Received: April 23, 2024. Revised: June 5, 2024.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Cosine Similarity Based Golden Jackal Optimization for Efficient Task Scheduling in Cloud Computing

Mohammed Ziaur Rahman[1]* **Anandaraj Shanthi Pichandi[1]**

[1]*School of Computer Science and Engineering & Information Science, Presidency University, Bangalore, India*
* Corresponding author's Email: ziyarmn@gmail.com

Abstract: Task scheduling in a process where the cloud has attained substantial attention because of an enhanced demand for computing resources and services. Various load balancing approaches are developed for allocating the tasks and Virtual Machines (VMs) according to the task's priority and execution. However, there are some problems faced in the identification of the best tasks for allocation and to classify the relevant tasks based on the load. In this research, Fuzzy Logic (FL) and Cosine Similarity based Golden Jackal Optimization (CSGJO) algorithm is proposed for solving the problem of load balancing and task scheduling in CC. The FL optimally distributes the tasks based on their fuzzy rules, whereas the CSGJO improves fuzzy rule sets and identifies the optimal resource allocation, minimizing the response time even when handling large number of tasks. The Longest Job to Fastest Processor (LJFP) and Minimum Completion Time (MCT) approaches are the heuristic approaches used to initialize CSGJO. The multi-objective functions of Makespan, Degree of Imbalance (DOI), Execution Time, Energy Consumption and Resource Utilization for validating the proposed method's effectiveness. The CSGJO consumes a minimum makespan of 1.9s, as opposed to Particle Swarm Optimization (PSO) and Enhanced Sunflower Optimization (ESFO).

Keywords: Cloud computing, Cosine similarity based golden jackal optimization, Fuzzy logic, Load balancing, Task scheduling, Virtual machines.

1. Introduction

In this advanced world, Cloud Computing (CC) plays a significant role in various applications because of its wide usage and the fast development of internet technologies [1]. CC is an integration of distributed as well as parallel computing which provides resources like software, hardware and files [2, 3]. The CC processes various services based on the service and working platforms which are required by the users [4]. The cloud involves three types of services: Infrastructure as a Service (IaaS), offers services like storage and computational resources. Platform as a Service (PaaS), allows users to develop their applications on cloud platform, and Software as a Service (SaaS) that permits users to utilize software directly from a cloud [5, 6]. The cloud providers accomplish virtualization technology and provides the computational resources in a procedure of Virtual Machines (VM). The scheduling approaches play an important role in optimizing the performance through effectively assigning geographically distributed resources to meet the requirements of end-users. When a number of users request tasks in a cloud, an effective task scheduling is needed to improve system performance of CC [7, 8].

Virtualization is crucial for cloud computing (CC), but there are certain issues that prevent it from meeting user requests, such as improper scheduling or assigning too many tasks to virtual machines.

The aim of effective task scheduling is mapping tasks to the optimal resources, guaranteeing task execution, alongside meeting the Quality of Service (QoS) [9]. Although, virtualization plays an important role in CC, but there are particular problems which prevents it from user request like inappropriate scheduling or offering a greater number of the tasks to VM [10]. To address this problem, scheduling and load balancing among the tasks must be calculated to optimize resource utilization in CC

[11, 12]. The task scheduling is a Non-deterministic Polynomial time (NP)-hard problem, which involves determining the order in which jobs must be processed to minimize execution time and maximize the resource utilization in CC. The meta-heuristic algorithms are introduced by various researchers to solve the NP-hard problem in CC [13, 14]. The meta-heuristics are isolated to determine the optimal solutions; however, the near-optimal solutions are distributed with less complication [15]. The various meta-heuristic approaches attain superior results in task scheduling, but face lag from local optimal problems, as their performance is distant from the model state [16]. The Machine Learning (ML) approaches are used to enhance the performance of task determination according to the search process of meta-heuristic approach, and it supports to select the effective attributes by examining test sets [17, 18]. There are some problems faced in the identification of priority tasks for allocation as the pattern sequence are generally utilized to classify the relevant tasks based on the load. In this research, FL and CSGJO algorithm is employed for solving the issue of load balancing as well as task scheduling in CC. The CSGJO approach continuously improves the scheduling process based on the priority of the tasks. While FL adjusts the task priorities, it leads to enhancing the system performance. The primary contributions of this research are listed below:

- The FL approach is utilized for solving the problem of load balancing in CC. The FL has optimally distributed the tasks based on the fuzzy rules, so it helps to reduce the makespan and execution time.
- The CSGJO based LJFP and MCT is proposed for efficient task scheduling in the cloud according to the VM. The CSGJO improves fuzzy rule sets through refining the rule structure and parameter tuning based on their importance or reliability. Then, it identifies the optimal resource allocation to minimize the response time even as handling a large number of tasks.
- The CSGJO considers the multi-objective functions as Makespan, Degree of Imbalance (DOI), Execution Time, Energy Consumption and Resource Utilization for validating the effectiveness of the model.

This research paper is organized as follows: Section 2 describes the literature survey. Section 3 describes the proposed methodology, while Section 4 shows the experimental results and the conclusion of this research is given in Section 5.

2. Literature survey

In this section, some of the state-of-the-art methods are discussed related to the idea of task scheduling in CC. Furthermore, this section determines the advantages and limitations of every state-of-the-art method according to its features and operation functions.

Alsaidy [19] presented an enhnaced initialization of Particle Swarm Optimization (PSO) for solving the NP-hard problem in task scheduling. LJFP and MCT approaches were utilized to initialize PSO. The effectiveness of LJFP-PSO as well as MCT-PSO approaches were estimated through makespan, total execution time, DOI as well as total energy consumption. An introduced approach attained effective DOI through initializing the weights in PSO. However, the presented approach did not estimate resource utilization, thereby leading to resource overload, so it leads to node failures.

