Received: May 19, 2023. Revised: July 25, 2023.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Task Classification and Scheduling Using Enhanced Coot Optimization in Cloud Computing

Syed.Karimunnisa[1]* **Yellamma Pachipala[1]**

[1]*Department of Computer Science and Engineering,*
Koneru Lakshmaiah Education Foundation Vaddesvaram, AP, 522502, India
* Corresponding author's Email: karimun1.syed@gmail.com, pachipala.yamuna@gmail.com.

Abstract: Cloud computing benchmarks the dream of rendering computing as a utility, providing high agility and reachability from an existing set of technologies. It facilitates a wider dimension to architect and manage remote resources. Cloud technology with exponential growth is drilling towards issues that tend to lower its explored possibilities. As cloud systems by virtue deal with various virtualized resources, scheduling is opted as an important metric for measuring and leveraging performance. But scheduling efficiency is deteriorated by various parameters that pave scope for our research and projects immense need for improvising the overall makespan of the system. The proposed work aims at projecting a greater drift in the first phase by witnessing a sequence of phases like pre-processing the user tasks for improved accuracy, classifying the tasks with respect to resource demand and execution time using the improved density based clustering method (IDCM). The second phase deals with enhanced coot optimization algorithm for task scheduling (ECOA-TS) that proceeds and proves its novelty by adopting Cauchy mutation overcoming the convergence backdrop for generating an optimal mapping between clustered user tasks and VMs. The overall performance of the proposed work overrides by reduced makespan against existing state-of-the-art optimization algorithms like particle swam optimization (PSO), grey wolf optimization (GWO) and whale optimization algorithm (WOA) by 27.41%, 19.8%, and 15.33% respectively.

Keywords: Task scheduling, Virtual machines, Classification, Fuzzy clustering, Optimization, Quality of service.

1. Introduction

Developments in the economic standards of present societies have given birth to an era of big data. Voluminous data generated from the internet broadens the scope for evolving technologies among which cloud computing ranks to be the voted one. Cloud computing facilitates appropriate, on-demand access to network scaling customized pool of functional resources through a virtualization mechanism [1]. Resource sharing with underlying virtualization aids in attaining coherence and economic feasibility.

Cloud computing renders several advantages at the risk of many challenging issues like security, cost management, multi-cloud disparities, and interoperability performance issues [2, 3]. Practically, digging deep into the performance issue of the cloud environment, scheduling tasks optimally, and efficient allotment of resources are the major hurdles. Hence this area of cloud computing is driving the attention of researchers and throwing challenges to practitioners.

Task scheduling is a process of ordering the user tasks for efficient utilization of cloud resources in a fashion that leverages the overall performance [4]. Eventually, user applications in the cloud environment are submitted over the internet media online and these applications divided as tasks with various uncertain characteristics that outworths researchers to get deep into the challenges posed by dynamically changing task behaviour [5]. The uncertain nature of tasks is hindering the service provider to complete the tasks at a given time resulting in performance downfall. This owes to be a serious issue to service providers and thereby needs an efficient task scheduling and resource allocation for improving QoS of the cloud system.

As an attempt to aforementioned problems, many researchers came up with [6-8] bio-inspired optimization algorithms for efficient scheduling tasks and allocating resources. Few practitioners achieved enhancement by initially classifying tasks related to historical data and designing various types of VMs [9, 10].

Major pitfalls are identified during the assignment phase of scheduling to various VMs, where in some cases larger tasks are assigned to less capable VMs and higher capacity VMs are left with smaller tasks. This leads to lowered performance where tasks failed to complete at specified deadlines. As a matter of fact, such imbalanced and improper assignment is handled in our proposed work by clustering and classifying the tasks and grouping the VMs into various classes for efficient scheduling.

