(S-\mu)}\right] \frac{1}{(S-\mu)^{3/2}} & \text{if } 0 < \mu < \infty \\ 0 & \text{if } S \leq 0 \end{cases} \quad (25)$$

Here, S is referred to as the scale parameter, and, is referred to as the position parameter, all of which are controlled by the distribution function's distribution function's scale. The global and local search is caused to be contained by this levy distribution function. Levy flight distribution is substituted for linear decline as the parameter A . The following is how the levy flight A variable is presented:

$$A = \text{levy}(s) * R_1 \quad (26)$$

Here, R_1 is described as the random vector, s is described as the position of a golden eagle. Levy flight-based A manages the local search abilities and global search characteristics.

4.3 Adaptive golden search optimization algorithm

The updating procedure of the suggested method, the golden search optimisation algorithm, is made possible by the levy flight distribution. Eight different types of eagles are shown here for merging different allocation and prioritisation approaches. This population is creating an order vector for the relationship between parents and kids in a workflow and assigns each job an index between 1 and N , where N is the total number of tasks in the workflow. The order in which tasks are running on the VM is also shown as part of the order vector. The vector parameter is established during the beginning population development and remains constant up until the algorithm's conclusion for each golden eagle. These tasks are directed at order vector-related processing. The proposed method's solution, known as the golden eagle position vector, is a sequence that takes into account the

associated tasks and assigns the management of workflow activities to VM. The index of the task is provided to the position vector dimension, which is related to the parameter that calculates the index of the virtual machine that the task is planned on.

4.3.1 Objective functions

The multi-objective problem in the proposed method has three competing goals, including cost, makespan, and resource utilisation. The following formulation is used to create it:

$$OF = \text{Min } f = \left(\frac{1}{\text{utilization}}, \text{cost}, \text{makespan} \right)^t \quad (27)$$

4.3.2 Fitness function

The appropriateness of each golden eagle is determined using the raw operation. A golden eagle's basic functioning is calculated taking into account the strength of its superiors. The definitions and functions indicate that the raw parameter is based on Pareto dominance. The move to Pareto supremacy to dangerously control the workflow is part of the intended method. The following formulation describes the condition under which the two eagles I,J dominate J:

$$\text{Cost}(i) \leq \text{Cost}(j) \wedge \quad (28)$$

$$\text{Utilization}(i) \geq \text{Utilization}(j) \wedge \quad (29)$$

$$\text{Makespan}(i) < \text{Makespan}(j) \wedge \quad (30)$$

This definition also alters the constituents of the dominated set. Although the definitions of the eagles' raw and strength parameters are different, they are the same as those described in the golden eagle optimisation algorithm. The duties are arranged effectively with the aid of the predicted strategy.

5 Performance evaluation

This section uses the task scheduling procedure to validate and defend the planned technique. The proposed method is put into practise using cloudSim devices and Java (JDK 1.6), and its performance is assessed. On a 2 GHz dual-core PC running a 64-bit version of Windows 2007, the anticipated system experiments were tested.

5.1 Experimental outcome

Virtual machine scheduling with the aid of the AGEIO technique is the fundamental information of the proposed technique. Give M resources and N tasks their initial assignments. The scheduling process can now take into account cost, resource use, and makespan thanks to the AGEIO approach (Table 1).

Table 1 Implementation parameter

Parameter Description	Workflow model 3 Range	Workflow model 2 Range	Workflow model 1 Range
Time	1000 ms – 100,000 ms	1000 ms – 100,000 ms	1000 ms – 100,000 ms
Memory	100 MB-1 GB	100 MB-1 GB	100 MB-1 GB
CPU	126	294	589
Tasks	100	200	400
Virtual Machine	126	294	589
Physical Machine	50	100	200

Three issues are taken into account in order to validate the proposed technique. The problem considers the high-range variables. (1) In the issue, middle-range variables are considered (2) In the issue, low-range parameters are employed. (3) The challenge takes into consideration 589 virtual computers and 200 real machines. (1) The problem measures 294 virtual computers and 100 real machines. (2) In problem 3, 50 actual machines and 126 virtual machines are considered. In problem 3, 400 tasks are taken into account. In problem 2, 200 tasks are taken into account. and in problem 1, 100 tasks are taken into account. the overall number of CPUs utilized, which corresponds to the overall number of virtual machines utilized in each issue. The resource is then used for a time of 1000 to 100,000 milliseconds, and the memory is used for a time of 100 MB to 1GB.