Khan and Santhosh [20] developed the hybrid approach of PSO and Grey Wolf Optimization called (PSGWO) for solving the problem of task scheduling. The developed approach effectively scheduled the task with a minimum amount of waiting time. Furthermore, the parameters like resource utilization, efficiency of the model, execution time and production time were considered during task scheduling. Support Vector Machine (SVM) was utilized for the classification of VM. The developed approach effectively searched by a large number of schedules to identify the solutions. Nonetheless, the developed model was not adapted to specific characteristics of the task scheduling due to complex constraints and dependencies.

Kruekaew and Kimpan [21] presented the Artificial Bee Colony (ABC) with Q-learning approach for the multi-objective task scheduling called MOABCQ in CC. A Q-learning is a Reinforcement Learning (RL) approach that supported for the ABC approach to work faster. The presented approach targeted to optimize the scheduling and resource utilization, enhanced the VM throughput as well as developed load balancing between the VMs according to makespan, cost and resource utilization. An ABC had faster convergence speed and efficient search capability, and so supported in enhancing the throughput. However, the effectiveness of the MOABCQ was not optimized to estimate the efficacy of the approach in every test case, so it causing a poor performance.

Emami [22] developed the Enhanced Sunflower Optimization (ESFO) approach for the partition of tasks on shared resources for the minimization of energy consumption and makespan. In ESFO, the

new pollination strategy was developed to enhance the exploration and exploitation capabilities for searching the solution space. The developed approach identified the optimal task scheduling problem with a polynomial time complexity. Nevertheless, the suggested approach was estimated only on limited number of parameters, and it does not estimate the other parameters to show the model efficiency.

Alghamdi [23] implemented the Binary Particle Swarm Optimization (BPSO) for obtaining optimal scheduling of user tasks. The BPSO was utilized to plan and balance a large number of heterogeneous VMs efficiently. The Artificial Neural Network (ANN) was utilized with the BPSO for attaining effective scheduling in the cloud. The implemented approach effectively considered the task priority in VM task sequence and utilized ANN with the BPSO. Nonetheless, the implemented approach does not consider the energy consumption and difficulty in handling dynamic environments.

From this section, the limitations of previous methods are: resource utilization and energy consumption were not identified, estimation of only the minimum number of parameters were carried out. These limitations are addressed in this research manuscript based on task scheduling approaches in the cloud. The CSGJO improves fuzzy rule sets by refining the rule structure and parameter tuning. Following that, it identifies the optimal resource allocation to minimize the response time even when handling a large number of tasks.

3. Proposed methodology

The aim of this research is to efficiently schedule tasks over different VMs in the CC by using the FL and CSGJO approaches. The contribution of this approach is to efficiently schedule the tasks and allocate resources based on the user requirements at minimum execution time. Fig. 1 shows the architecture of task scheduling.

3.1 System architecture

The number of tasks such as $ta_k = \{ta_1, ta_2, ta_3, \dots, ta_k\}$ is denoted as k, with n number of VM such as $v_n = \{v_1, v_2, v_3, \dots, v_n\}$, i number of physical hosts such as $h_i = \{h_1, h_2, h_3, \dots, h_i\}$, and j number of datacenters such as $d_j = \{d_1, d_2, d_3, \dots, d_i\}$. These k number of tasks are efficiently scheduled by n number of VMs, which exist in i physical hosts and j datacenters. Mapping or scheduling the tasks requires to examine the priorities of tasks and VM, while reducing the makespan and energy consumption. The makespan as well as energy consumption is reduced by running the multiple tasks concurrently in the scheduling process. The number of tasks determines the number of VMs and it should be allocated on physical hosts based on the resource requirements. The number of physical hosts determines the capacity and scalability of a datacenter. Therefore, the efficiency is improved by assigning the task priorities based on the factors such as deadlines, importance or criticality.

3.2 Load balancing

Load balancing [24] is the procedure of allocating the load in the network to enhance resource utilization with high throughput. It is the network device utilized in the distributed systems to improve resource utilization and to ignore the single point of failure.

Hence, load balancing supplements the entire system as it assures the reliability as well as availability. However, if the load balancing is not performed in an appropriate way, it leads to challenges in the networking. Hence, in this research, the FL approach is utilized to efficiently solve load balancing. The detailed information about FL is provided the subsequent sections.

3.2.1. Fuzzy logic algorithm

Fuzzy inference is the procedure of formulating the mapping of the provided input to the output by utilizing FL, and on the other hand, mapping gives a basis from which the decision must be made or patterns are to be recognized. FL analyzes the historical traffic patterns and predicts the future load conditions on the servers or VM. Then, it dynamically adjusts load balancing decisions based on these predictions. In this research, the fuzzifier employs the fuzzification process to convert the two types of an input data such as processing speed and allocated load of the VM. Additionally, it is assumed that the balanced load is the output required in an inference system. In this proposed method, the execution time and load in VM are considered as the two input parameters to develop an efficient value to balance the load in cloud by utilizing FL.