Our proposed approach encompasses of two phases, where the first phase performs clustering of tasks by classifying them as Low Processor- Memory Intensive group (LPMI), High Processor-Memory Intensive group (HPMI), Processor Intensive group (PI), and the Memory Intensive group (MI).The respective categories of tasks are pipelined into the task queue for assigning them to proper VMs using proposed enhanced coot optimization algorithm, (ECOA-TS) which aids in optimal task scheduling and resource allocation with lowered makespan and improved resource utilization.

The contributions of the profound work can be phased as below.

- The proposed work defines a framework for classifying the pre-processed user tasks using an improved density based clustering method based on task prerequisites enhancing the quality of classification results.
- An enhanced coot optimization algorithm proposed for efficient scheduling of user tasks facilitating optimal mapping to respective virtual machines.
- The proposed model results compared against existing state-of-art optimization algorithms in a simulated environment using cloudsim.

The flow of the paper progresses initially by introducing the concepts and issues concerned with cloud computing and its challenging factors in section 1. Section 2 narrates the intense literature survey undergone and their respective analysis and discussions of researchers' contributions. Section 3 gives system model and problem description. Section 4 describes the contributed work, its architecture, and the system flow of contributed methodology. Section 5 summarizes the outcome of our proposed methodology in comparison with existing algorithms and is depicted using a graphical representation. The conclusion of the work is addressed in Section 6, which images future directions and enhancements achievable.

2. Related work

Several researchers have contributed innumerable solutions addressing issues of scheduling and resource allocation.

S.kanwal et.al [11] explored genetic algorithm based intelligent scheduling approach with added fault tolerant features for enhanced task scheduling. The proposed method passes through four stages namely task phase then local phase succeeded by global phase and final phase dealing with fault tolerance. The approach overtakes the basic genetic algorithm and adaptive models in metrics like execution time, memory usage and overall cost.

Praveen s. et.al [12] contributed a hybrid algorithm that combines the features of basic genetic algorithm and the gravitational emulation local search for overriding the pitfalls of legacy models like PSO & GA in terms of performance related to execution time. The proposed model mainly focusses on the issues related to the size of the search space and search strategies for finding optimal solution.

Seema A. Alsaidy et.al [13] contributed an improved version of PSO Algorithm termed LJFP-PSO (Longest job to fastest processor) and MCT_PSO (minimum completion time) an metaheuristic algorithms with improved initialization parameters. The Algorithm tends to refine the initialization factors of basic PSO and schedule longest task to fastest processor, resulting in reduced makespan, execution time, degree of imbalance and energy consumption.

Poria Pirozmand et.al [14] proposed an improved PSO for efficient task scheduling using a multi adaptive learning strategy by defining particles as ordinary and local best for particle swam optimization. The approach ascertains reduced varieties of population thereby promising increased likelihood of reaching the local optima, resulting in enhanced performance with respect to evaluation metrices like makespan, load balancing, effectiveness and stability when tested against other approaches.

Said Nabi et.al [15] contributed an adaptive PSO based task scheduling for lowered task execution time and improved throughput. The proposed method introduces linearly descending and adaptive inertial weight strategy that aids in providing a balance of local and global search for optimal scheduling of tasks.

Mohammad zadeh et.al [16] presented an improved chaotic binary grey wolf optimization (IGWO) algorithm, that aimed at minimizing execution cost and makespan time when compared with existing approaches. The proposed approach targeted on leveraging the convergence speed and avoiding the algorithm from falling into local optimum, using the cha's theory and hill climbing techniques that rendered improved performance.

Hongji Liu et.al [17] proposed an adaptive ant colony optimization algorithm for task scheduling in cloud environment. The work shows improvement in terms of adaptive updating of pheromone values for improving the convergence speed of contributed algorithm, resulting in improved execution time, cost and improved load balancing compared with traditional ACO algorithm.

Zade et.al [18] contributed a two-phase algorithm which resulted in improved performance with first phase dealing with meta scheduling that assigns the tasks to host machine and in the second phase the scheduling is rein enforced using parallel reinforcement learning caledonian crow for optimal local scheduling to present optimal mapping of tasks and VMs.