5.1.1 Workflow model 1: Performance validation with 589 virtual machines and 200 physical machines (problem instance 1)

Utilising 589 virtual computers and 200 actual machines, the performance evaluation of problem issue 1 is assessed. 400 tasks are then assigned to be scheduled in virtual machines after that. In this evaluation, the anticipated result is put up against a traditional method like GWO or PSO. It is possible to schedule tasks with the aid of the projected method. In the context of a multi-cloud environment, three fitness functions—resource utilisation, makespan, and cost—are taken into account (Tables 2, 3 and 4).

The 200 physical machines and 589 VMs used to determine the normalised cost are used to analyse the performance validation of workflow model 1. Following that, 400 jobs were assigned to virtual machines for scheduling. In terms of normalised cost, the proposed method is compared to the traditional methods. In terms of normalised cost, the projected strategy produced the best results. The anticipated technique reached 70 normalised costs by iteration five. The PSO and GA also reached 95 and 115 normalised

Table 2 Normalized cost parameters

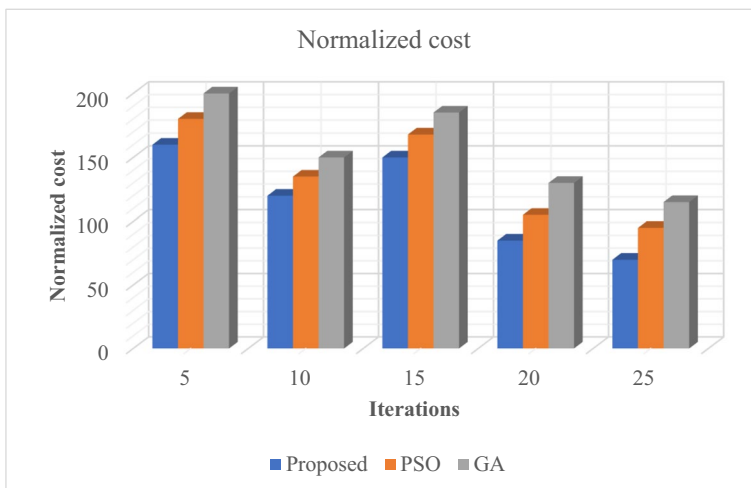
Iterations	Proposed	PSO	GA
5	160	180	200
10	120	135	150
15	150	168	185
20	85	105	130
25	70	95	115

Table 3 Makespan parameters

Iterations	Proposed	PSO	GA
5	200	225	250
10	175	185	200
15	160	175	195
20	165	185	205
25	150	165	195

Table 4 Utilization parameters

Iterations	Proposed	PSO	GA
5	1950	1900	1895
10	1925	1895	1800
15	1910	1850	1825
20	1905	1825	1810
25	1900	1800	1795

**Fig. 3** Normalized cost for workflow model 1

costs, respectively. Figure 3 displays the normalized cost performance of model 1. Utilizing the 200 physical machines and 589 virtual machines shown in Fig. 4 and the performance validation of workflow model 1 is computed in terms of make span. Following that, 400 jobs were assigned to virtual machines for scheduling. In terms of make span, the planned methodology is contrasted with the traditional methods. In terms of normalised cost, the projected strategy produced the best results. The anticipated approach is reached with 150 makes pans in iteration 5. The PSO and GA also attained 165 and 195 make span. Utilising the 200 physical machines and 589 virtual machines shown