The defuzzification is performed by using the Smallest of Minimum (SOM) approach to transform the fuzzy outcome set in the individual number. The fuzzy set aggregation involves a range of outcome values that are defuzzied based on solving the individual output value from the fuzzy set. The defuzzifier accepts the aggregated linguistic values from an indirect fuzzy control action and develops the non-fuzzy control output, which determines the

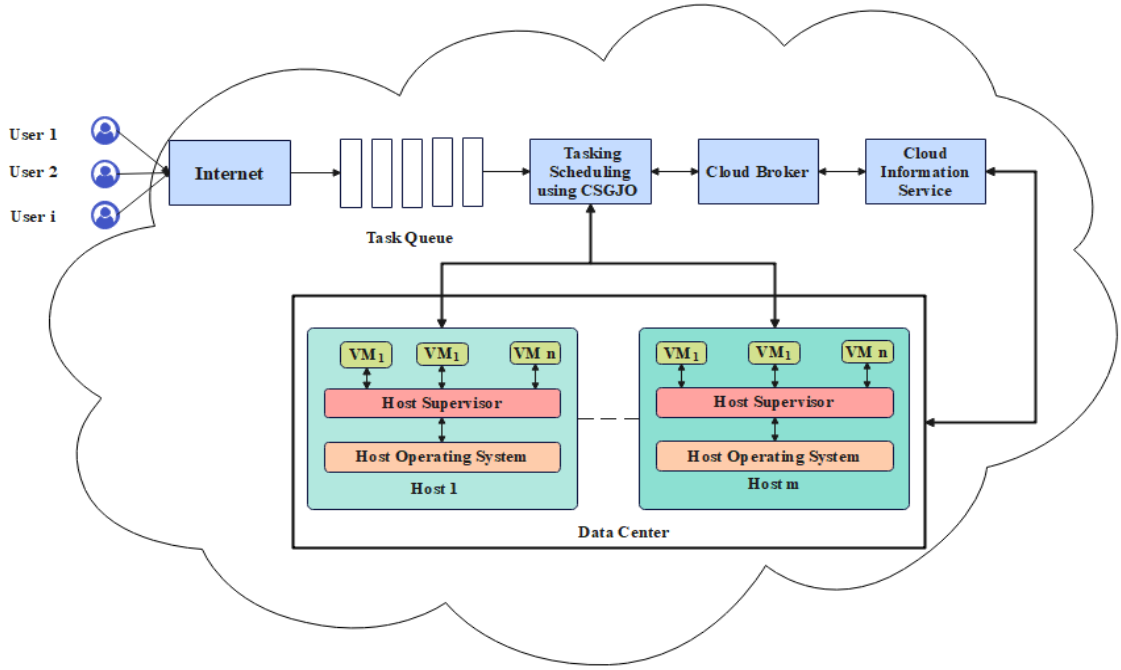

Figure. 1 Workflow of the task scheduling

accepted balanced load to the load conditions. The defuzzification approach is performed to implement the membership function for an accepted result.

In membership function, the fuzzy set is represented mathematically for assigning to every possible individual in the dissertation value by denoting its membership grade in a fuzzy set. These membership grades are depicted through the actual number values ranging between 0 and 1. T is a function of T: $[0,1] \times [0,1] \rightarrow [0,1]$. With the aid of fuzzy set A involves all elements of x, which has a non-zero membership grade. $\text{Support}(A) = \{x \in X | u_A(x) > 0\}$. Assuming that the queue length and fuzzy sets of the CPU represents the linguistics concepts system load as low, moderate and high. The fuzzification is developed for every input variable to express an integrated measurement uncertainty. Because of the efficient clustering of the data attributes using FL, the scheduling approach has effectively balanced the load over the VM as cloud server.

3.3 System architecture

The scheduling is an equilibrium condition in which activities are arranged based on the specific criteria and a specific method. The aim of the scheduling approach is to minimize an execution time of tasks. The CSGJO is performed to schedule the tasks according to the optimal fitness value, which supports to identify the optimal load system based on their server capacity.

3.3.1. Golden jackal optimization algorithm

GJO [25] is a meta-heuristic optimization apporach which inspired through hunting behavior of golden jackals. In GJO, every golden jackal determines the search agent or the candidate solution. The prey searching, encircling and attacking are the general things followed by the GJO. The parameters in GJO are control variables such as population size, number of iterations end exploration rate that direct the search process. In GJO, the search space is described by the variables which depict the potential solutions to an optimization problem.

1) **Search space design**

GJO is the population-based approach where the initial positions are arbitrarily placed in the search area, as formulated in Eq. (1).

$$Y_0 = Y_{min} + rand \times (ub - lb) \quad (1)$$

Where, Y_0 is the initial randomized population, rand is the random number which falling in the range of 0 and 1. Y_{min} is the minimum value. ub and lb are the upper and lower boundaries of decision variables. The initialization process comprises developing the prey matrix with Male Jackals (MJ) as well as Female Jackals (FMJ) inhabiting the primary and secondary positions, respectively. The prey matrix is formulated in Eq. (2).

$$Prey = \begin{bmatrix} Y_{1,1} & Y_{1,2} & \dots & Y_{1,d} \\ Y_{2,1} & Y_{2,2} & \dots & Y_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ Y_{n,1} & Y_{n,2} & \dots & Y_{n,d} \end{bmatrix} \quad (2)$$

Where, Y_{ij} is the jth element of ith prey, n and d is the total number of prey and variables. Here, the location determines the optimal solution. In the process of optimization, the objective function is utilized to identify fitness of every prey. A fitness of every prey is calculated using Eq. (3).

$$F_{OA} = \begin{bmatrix} f(Y_{1,1}; Y_{1,2}; \dots; Y_{1,d}) \\ f(Y_{2,1}; Y_{2,2}; \dots; Y_{2,d}) \\ \vdots \\ f(Y_{n,1}; Y_{n,2}; \dots; Y_{n,d}) \end{bmatrix} \quad (3)$$

Where, f is an objective function, and F_{OA} is the matrix for storing fitness of every prey. $f(.)$ – function of multiple variables $Y_{i,j}$, where, i ranges from 1 to n and j ranges from 1 to d.