Sudheer Mangalampalli et.al [19] proposed an optimal scheduler which targets reduced energy consumption and power costs at datacenters by considering the priorities of tasks and VM's. The optimizer uses an improved whale optimization that prioritizes energy parameters pertaining to multi-objective fitness function for scheduling the appropriate tasks to relative VM's thereby improving the quality metric of reduced power consumption and makespan again existing approaches.

Lewei Jia et.al [20] proposed an improved whale optimization algorithm for efficient scheduling, cost reduction and resource utilization. The author's work initially performs task scheduling and designs a distributed model, successively generating an optimal and feasible plan for each individual whale by considering inertial weight strategy. This aids in improving local weight ability and avoids early convergence.

Jian Li et al. [21] proposed an improved FIFO scheduling algorithm which initially builds a task model and resource model. Further task preferences are considered for which resource clusters are assigned that are constructed based on fuzzy clustering. The author's work success in bringing down the tasks waiting time and improves resource utilization.

Jivrajani et.al. [22] contributed a scheduling approach by grouping similar jobs into batches using hierarchical clustering for executing them on clustered virtual machines with similar resource characteristics. The proposed method excels in improved makespan time, as it selects the best cluster of machines for executing the grouped tasks. The work focuses on reduced energy consumption.

Junqing Li et.al [23] proposed a hybrid approach based artificial bee colony algorithm (ABC) for handling issues related to scheduling. The approach includes an enhanced perturbation structure for extended searching proficiency and improved selection and updating mechanism leveraging outcome. The work sheerly aids in minimizing the makespan time and holds to be robust enough with various problems.

Sudheer M et.al [24] have contributed an efficient W-Scheduler that mainly relies on the SLA specifications of the task being considered. The work progresses by taking the task priorities and VM's, relative to their SLA policies which is subjected to Whale Optimization for enhanced scheduling aiding in lowered make span and SLA Violations in comparison to other competitive approaches.

As per the literature survey many metaheuristic algorithms reflected improved performance by scheduling user tasks to VMs in cloud environment but still left scope for improvements in aspects like efficient resource utilization and task requirements. Majority of earlier research contributions are looked upon for improvement pertaining to resource capability and user requirements.

3. System model and problem description

This section describes the design of the system and problem description for optimal task scheduling in the proposed approach.

System model: Scheduling problems deal with the proper assignment, ordering, and managing of tasks in a timely fashion. In a cloud environment, heterogeneous tasks with heterogeneous resource requirements assemble for execution, which needs to be scheduled effectively for improved parameters like makespan time, execution cost etc. Cloud-based systems formulate scheduling issues as a methodical approach for mapping 'N' jobs on 'M' machines that ensures balanced task distribution on every machine resulting in reduced execution time. The ideal goal of the scheduling algorithm is improved task allotment across VMs and maintaining minimum variation among workloads of available VMs.

Problem description: This section explores the mathematical characteristics pertaining to task

Received: May 19, 2023. Revised: July 25, 2023.

scheduling problem. Tasks owe to be the needs of the user to be accomplished by the computing devices. Assuming 'N' independent Tasks whose dimensions are characterized with parameters as follows:

$$Task_i = \begin{Bmatrix} Task_{id,} Task_{len,} Task_{pes,} Task_{bdwidth,} \\ Task_{storage,} Task_{inputfile,} Task_{outputfile} \end{Bmatrix} \quad (1)$$

Where i=1,2,..,N and for each i^th task, $Task_{id}$ the ID of the task, $Task_{len,}$ is million instructions per second(MIPS), $Task_{pes}$ is the processing unit (PE) of the task, $Task_{bdwidth}$ is the bandwidth (unit is MB) of current network, $Task_{storage}$ to store the task (unit is MB), $Task_{input}$, $Task_{output}$ are size of input and output files respectively. In addition, consider 'M' virtual machines with variable performance aspects at the data centers and each virtual machine has the following properties defining its capabilities as below.