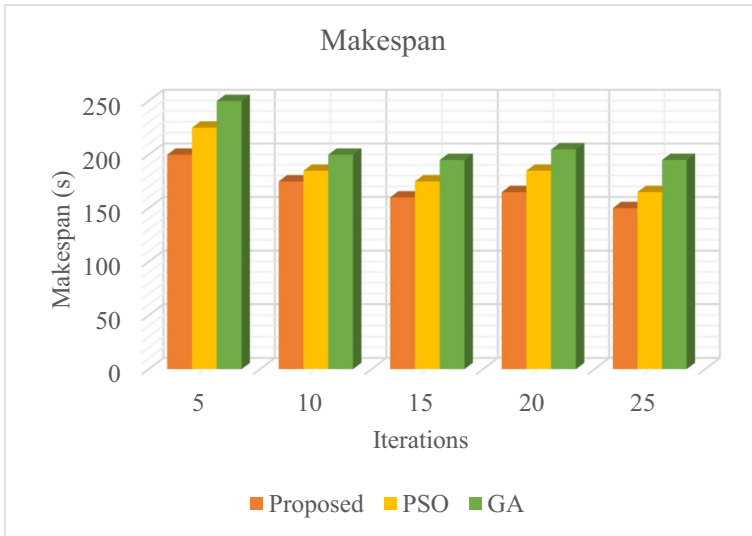


Fig. 4 Makespan for workflow model 1

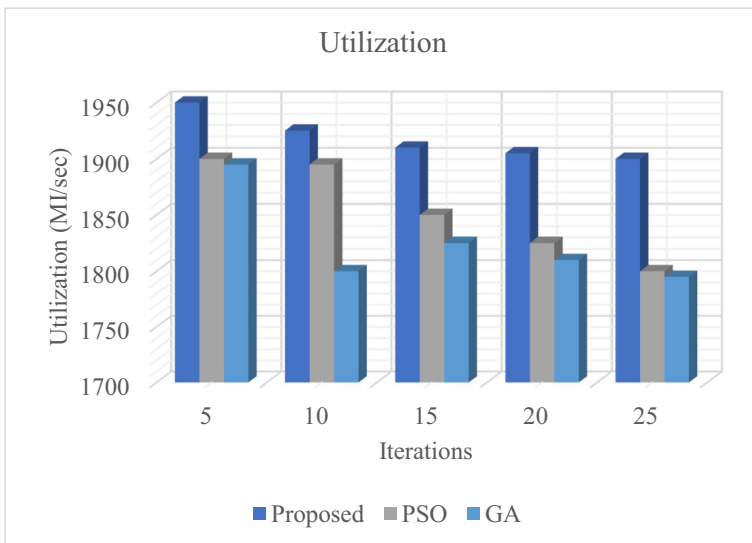


Fig. 5 Utilization of workflow model 1

in Fig. 5, the performance analysis of workflow model 1 is computed in terms of utilisation. Following that, 400 jobs were assigned to virtual machines for scheduling. In terms of utilisation, the proposed strategy is contrasted with the traditional methods. Utilisation-wise, the predicted method produced the best results. The planned approach reached 1900 utilisation at iteration 5. The PSO and GA also attained 1800 and 1795 utilisation, respectively.

5.1.2 Workflow model 2: performance validation with 294Virtual Machines and 100Physical Machines (Problem instance 2)

Using 294 virtual machines and 100 actual machines, the performance evaluation of problem issue 2 is conducted. 200 tasks are then assigned to be scheduled in virtual machines after that. In this evaluation, the anticipated result is put up against a traditional method like GWO or PSO. It is possible to schedule tasks with the aid of the projected method. In the context of a multi-cloud environment, three fitness functions—resource utilisation, makespan, and cost—are taken into account (Table 5, 6 and 7).

The performance validation of workflow model 2 is assessed in terms of normalized cost, which is computed by employing the 100 actual computers and 294 VMs. 200 jobs were then assigned to be scheduled in VMs. In terms of normalised cost, the proposed method is compared to the traditional methods. In terms of normalised cost, the projected strategy produced the best results. The estimated approach reached 63 normalised costs during iteration 5. The PSO and GA are also reached at 68 and 80 normalised prices, respectively. Figure 6 displays the normalised cost performance of model 2. The 100 physical computers and 294 VMs that are used to compute the make span analysis for the performance validation of workflow model 2 in the Fig. 7. 200 jobs were then assigned to be scheduled in VMs. In terms of makespan, the predicted strategy is compared to the traditional methods. The projected strategy produced the best results in terms of timeliness. The anticipated approach is achieved with 50 makespans during iteration 5. The PSO and

Table 5 Normalized cost parameters

Iterations	Proposed	PSO	GA
5	80	88	102
10	73	76	78
15	65	68	79
20	60	65	75
25	63	68	80

Table 6 Makespan parameters

Iterations	Proposed	PSO	GA
5	100	125	135
10	80	95	105
15	60	68	75
20	55	58	68
25	50	55	65

Table 7 Utilization parameters

Iterations	Proposed	PSO	GA
5	1995	1985	1965
10	1978	1965	1950
15	1965	1955	1932
20	1935	1920	1900
25	1932	1910	1898