2) Exploration phase

The jackals have the ability to detect and track a prey, and they track down the prey effortlessly. The MJ usually carries the lead, whereas FMJ trains the MJ which is formulated in Eqs. (4) and (5).

$$Y_1(t) = Y_M(t) - E.|Y_M(t) - rl.Prey(t)| \quad (4)$$

$$Y_2(t) = Y_{FM}(t) - E.|Y_{FM}(t) - rl.Prey(t)| \quad (5)$$

Where, t is the present iterations, $Prey(t)$ is the position vector, $Y_M(t)$ and $Y_{FM}(t)$ are the positions of MJ and FMJ in a search space, E is the prey's escape energy, rl is the random vectors according to the Levy Flight (LF) distribution, as formulated in Eq. (6) and LF is formulated in Eq. (7).

$$rl = 0.05 * LF(y) \quad (6)$$

$$LF(y) = 0.01 \times \frac{(\mu \times \sigma)}{\left(\left|v^{\left(\frac{1}{\beta}\right)}\right|\right)}; \sigma = \left(\frac{\Gamma(1+\beta) \times sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{(1+\beta)}{2}\right) \times \beta \times \left(2^{\frac{\beta-1}{2}}\right)}\right)^{1/\beta} \quad (7)$$

Where, β is a constant fixed to 1.5, μ, σ is the scaling factor between the range of 0 and 1. $\Gamma(.)$ is the gamma function. $LF(y)$ is the output of LF. $|.|$ is the absolute value.

Eventually, the updated position is formulated in Eq. (8).

$$Y(t+1) = \frac{Y_1(t) + Y_2(t)}{2} \quad (8)$$

Where, $Y(t+1)$ is the variable Y at the next time step $t+1$.

3) Exploitation phase

Once the prey's escaping capability is minimized, the jackals encircle the prey identified in the prior stage. After encircling, the jackals jump on the prey and bother it. The collected pursuing of the jackal's hunting behavior is denoted in Eqs. (9) and (10).

$$Y_1(t) = Y_M(t) - E.|rl.Y_M(t) - Prey(t)| \quad (9)$$

$$Y_2(t) = Y_{FM}(t) - E.|rl.Y_{FM}(t) - Prey(t)| \quad (10)$$

The goal of rl is shown in the above Eqs. (9) and (10) used to generate the random behavior in an exploitation phase, supporting behavior, ignoring the local optima problem.

4) Switching from exploration to exploitation

In the process of GJO, E value is utilized to be shifted from exploration to exploitation. The prey's energy is minimized meaningfully during an escaping behavior. Once E_0 is reduced from 0 to -1, the prey is flagging and while E_0 enhances from 0 to 1, the prey's strength is enhancing. If $|E| < 1$, jackals attack their prey and move to exploitation.

3.3.2. Cosine similarity based GJO

The traditional GJO updates the jackal's positions through Eq. (8) at the training process, similar to utilizing the mean as an optimal solution. Though, this approach ensures the flatness of the updates of the jackal's positions, but also has limitations. The most apparent error is the lack of considering the relationship among the various features. When there is a relationship among the features, utilizing a mean update device leads to few features that are exaggerated or eliminated, thus affecting its performance. Furthermore, once the data distribution is unequal, the utilization of mean update mechanism leads to the worst prediction effectiveness for the particular data. Hence, the cosine similarity is proposed for updating MJ and FMJ's positions. As estimated to mean update mechanism, the benefit of utilizing the cosine similarity as update mechanism examines the relationship among the various features, hence updating the model parameters efficiently. It is appropriate for the high-dimensional data and which

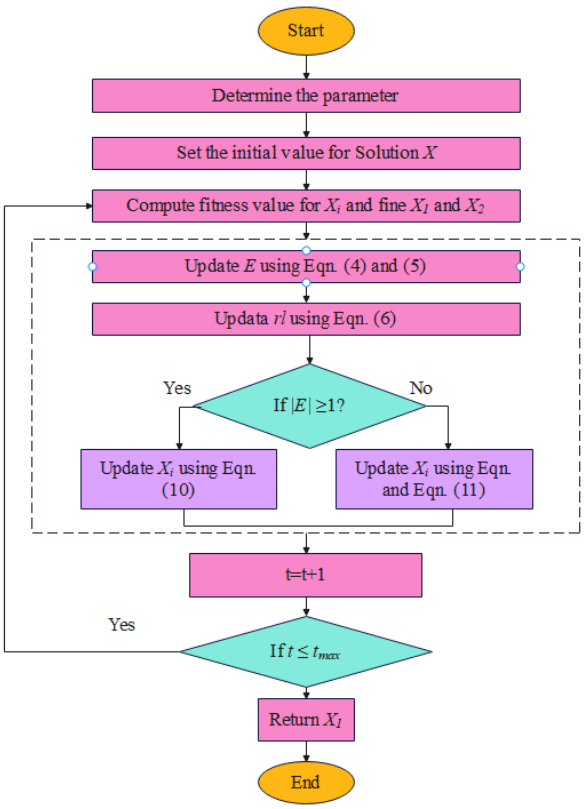

Figure. 2 Flowchart of the CSGJO

is not influenced through the vector length or data distribution. The cosine similarity is displayed in Eq. (11).

$$Cos_{sim}\left(Y_1(t), Y_2(t)\right) = \frac{Y_1(t).Y_2(t)}{\|Y_1(t)\|\|Y_2(t)\|} \tag{11}$$

The cosine similarity among the pairs of the golden jackal is enhanced through the complete value as the position update's weight, which is expressed in Eq. (12).

$$Y(t+1) = Y_1(t) \times \left|Cos_{sim}\left(Y_1(t), Y_2(t)\right)\right| + Y_2(t) \times \left(1 - \left|Cos_{sim}\left(Y_1(t), Y_2(t)\right)\right|\right) \tag{12}$$

Where, $Y_1(t)$ and $Y_2(t)$ is the MJ and FMJ's position, . is the dot product, $\|Y_1(t)\|$ and $\|Y_2(t)\|$ is respectively the MJ and FMJ's lengths. The range value of $Cos_{sim}\left(Y_1(t), Y_2(t)\right)$ is among -1 and 1. By using CSGJO, the tasks are effectively scheduled, and then these tasks are given to the cloud distributor to forward data to the task scheduler. The scheduler has the responsibility for assigning the tasks to every VM over the cloud, through examining its feature attributes such as capacity of the storage, execution time and weight value. Fig. 2 shows the flowchart of the CSGJO.