$$VM_j = \begin{Bmatrix} VM_{id,} VM_{pesnum,} VM_{mips,} \\ VM_{bdwidth,} VM_{size,} \end{Bmatrix} \quad (2)$$

Where j=1,2,..,M and $VM_{id,}$ is the ID of the resource, VM_{pes} is the quantities of CPU, VM_{mips} is million instructions per second (MIPS) of a CPU, $VM_{bdwidth}$ is the bandwidth and VM_{size} is the storage capacity of VM_j.

Therefore, the Expected computing time considered for $Task_i$, i=1,2,..N requests on VM_j, j=1,2,..M, number of virtual machines is given as ECT and Task scheduler engages in proper task scheduling decisions. Basically, the ECT of a $Task_i$ on VM_j and makespan is given as below:

$$ECT = \frac{Task_{len}}{VM_{mips} * VM_{pesnum}} \quad (3)$$

$$MST = max \sum_{j=1}^{N} ECT_{ij} \quad (4)$$

Our approach promises to attain the objective function improving the overall performance with minimized makespan time (*MST*), herewith the objective function is given as follows:

$$O_{obj\_fun} = \min(MST) \quad (5)$$

4. Methodology

Our approach initiates with a pre-processing mechanism facilitating efficient and accurate results.

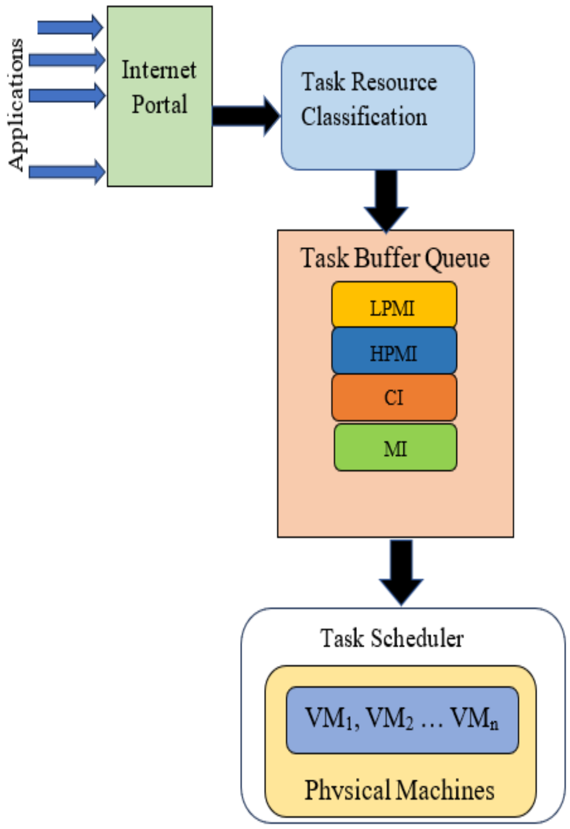

Figure. 1 Flowchart of proposed method

The processed data pertaining to user tasks are classified into groups of similar characteristics, followed by efficient mapping of task groups to relevant VM groups using an enhanced coot optimization algorithm (ECOA-TS). This algorithm accelerates for finding best solutions by improving the convergent evolution of the nearest optimal solutions using Cauchy mutation to prevent from entering local optimization and balance its exploration and development capabilities. Fig. 1 describes the flow of proposed work.

4.1 Dataset analysis

The Google cluster traces [25] containing cell information of around 29 days, were made available by Google in May 2019, where each cell is made up of a collection of machines that use the same cluster management system.

The proposed work considers job-id, task-id, status, plan-CPU, plan-memory, real-CPU, real-memory, average-CPU, average- memory, start and finish timestamps of tasks. The jobs with finished status are considered for analysis purposes. As memory and CPU are key factors in determining resource usage, this study takes into account the average CPU in core, normalized average memory, and job execution time in seconds.