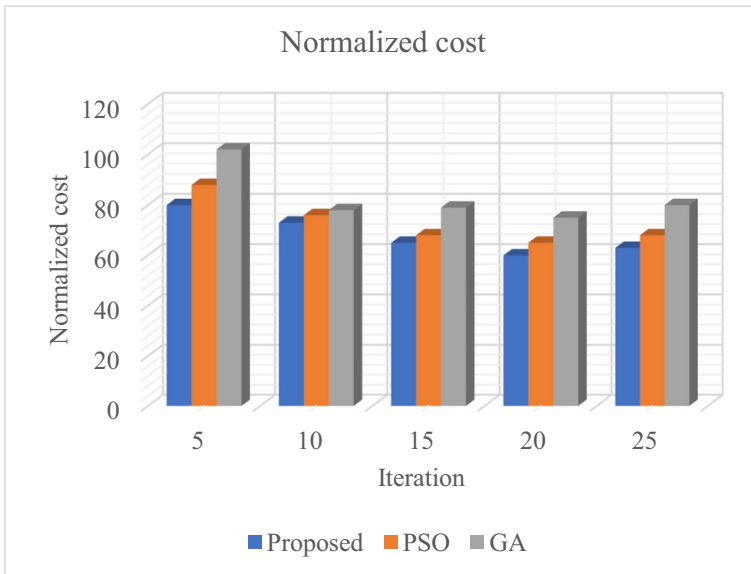


Fig. 6 Normalized cost for workflow model 2

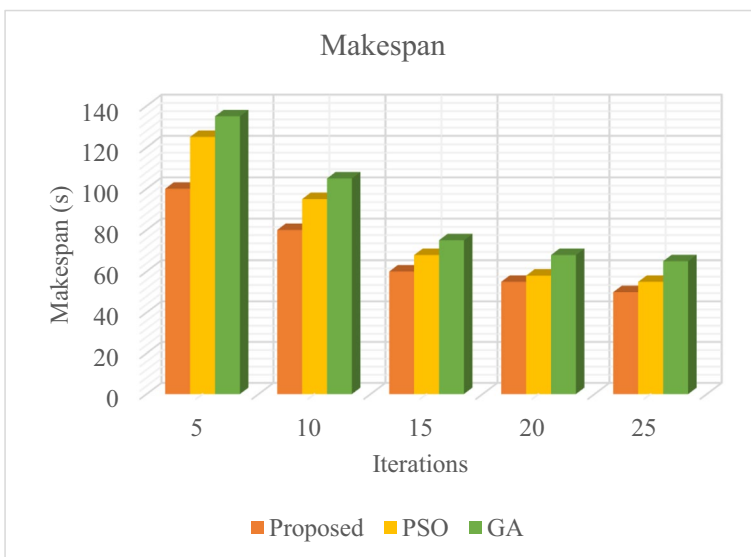


Fig. 7 Makespan for workflow model 2

GA have accomplished 55 and 65 makespan, respectively. The utilization of the 100 physical computers and 294 virtual machines shown in Fig. 8 is computed for the performance validation of workflow model 2 in terms of utilisation. 200 jobs were then assigned to be scheduled in virtual machines. In terms of utilisation, the proposed strategy is contrasted with the traditional methods. Utilization-wise, the predicted method produced the best

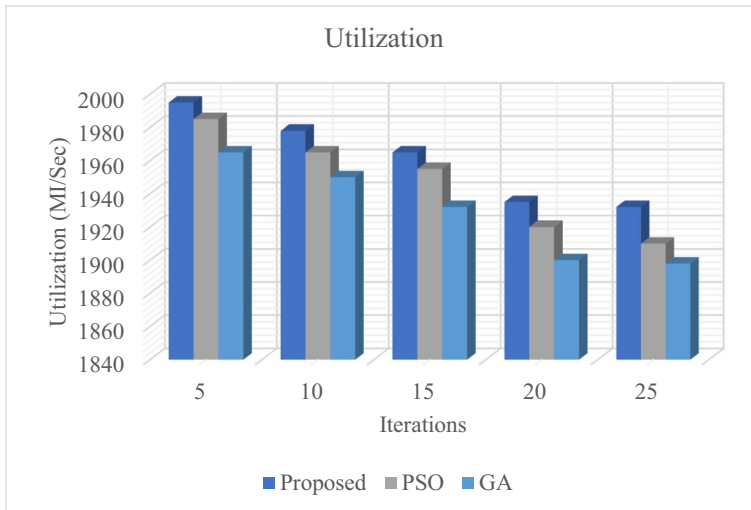


Fig. 8 Utilization of workflow model 2

Table 8 Normalized cost parameters

Iterations	Proposed	PSO	GA
5	40	48	55
10	35	38	42
15	32	39	46
20	28	35	40
25	20	28	35

results. The estimated approach reached 1932 utilisation during iteration 5. The PSO and GA also met their respective utilisation goals of 1910 and 1898.