3.4 Multi-objective functions

In this section, the multi-objective function is considered as the fitness function to estimate the optimal solution. There are five parameters of makespan, DOI, execution time, energy consumption and resource utilization considered for efficient task scheduling and load balancing in the data center. When load balancing and task scheduling work together effectively, they improve an efficiency and performance. The estimated solution of this best fitness value is assumed as an optimal solution. The mathematical expressions of the multi-objective functions are expressed in Eqs. (13) to (17) below.

$$Makespan = max(Ct_i)_{t_i \epsilon T} \tag{13}$$

$$DoI = \frac{Max_{load} - min_{load}}{Average_{load}} \tag{14}$$

$$Execution\ time\ (T_i) = \frac{Number\ of\ instructions\ (task(i))}{MIPS\ rating\ (Machine\ (i))} \tag{15}$$

$$Energy\ Consumption = \frac{Total\ Temperature}{Average\ Temperature} * Total\ number\ of\ resources\ utilized \tag{16}$$

$$Resource\ utilization = \sum_{i=1}^{i=n} Number\ of\ tasks(K_i)\ executedper\ resource\ (R_i) \tag{17}$$

All the above-mentioned objectives are incompatible with each other. Usually, all the objective functions contain different units and values. Therefore, the min-max normalization function is substituted to every objective function to normalize the units into numerical values by using Eq. (18).

$$F(x) = \frac{f_i - f_{min}}{f_{max} - f_{min}} \tag{18}$$

Table 1. Simulation parameters

Parameters	Values
Size of tasks (MI)	1000-4000
VM execution rate (MIPS)	1000-5000
Power consumption	0.6-0.7
Power consumption of VMs (Watt)	200-1000
Number of tasks	100-1000
Number of VMs	40-200

Hence, the weighted sum method is substituted and multi-objective function is converted into an individual objective function. It is formulated in Eq. (19) as:

$$Minimize\ fitness = \alpha_1 \times f_1 + \alpha_2 \times f_2 + \alpha_3 \times f_3 + \alpha_4 \times f_4 + \alpha_5 \times f_5 \quad (19)$$

Where, $max(Ct_i)_{t_i \epsilon T}$ is the maximum completion time between all tasks; min_{load} and max_{load} are minimum and maximum load values observed in a given period; $Average_{load}$ is the average load value. Ct_i is an execution time of the longest task t_i, and T is the number of tasks on an application workflow. α_1 to α_5 are the weights applied to all objective functions, f_i is the function value, f_{min} and f_{max} are the minimum and maximum values, F(x) is the normalized value among 0 and 1. By utilizing the CSGJO, the tasks are effectively scheduled by identifying the optimal resource allocation to minimize the response time even when handling the large number of tasks.

4. Author name(s) and affiliation(s)

In this research, the results and discussion of the proposed method based on the task scheduling in cloud are presented. The simulation is taken out by Cloud Sim simulator based on Java. This research utilizes intel core i7 processor system configuration, windows 10 OS, and 16GB RAM. To measure the effectiveness of the proposed method, different performance matrices such as makespan, DOI, Execution time, Energy consumption and Resource Utilization are considered. Table 1 depicts the parameter settings of the CSGJO.

4.1 Performance analysis

In this section, the effectiveness of the CSGJO is validated for two different scenarios which are provided below. The existing meta-heuristic approaches like PSO, GWO and Ant Colony Optimization (ACO) are compared with the proposed CSGJO. As compared to these existing approaches, the proposed CSGJO attains better results, because it effectively schedules the tasks and balances the load based on the task priorities.

4.1.1. Scenario 1

The number of tasks is taken from 200 to 1000 and the total number of VM is set to 100 for examining the task scheduling in this scenario. Table 2 depicts the analysis of CSGJO for scenario 1. In Table 2, the comparison of the CSGJO with the existing approaches is discussed by using various performance metrices. The results from table 2 show that the enhancing of the number of tasks leads an improvement in the number of resources. The CSGJO attains better outcomes as compared to the existing methods, even when the number of tasks is increased. A CSGJO offers superior scalability than the existing approaches for minimum and maximum number of tasks.

Table 2. Analysis of CSGJO for Number of tasks at VM = 100

Performance Metrics	Method	Number of tasks				
		200	400	600	800	1000
Makespan (s)	PSO	3.8	6.7	12.3	17.7	22.3
	GWO	3.2	5.5	11.4	16.9	20.6
	ACO	2.5	4.7	10.6	16.4	19.7
	CSGJO	1.9	3.8	8.3	11.5	16.7
DOI	PSO	0.42	0.45	0.51	0.53	0.57
	GWO	0.31	0.41	0.49	0.52	0.55
	ACO	0.28	0.38	0.44	0.39	0.41
	CSGJO	0.23	0.25	0.31	0.33	0.34
Execution Time (s)	PSO	179.5	388.9	598.7	791.2	986.1
	GWO	172.6	367.8	577.2	789.3	978.3
	ACO	169.3	345.7	567.8	772.6	971.2
	CSGJO	153.6	312.7	543.1	753.7	954.0
Energy Consumption (J)	PSO	167	321	563	711	867
	GWO	145	298	523	703	842
	ACO	135	278	498	685	834
	CSGJO	117	236	423	657	806
Resource Utilization (%)	PSO	92.3	92.1	91.2	90.2	91.3
	GWO	95.2	94.3	93.2	93.1	93.2
	ACO	96.2	97.5	95.6	96.4	96.4
	CSGJO	98.9	99.1	99.2	99.5	99.5

4.1.2. Scenario 2

The number of VM taken from 40 to 200 VMs and the number of tasks is fixed to the 500 in this scenario. In Table 3, the comparison of the CSGJO with the existing approaches are discussed by using various performance metrices.