Received: May 19, 2023. Revised: July 25, 2023.

```
Algorithm 1: Data cleansing and normalization
Input: N as the input
Output: Cleaned and Normalized data
1:    Initialize k=0
2:    While k<=Sizeof (N),
3:    If nan (Nk), then use the function nan to
      verify the vector items.
4:    Approximate (Nk)
      \* Calculate the average of the
      neighboring points to estimate the missed
      value. * /
5:    End if
6:    FindMaxMaxk (Nk)
7:    Dk Normalize (Nk, Maxk)
8:    End While.
```

```
Algorithm 2: Improved density based clustering
method to classify the tasks
Input: Google Cluster Traces.
Output: Clustered Tasks.
• Compute local density and distance levels
  of sample data and arrange them in
  descending sequence.
• Consider the x sample bearing the largest
  density value satisfying neighbourhood
  distance with min_pts criteria, and
  compute $\emptyset_{avg}$.
• Finally, the values with $\emptyset_i > \emptyset_{avg}$ are
  designated as cluster centers that aid in
  the formation of the decision graph.
• Execute the Fuzzy C Means for Task
  Clustering.
• Let D ={d1,d2,d3…dn} be the set of data
  points and C={c1,c2,c3…..Cc} the
  cluster centres selected randomly.
• Membership functions are computed using
  Eq. (8), Fuzzy centres computed by using
  Eq. (9).
• Iterate above two steps until the minimal
  value of objective function in Eq. (7) is
  attained.
```

4.2 Pre-processing

Data collected is subjected to data cleaning, and filtering activities aiding in accurate results. The tuples with null values are removed in the first phase. In the second stage, only tasks with a finished state are filtered for this work. The third phase involves choosing attributes for CPU, RAM, and execution time whose values are finally normalized before being used in the clustering method.

4.3 Phase I: Task classification by improved density based clustering method

Clustering ascertains the number of classes and the respective membership points of each class, generating clustered labelled data. Fuzzy enhances the clustering approach in our work by depicting the possibility of assigning data points to more than one cluster by calculating the membership function belonging to available classes.

Improved density based clustering method overcomes the sensitivity to initial clusters that keep updating until optimal clustering, which adds to initially fix the number of cluster centers for reducing the number of iterations, and to form initial clusters for improved convergence. The density peek (DP) algorithm ascertains improvement by initially locating each cluster centres with parameters $\emptyset_i$ and δ_i of the considered data set.

$$\emptyset_i = \alpha_i \delta_i \tag{5}$$

Compute the value of *i* for each data point through an iterative sampling of data points. Initial z points are retrieved by arranging the distances in descending order. The average density distance is formulated as:

$$\frac{1}{x}\sum_{i=1}^{x}\emptyset_i \tag{6}$$

The higher the density distance values, the higher the probability of being the cluster centre i.e., the chance of maximum value aids in dictating the cluster centre. Thus, this cluster centre is picked as input to the second phase.

Consider the dataset D = {D1, D2, D3…Dn}, where Dn is a set of n samples tuple exhibits p features. These samples are used to generate C fuzzy groups. The FCM's objective function is formulated as:

$$J(F,Y) = \sum_{i=0}^{s}\sum_{i=0}^{n} f_{ij}^{m} \times dis_{ij}^{2} \tag{7}$$

where f_{ij} is an (n × s) membership matrix, and dis_{ij} is the Inverse Euclidean distance of the sample j and the cluster center *i*.