5.1.3 Workflow model 3: performance analysis with 126 virtual machines and 126 physical machines (Problem instance 3)

126 virtual machines and 50 actual machines are used to analyse the performance of problem issue 3. 100 jobs are then assigned to be scheduled in virtual machines after that. In this evaluation, the anticipated result is put up against a traditional method like GWO or

Table 9 Makespan parameters

Iterations	Proposed	PSO	GA
5	50	55	65
10	35	48	55
15	30	45	55
20	25	35	40
25	15	28	35

Table 10 Utilization parameters

Iterations	Proposed	PSO	GA
5	1975	1854	1845
10	1950	1825	1820
15	1925	1900	1895
20	1910	1925	1910
25	1920	1901	1900

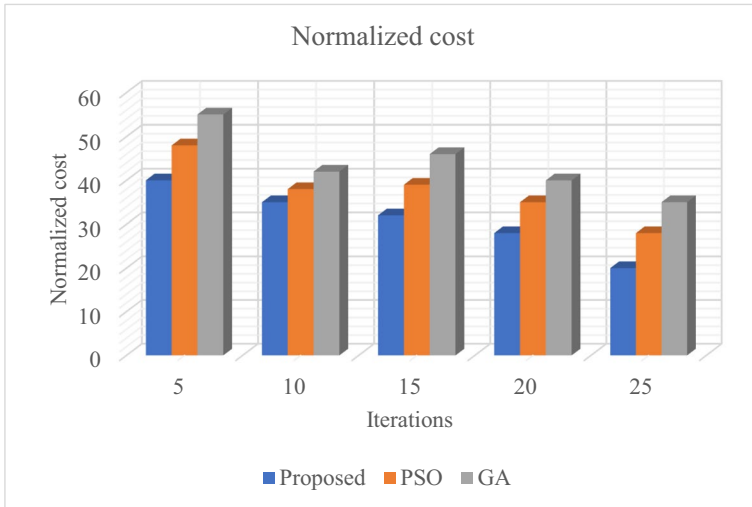


Fig. 9 Normalized cost for workflow model 3

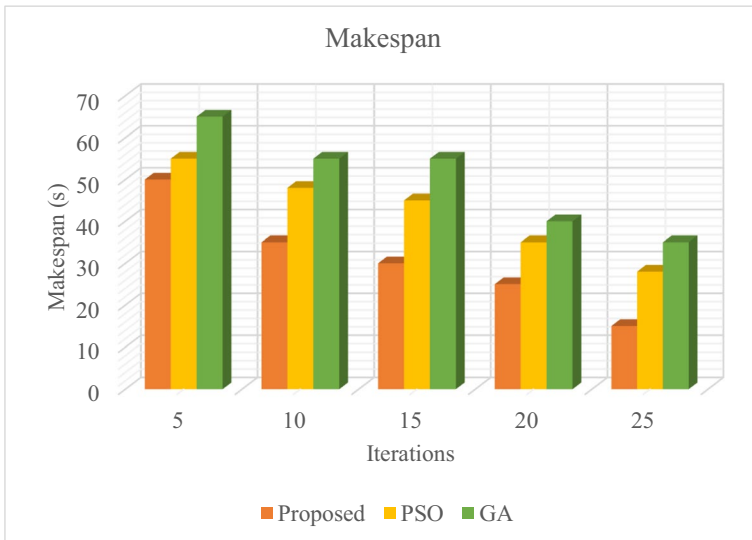


Fig. 10 Makespan for workflow model 3

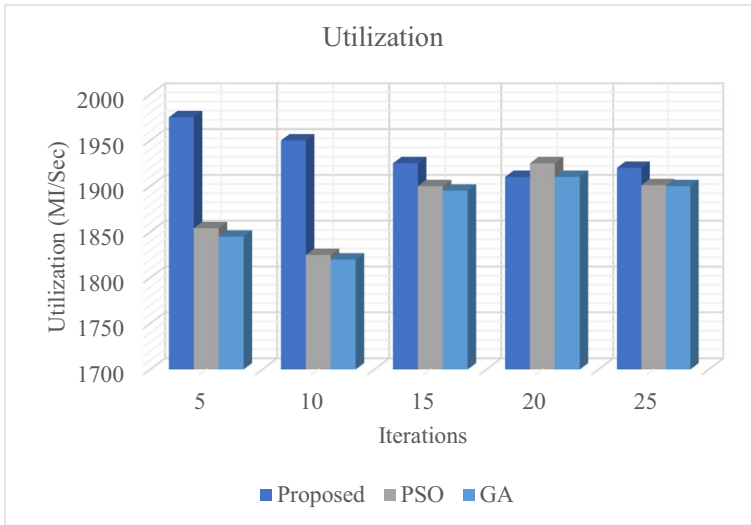


Fig. 11 Utilization of workflow model 3

PSO. It is possible to schedule tasks with the aid of the projected method. In the context of a multi-cloud environment, three fitness functions—resource utilisation, makespan, and cost—are taken into account (Tables 8, 9 and 10).