The results from Table 3 exhibit the enhancing number of task origins and improvement in the number of resources. The CSGJO attains superior outcomes when compared to the existing methods even when increasing the number of tasks. The CSGJO offers a commendable scalability than the existing approaches for minimum and maximum number of tasks. Scalability in task scheduling refers to the ability of a task scheduling system to handle the increasing amount of work or tasks efficiently without significantly impacting the performance.

Table 3. Analysis of CSGJO for Number of VM at Task =500

Performance Metrics	Method	Number of VM				
		200	400	600	800	1000
Makespan (s)	PSO	36.2	18.6	7.3	6.3	5.8
	GWO	35.8	15.7	6.9	5.8	5.3
	ACO	34.6	14.6	6.3	4.5	4.6
	CSGJO	23.7	9.6	4.5	3.3	3.2
DOI	PSO	18.6	3.1	1,7	2.4	3.5
	GWO	15.7	2.6	1.6	1.7	1.4
	ACO	12.5	1.4	1.4	1.6	0.7
	CSGJO	0.9	0.7	0.6	0.5	0.4
Execution Time (s)	PSO	544.7	542.4	446.8	457.8	486.9
	GWO	534.6	534.1	435.6	434.6	479.2
	ACO	512.3	523.6	432.0	421.5	462.1
	CSGJO	475.9	503.2	412.5	403.8	434.5
Energy Consumption (J)	PSO	389	391	403	412	437
	GWO	384	383	398	399	435
	ACO	375	378	378	346	412
	CSGJO	354	356	302	311	321

Table 4. Simulation parameter with the different scenarios

Parameters	Scenarios		
	3	4	5
Number of tasks	200 to 1000	100 to 500	1000 to 5000
Number of VM	100	50	150

Table 5. Comparison of the proposed CSGJO with MCT-PSO [19] and PSGWO [20]

Scenario	Methods	Performance Metrices	No. of tasks			
			200	400	600	1000
3	MCT-PSO [19]	Makespan (s)	2.2	4.2	8.5	17.6
		DOI	0.9	0.6	1.0	1.2
		Execution Time (s)	155.8	326.7	535.8	972.3
		Energy Consumption (J)	122	242	440	822
		Resource Utilization (%)	N/A	N/A	N/A	N/A
	PSGWO [20]	Makespan (s)	75.4	125.6	180.6	375.7
		DOI	N/A	N/A	N/A	N/A
		Execution Time (s)	N/A	N/A	N/A	N/A
		Energy Consumption (J)	N/A	N/A	N/A	N/A
		Resource Utilization (%)	98.5	98.4	98.5	98.4
	Proposed CSGJO	Makespan (s)	1.9	3.8	8.3	16.7
		DOI	0.23	0.25	0.31	0.034
		Execution Time (s)	153.6	312.7	543.1	954.0
		Energy Consumption (J)	117	236	423	806
		Resource Utilization (%)	98.9	99.1	99.2	99.5

Table 6. Comparison of the proposed CSGJO with ESFO [22]

Scenario	Methods	Performance Metrices	No. of tasks			
			100	200	300	400
4	ESFO [22]	Makespan (s)	58	147	188	271
	Proposed CSGJO	Makespan (s)	55	143	183	268

Table 7. Comparison of the proposed CSGJO with ANN-BPSO [23]

Scenario	Methods	Performance Metrices	No. of tasks			
			1000	2000	3000	4000
5	ANN-BPSO [23]	Makespan (s)	90	96	100	120
		DOI	0.0365	0.0628	0.8751	0.0911
		Resource Utilization (%)	96.84	95.21	94.54	94.09
	Proposed CSGJO	Makespan (s)	88	94	97	115
		DOI	0.0342	0.0426	0.0643	0.0834
		Resource Utilization (%)	97.34	96.34	95.35	95.04

Table 8. Comparative Analysis with different number of VM

Performance metric	Method	Number of Virtual Machines			
		40	80	120	160
Makespan (s)	MCT-PSO [19]	25.9	9.7	4.9	3.7
	Proposed CSGJO	23.7	9.6	4.5	3.3
DOI	MCT-PSO [19]	1.0	0.8	0.7	0.7
	Proposed CSGJO	0.9	0.7	0.6	0.5
Execution Time (s)	MCT-PSO [19]	486.8	518.7	427.0	417.6
	Proposed CSGJO	475.9	503.2	412.5	403.8
Energy Consumption (J)	MCT-PSO [19]	362	365	311	318
	Proposed CSGJO	354	356	302	311

4.2 Comparative analysis

Table 4 shows the simulation parameters of the different scenarios. The scenarios 3 depicted the MCT-PSO [19] and PSGWO [20], whereas scenario 4 and 5 are depicted as the ESFO [22] and ANN-BPSO [23] respectively. These different scenarios are compared with the proposed CSGJO approach for estimating the effectiveness of the model. Table 5, 6, 7 and 8 shows the comparison of the proposed CSGJO with MCT-PSO [19], PSGWO [20], ESFO [22] and ANN-BPSO [23]. The performance of proposed CSGJO is compared with the MCT-PSO [19] and PSGWO [20] in scenario 3 using the number of tasks of 200, 400, 600 and 1000 respectively. The performance of proposed CSGJO is compared with the ESFO [22] in scenario 4 using the number of tasks of 100, 200, 300 and 400 respectively. The performance of proposed CSGJO is compared with the ANN-BPSO [23] in scenario 5 using the number of tasks of 1000, 2000, 3000 and 4000 respectively. Table 8 represents the comparative analysis of the proposed CSGJO with MCT-PSO [19] on a number of VMs. The number of VMs such as 40, 80, 120 and 160 respectively.