The minimum objective function $J(F,Y)$ of the respective FCM approach is derived by computing the *F* and *Y* values using the below formulation:

$$\mathrm{f}_{ij} = \frac{1}{\sum_{\mathrm{k}=1}^{\mathrm{s}} \frac{dis_{ij}}{dis_{kj}}^{\frac{1}{m-2}}} \tag{8}$$

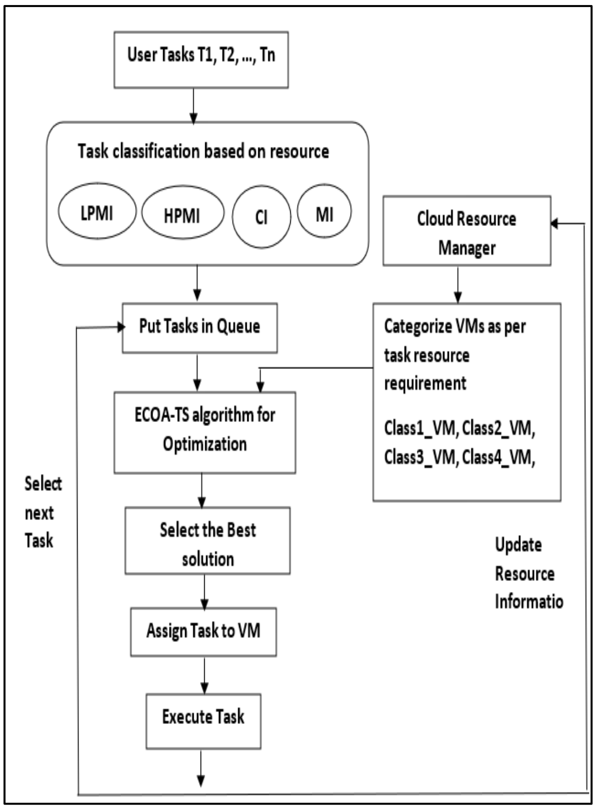
User Tasks T1, T2, ..., Tn
Task classification based on resource
LPMI
HPMI
CI
MI
Cloud Resource Manager
Put Tasks in Queue
Categorize VMs as per task resource requirement
Class1_VM, Class2_VM, Class3_VM, Class4_VM,
ECOA-TS algorithm for Optimization
Select next Task
Select the Best solution
Update Resource Informatio
Assign Task to VM
Execute Task

Figure. 2 Architecture of proposed work

$$\mathrm{Y_i} = \frac{\sum_{a=1}^{n} (\mathrm{f}_{ij})^m x_i}{\sum_{a=1}^{n} (\mathrm{f}_{ij})^m} \quad (9)$$

The Novelty is ascertained in aspect that the density peak ensures efficient clustering by tunning parameters like neighbourhood radius and min-points criteria. In addition, Fuzzy c means clustering pertaining to inverse Euclidean distance promises lowered sum of squared error (SSE). The resultant clusters are categorized into four buffer groups pertain to the low processor- memory intensive group (LPMI), high processor-memory intensive group (HPMI), processor intensive group (PI), and the memory intensive group (MI). The LPMI queues define tasks that have low memory and CPU needs. Tasks with heavy needs are placed in the HPMI queue, and higher CPU requirements and higher Memory requirements are pipelined to PI and MI queues respectively.

The VMs are divided into four major classes class-1, class-2, class-3, and class-4 as per the task resource requirement. The scheduling performance is herewith enhanced by mapping appropriate tasks from each task buffer queue to their optimal VM type by using an enhanced coot optimization algorithm. The following Fig. 2 describes the architecture of the proposed work.

4.4 Phase II: Enhanced coot optimization algorithm for optimal task scheduling (ECOA-TS)

Optimization algorithms aim at reducing the complexity and search space resulting in an improved and enhanced solution [26, 27]. They tend to evaluate the possible solutions against each other, that better fit into the objective function [28]. Our work focuses on improving makespan time inspired by Coot birds' natural behavior that overrides undesirable features and gives efficient results.

Enhanced coot optimization algorithm:

A Coot bird is characterized to be a small water bird [29] with variable movements and behavioural aspects that make it an optimal choice. Our work relies on the behaviour of coots that tend to cover distances with several directions of motion. The major movements imaging its behaviour are Random motion to this and that direction, a chain movement, adjusting position in accordance with the group leaders and leader movement.