The normalised cost estimated by using the 50 physical computers and 126 virtual machines is used to analyse the performance validation of workflow model 3. Following that, 100 jobs were assigned to virtual machines for scheduling. In terms of normalised cost, the proposed method is compared to the traditional methods. In terms of normalised cost, the projected strategy produced the best results. The anticipated technique reached 20

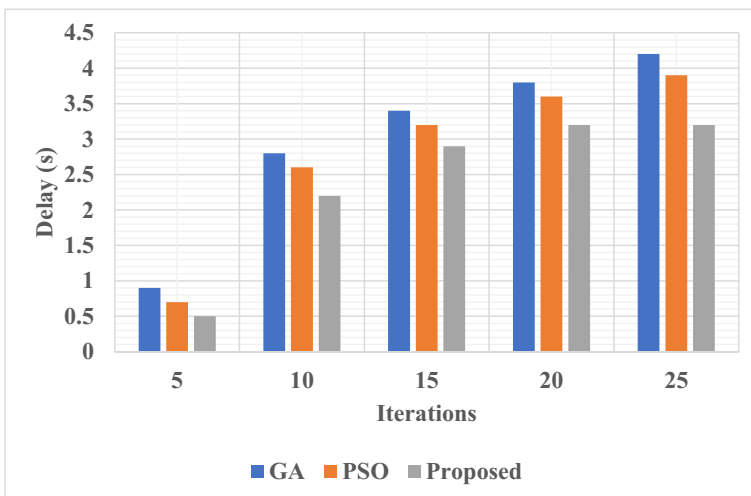


Fig. 12 Delay

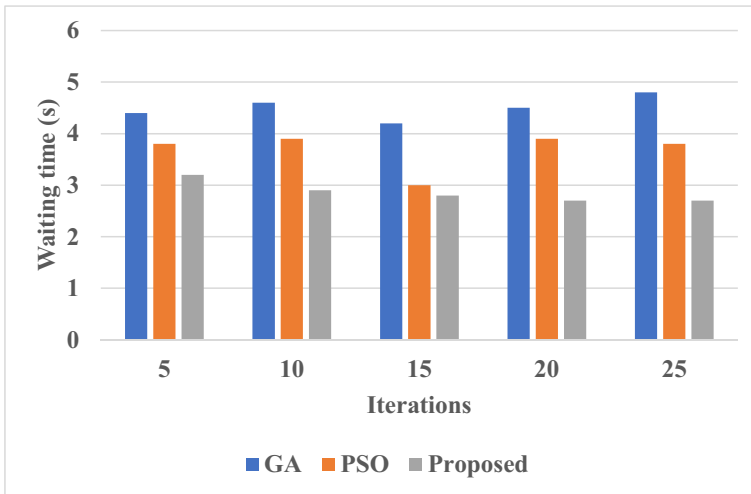


Fig. 13 Waiting time

normalised costs by iteration five. The PSO and GA also achieved 28 and 35 normalised costs, respectively. Figure 9 provides a graphic representation of the model 3’s normalised cost. Utilising the 50 real machines and 126 virtual machines that make up the workflow model 3’s performance validation, make span is computed to analyse the performance of the model 3 and given in Fig. 10. Following that, 100 jobs were assigned to virtual machines for scheduling. In terms of makespan, the predicted strategy is compared to the traditional methods. The projected strategy produced the best results in terms of timeliness. The anticipated approach is reached with 15 makespans during iteration 5. The PSO and GA have also accomplished 28 and 35 makespan, respectively. The 50 physical computers