4.3 Discussion

In this section, the achievement of the proposed CSGJO approach is discussed along with their advantages. The existing works have the limitations such as resource utilization and energy consumption not being identified, estimating only a minimum number of parameters, while also the efficiency not being optimized. Hence, in this research, the resource utilization and energy consumption are estimated to schedule the tasks and to minimize the over and underutilization problems. The proposed CSGJO effectively estimates the different parameters namely, makespan, DOI, execution time, energy consumption and resource utilization through continuously improving the scheduling process based on the actual data. CSGJO leverages the cooperative hunting behavior of golden jackals to efficiently allocate tasks to resources, also minimizing the makespan. The algorithm dynamically adjusts the task assignments based on VM load and task requirements. The CSGJO effectively refines the fuzzy sets and identifies the optimal resource allocation. The proposed CSGJO attains minimum makespan, DOI,

execution time, energy consumption and resource utilization of 1.9s, 0.23, 153.6s, 117J and 98.9%, respectively. These results exhibit a commendable improvement as compared to the existing approaches.

5. Conclusion

In this section, the achievement of the proposed CSGJO approach is discussed along with their advantages. The existing works have the limitations such as resource utilization and energy consumption not being identified, estimating only a minimum number of parameters, while also the efficiency not being optimized. Hence, in this research, the resource utilization and energy consumption are estimated to schedule the tasks and to minimize the over and underutilization problems. The proposed CSGJO effectively estimates the different parameters namely, makespan, DOI, execution time, energy consumption and resource utilization through continuously improving the scheduling process based on the actual data. CSGJO leverages the cooperative hunting behavior of golden jackals to efficiently allocate tasks to resources, also minimizing the makespan. The algorithm dynamically adjusts the task assignments based on VM load and task requirements. The CSGJO effectively refines the fuzzy sets and identifies the optimal resource allocation. The proposed CSGJO attains minimum makespan, DOI, execution time, energy consumption and resource utilization of 1.9s, 0.23, 153.6s, 117J and 98.9%, respectively. These results exhibit a commendable improvement as compared to the existing approaches.

In CC, obtaining an efficient task scheduling approach is significant for cloud providers and users. In this research, CSGJO approach is proposed for efficient task scheduling in a cloud. The CSGJO has the capability to handle a large number of tasks and resources, furthermore, efficiently optimizing the schedules even for complex scenarios with various constraints. Commencing the search process with the LJFP and MCT approaches based on the selected solutions efficiently effect a convergence speed, as well as the performance. The FL approach is utilized to solve the problem of load balancing effectively, thereby optimally distributing the tasks according to the fuzzy rules and supporting to minimize the execution time. The proposed CSGJO accomplishes a minimum makespan of 1.9s at the number of tasks being 200. In the future, different meta-heuristic approaches will be used with a greater number of parameters to enhance the system performance.

Notation

Variables	Descriptions
Y_0	Initial randomized population
rand	Random number which falling in the range of 0 and 1
Y_{min}	Minimum value
ub and lb	Upper and lower boundaries of decision variables
Y_{ij}	jth element of ith prey
n and d	Total number of prey and variables
f	Objective function
F_{OA}	Matrix for storing fitness of every prey
$f(.)$	Function of multiple variables $Y_{i,j}$, where, i ranges from 1 to n and j ranges from 1 to d.
t	Present iterations
$Prey(t)$	Position vector
$Y_M(t)$ and $Y_{FM}(t)$	Positions of MJ and FMJ in the search space
E	Prey's escape energy
rl	Random vectors according to the Levy Flight (LF) distribution
β	Constant fixed to 1.5
μ, σ	Scaling factor between the range of 0 and 1.
$(.)$	Gamma function
$LF(y)$	Output of LF
$\|.\|$	Absolute value
$Y(t+1)$	Variable Y at the next time step $t+1$
$Y_1(t)$ and $Y_2(t)$	MJ and FMJ's position
.	Dot product
$\|Y_1(t)\|$ and $\|Y_2(t)\|$	MJ and FMJ's lengths
$max(Ct_i)_{t_i \epsilon T}$	Maximum completion time between all tasks
min_{load} and max_{load}	Minimum and maximum load values observed in a given period
$Average_{load}$	Average load value
Ct_i	Execution time of the longest task t_i
T	Number of tasks on the application workflow
α_1 to α_5	Weights applied to all objective functions
f_i	function value
f_{min} and f_{max}	Minimum and maximum values
(x)	Normalized value among 0 and 1

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization, MZR and ASP; methodology, MZR; software, ASP; validation, ASP; formal analysis, ASP; investigation, MZR; resources, ASP; data curation, MZR; writing—original draft preparation, MZR; writing—review and editing, ASP; visualization, MZR; supervision, ASP; project administration, ASP

References

[1] N.K. Walia, N. Kaur, M. Alowaidi, K.S. Bhatia, S. Mishra, N.K. Sharma, S.K. Sharma, and H. Kaur, "An energy-efficient hybrid scheduling algorithm for task scheduling in the cloud computing environments", *IEEE Access*, Vol. 9, pp. 117325-117337, 2021.

[2] H. Mahmoud, M. Thabet, M.H. Khafagy, and F.A. Omara, "Multiobjective task scheduling in cloud environment using decision tree algorithm", *IEEE Access*, Vol. 10, pp. 36140-36151, 2022.

[3] J.K. Konjaang, J. Murphy, and L. Murphy, "Energy-efficient virtual-machine mapping algorithm (EViMA) for workflow tasks with deadlines in a cloud environment", *Journal of Network and Computer Applications*, Vol. 203, p. 103400, 2022.

[4] S. Mangalampalli, G.R. Karri, and M. Kumar, "Multi objective task scheduling algorithm in cloud computing using grey wolf optimization", *Cluster Computing*, Vol. 26, No. 6, pp. 3803-3822, 2023.

[5] H. Mahmoud, M. Thabet, M.H. Khafagy, and F.A. Omara, "An efficient load balancing technique for task scheduling in heterogeneous cloud environment", *Cluster Computing*, Vol. 24, No. 4, pp. 3405-3419, 2021.

[6] X. Fu, Y. Sun, H. Wang, and H. Li, "Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm", *Cluster Computing*, Vol. 26, No. 5, pp. 2479-2488, 2023.

[7] L. Yin, C. Sun, M. Gao, Y. Fang, M. Li, and F. Zhou, "Hyper-Heuristic Task Scheduling Algorithm Based on Reinforcement Learning in Cloud Computing", *Intelligent Automation & Soft Computing*, Vol. 37, No. 2, pp. 1587-1608, 2023.

[8] B. Sellami, A. Hakiri, S.B. Yahia, and P. Berthou, "Energy-aware task scheduling and offloading using deep reinforcement learning in SDN-enabled IoT network", *Computer Networks*, Vol. 210, p. 108957, 2022.

[9] G. Shruthi, M.R. Mundada, B.J. Sowmya, and S. Supreeth, "Mayfly taylor optimisation-based scheduling algorithm with deep reinforcement learning for dynamic scheduling in fog-cloud computing", *Applied Computational Intelligence and Soft Computing*, Vol. 2022, p. 2131699, 2022.

[10] D.A. Amer, G. Attiya, I. Zeidan, and A.A. Nasr, "Elite learning Harris hawks optimizer for multi-objective task scheduling in cloud computing", *The Journal of Supercomputing*, Vol. 78, No. 2, pp. 2793-2818, 2022.

[11] K.R.R. Laxman, A. Lathigara, R. Aluvalu, and U.M. Viswanadhula, "PGWO-AVS-RDA: An intelligent optimization and clustering based load balancing model in cloud", *Concurrency and Computation: Practice and Experience*, Vol. 34, No. 21, p. e7136, 2022.

[12] S. Nabi, M. Ahmad, M. Ibrahim, and H. Hamam, "AdPSO: adaptive PSO-based task scheduling approach for cloud computing", *Sensors*, Vol. 22, No. 3, p.920, 2022.

[13] P. Pirozmand, H. Jalalinejad, A.A.R. Hosseinabadi, S. Mirkamali, and Y. Li, "An improved particle swarm optimization algorithm for task scheduling in cloud computing", *Journal of Ambient Intelligence and Humanized Computing*, Vol. 14, No. 4, pp. 4313-4327, 2023.

[14] A. Chhabra, K.C. Huang, N. Bacanin, and T.A. Rashid, "Optimizing bag-of-tasks scheduling on cloud data centers using hybrid swarm-intelligence meta-heuristic", *The Journal of Supercomputing*, Vol. 78, No. 7, pp. 9121-9183, 2022.

[15] N. Sharma, and P. Garg, "Ant colony-based optimization model for QoS-Based task scheduling in cloud computing environment", *Measurement: Sensors*, Vol. 24, p. 100531, 2022.

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

[17] K. Malathi, and K. Priyadarsini, "Hybrid lion–GA optimization algorithm-based task scheduling approach in cloud computing", *Applied Nanoscience*, Vol. 13, No. 3, pp. 2601-2610, 2023.

[18] F.S. Alsubaei, A.Y. Hamed, M.R. Hassan, M. Mohery, and M.K. Elnahary, "Machine learning approach to optimal task scheduling in cloud

communication", *Alexandria Engineering Journal*, Vol. 89, pp. 1-30, 2024.
[19] S.A. Alsaidy, A.D. Abbood, and M.A. Sahib, "Heuristic initialization of PSO task scheduling algorithm in cloud computing", *Journal of King Saud University-Computer and Information Sciences*, Vol. 34, No. 6, pp. 2370-2382, 2022.
[20] M.S.A. Khan, and R. Santhosh, "Task scheduling in cloud computing using hybrid optimization algorithm", *Soft computing*, Vol. 26, No. 23, pp. 13069-13079, 2022.
[21] B. Kruekaew, and W. Kimpan, "Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning", *IEEE Access*, Vol. 10, pp. 17803-17818, 2022.
[22] H. Emami, "Cloud task scheduling using enhanced sunflower optimization algorithm", *Ict Express*, Vol. 8, No. 1, pp. 97-100, 2022.
[23] M.I. Alghamdi, "Optimization of load balancing and task scheduling in cloud computing environments using artificial neural networks-based binary particle swarm optimization (BPSO)", *Sustainability*, Vol. 14, No. 19, p. 11982, 2022.
[24] F.M. Talaat, H.A. Ali, M.S. Saraya, and A.I. Saleh, "Effective scheduling algorithm for load balancing in fog environment using CNN and MPSO", *Knowledge and Information Systems*, Vol. 64, No. 3, pp. 773-797, 2022.
[25] J. Zhang, G. Zhang, M. Kong, and T. Zhang, "Adaptive infinite impulse response system identification using an enhanced golden jackal optimization", *The Journal of Supercomputing*, Vol. 79, No. 10, pp. 10823-10848, 2023.