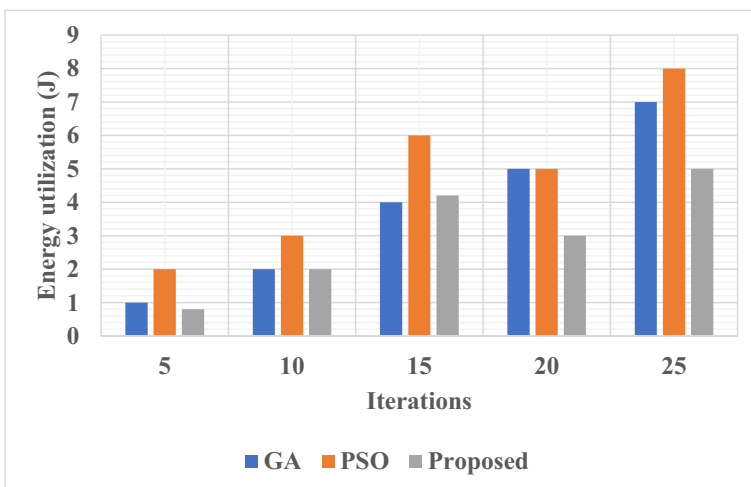


Fig. 14 Energy utilization

Table 11 Comparison analysis

S.No	Author	Method	Delay
1	Belal Ali al-maytami et al. [12]	intelligent scheduling algorithm	2
2	V.Priya et al. [13]	load adjustment calculations	5.5
3	Seema A. Alsaidy et al. [14]	particle swarm optimization (PSO)	4.8
4	Fatemeh Ebadifard et al. [15]	standard error planning technique given particle swarm optimization (PSO)	3.5
5	Reihaneh Khorsand et al. [16]	Energy-Effective Assignment Booking Calculation in Order Priority Technique by best-worst (BWM) and Technique for Order Preference by Similarity to Ideal Solution (TOPSIS)	3
6	-	Proposed method	0.8

and 126 virtual machines shown in Fig. 11 are used to compute the utilisation analysis of the performance validation of workflow model 3. Following that, 100 jobs were assigned to virtual machines for scheduling. In terms of utilisation, the predicted strategy is compared to the traditional methods. Utilization-wise, the predicted method produced the best results. The planned approach reached 1920 utilisation on iteration 5. The PSO and GA also met their respective utilisation goals of 1901 and 1900. The delay measure of the proposed model is illustrated in Fig. 12. Waiting time and energy utilization is analyzed and given in Figs. 13 and 14. From the validation, the proposed technique has been obtained best outcomes (Table 11).

6 Conclusion

This study proposes multi-objective scheduling for a logical work process in a multi-cloud environment, with the goal of controlling the work process while at the same time controlling cost and makespan and satisfying the dependability requirement. The AGE0 algorithm has been developed in order to realise this idea. The suggested approach uses updating, fitness calculation, and solution encoding functions. Different workflow models have been utilised for experimental analysis. The effectiveness of the suggested approach has been assessed using a variety of indicators. Three issues have been taken into account in order to validate the proposed technique. The high-range parameters are used in problem (1) Middle-range parameters are used in problem (2) Low-range parameters have been taken for problem (3) 200 physical machines and 589 virtual machines have both been taken into account in problem 1. Problem 2 takes into account 294 virtual machines and 100 physical machines. 50 real machines and 126 virtual machines are taken into account in problem 3. In problem 3, 400 tasks are taken into account. In problem 2, 200 tasks are taken into consideration, whereas in problem 1, 100 tasks are. The predicted technique has produced effective results in terms of utilisation, normalisation cost, and makespan, according to the analysis.

Authors' contributions The author read and approved the final manuscript.

Funding The authors declare that they do not have competing interests and funding.

Data availability Data sharing is not applicable to this article.

Declarations

Conflict of interest The corresponding author states that there is no conflict of interest.

References

1. Awad AI, El-Hefnawy NA, Abdel_kader HM (2015) Enhanced particle swarm optimization for task scheduling in cloud computing environments. *Procedia Comput Sci* 65:920–929
2. Abd Elaziz M, ShengwuXiong KPN, Jayasena, Li L (2019) Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution. *Knowl Based Syst* 169:39–52
3. Tian W, Xu M, Chen A, Li G, Wang X (2015) Open-source simulators for cloud computing: comparative study and challenging issues. *Simul Model Pract Theory* 58:239–254

4. Chaudhary D, Kumar B (2019) Cost optimized hybrid genetic-gravitational search algorithm for load scheduling in cloud computing. *Appl Soft Comput* 83:105627
5. Ismayilov G (2020) Neural network based multi-objective evolutionary algorithm for dynamic workflow scheduling in cloud computing. *Futur Gener Comput Syst* 102:307–322
6. Nayak SC, Tripathy C (2018) Deadline sensitive lease scheduling in cloud computing environment using AHP. *J King Saud Univ-Computer Inform Sci* 30(2):152–163
7. Babu LDD, Venkata Krishna P (2013) Honey bee behavior inspired load balancing of tasks in cloud computing environments. *Appl Soft Comput* 13(5):2292–2303
8. Abujassar RS, Jazzar M (2017) Enhancing the cloud computing performance by labeling the free node services as ready-to-execute tasks. *J Electr Comput Eng* 2017:1–9
9. Sanaj MS, Joe Prathap PM (2020) Nature inspired chaotic squirrel search algorithm (CSSA) for multi objective task scheduling in an IAAS cloud computing atmosphere. *Eng Sci Technol Int J* 23(4):891–902
10. Abujassar RS (2016) Mitigation fault of node mobility for the MANET networks by constructing a backup path with loop free: enhance the recovery mechanism for pro-active MANET protocol. *Wireless Netw* 22(1):119–133
11. Jena RK (2015) Multi objective task scheduling in cloud environment using nested PSO framework. *Procedia Comput Sci* 57:1219–1227
12. Al-Maytami BA, Fan P, Hussain A, Baker T, Liatsis P (2019) A task scheduling algorithm with improved makespan based on prediction of tasks computation time algorithm for cloud computing. *IEEE Access* 7:160916–160926
13. Priya V, Sathiya Kumar C, Kannan R (2019) Resource scheduling algorithm with load balancing for cloud service provisioning. *Appl Soft Comput* 76:416–424
14. Seema A, Alsaidy AD, Abbood Mouayad A, Sahib (2020) Heuristic initialization of PSO task scheduling algorithm in cloud computing. *J King Saud Univ-Computer Inform Sci* 4(6):–13
15. Ebadifard F, SeyedMortezaBabamir (2018) A PSO-based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment. *Concurr Comput: Pract Exp* 30(12):e4368
16. Khorsand R, Ramezanpour M (2020) An energy-efficient task-scheduling algorithm based on a multi-criteria decision-making method in cloud computing. *Int J Commun Syst* 33(9):e4379
17. Doostali S, Babamir SM, Eini M (2021) CP-PGWO: multi-objective workflow scheduling for cloud computing using critical path. *Cluster Comput* 24(4):3607–3627
18. Haiyang H, Li Z, Hu H, Chen J, Ge J, Li C, Chang V (2018) Multi-objective scheduling for scientific workflow in multicloud environment. *J Netw Comput Appl* 114:108–122
19. Joshi SS (2021) A Novel Golden Eagle Optimizer Based Trusted Ad Hoc On-Demand Distance Vector (GEO-TAODV) Routing Protocol. *Int J Comput Networks Appl* 8(5):538–548
20. Mokeddem D (2021) Parameter extraction of solar photovoltaic models using enhanced levy flight based grasshopper optimization algorithm. *J Electr Eng Technol* 16(1):171–179
21. Duan H, Chen C, Min G (2017) Energy-aware scheduling of virtual machines in heterogeneous cloud computing systems. *Futur Gener Comput Syst* 74:142–150
22. Mohammadi-Balani A, DehghanNayeri M, Azar A, Taghizadeh-Yazdi M (2021) Golden eagle optimizer: a nature-inspired metaheuristic algorithm. *Comput Ind Eng* 152:107050

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.