Eq. (10) aids in randomly generating coot population by considering a small sample area.

$$CPos(i) = rand(1, dim) \times (u_{bd} - l_{bd}) + l_{bd} \quad (10)$$

Where $CPos\,(i)$ is the coot position, dim is the number of problem dimensions, l_{bd} , u_{bd} are the lower and upper limits of the search space that is defined below.

$$l_{bd} = [\, l_{bd1}, l_{bd2}, l_{bd3}, \dots, l_{bdn}]$$
$$u_{bd} = [\, u_{bd1}, u_{bd2}, u_{bd3}, \dots, u_{bdn}] \quad (11)$$

Random movement to this and that direction:

Various movements related to coot migration in exploring search space are considered. The movement of the coot bird will permit to avoid converging at the local optima or getting looped in the local optimal. Coot's updated position is computed in Eq. (13).

$$\mathrm{Z} = rand(1, dim) \times (u_{bd} - l_{bd}) + l_{bd} \quad (12)$$

$$CPos(i) = CPos(i) + G \times K2 \times (Z - CPos(i)) \quad (13)$$

In the above equation $K2$ denotes a random number generated using tent chaotic function accordingly and G is computed as below.

$$G = 1 - Iter \times \left(\frac{1}{M_{Iter}}\right) \quad (14)$$

Received: May 19, 2023. Revised: July 25, 2023.

Where $Iter$ is the current iteration, M_{Iter} is the maximum iteration.

Tent map: Tent chaotic map is defined as a logistic map that visualizes certain chaotic effects. It is formulating as below.

$$s_{r+1} = \begin{cases} 2sr, sr < 0.5 \\ 2(1 - s_r), sr >= 0.5 \end{cases} \tag{15}$$

Chain movement: The explored paths are connected to form a chain movement using the mean positions of both coots, which is formulated in Eq. (16) and is used to calculate the coot's new position, where CPos$(i - 1)$ represents the second coot.

$$CPos(i) = 0.5 \times (\ CPos(i-1) + CPos(i)) \tag{16}$$

Adjusting position in accordance with the group leaders:

Each coot updates its position by considering average position of Leaders and below Eq. (17) used to choose the Leader.

$$E = 1 + (iMODL_c) \tag{17}$$

Where i represents the current coot index, L_c , E represents the count and index of Leaders.

Compute the position of next coot with respective selected Leader by using Eq. (18)

$$CPos(i) = (L_{pos}(E) + 2 \times K1 \times cos(2K\pi) \times (L_{pos}(E) - CPos(i) \tag{18}$$

$CPos(i)$ is Coot current location,$L_{pos}(E)$is chosen Leader position, π =3.14, and $K1$, K are random values in interval [0,1], [-1,1] respectively.

Leader movement: Leader positions are updated with the formula provided below by searching around optimal points and the selected leaders keep changing their location to move the group of followers toward the ideal region.

$$L_{pos}(i) = \begin{cases} H * K3 \times cos(2K\pi) \times \left(Best_g - L_{pos}(i)\right) \\ \quad +Best_g K4 < 0.5 \\ H * K3 \times cos(2K\pi) \times \left(Best_g - L_{pos}(i)\right) \\ \quad -Best_g K4 \geq 0.5 \end{cases} \tag{19}$$

Where $Best_g$ is the optimal position attained, $K3$ and $K4$ indicate random values between [0, 1], K, a random number ranging between [-1,1] intervals. H is formulated and computed as below.

$$H = 2 - Iter \times \left(\frac{1}{M_{Iter}}\right) \tag{20}$$

Cauchy mutation:

It is designated as an optimization tool that utilizes the Cauchy distribution to introduce variations in actual positions thereby exploring local neighborhood. Its density function is given below:

$$f_t(y) = \frac{t}{\pi(t^2+y^2)}, -\infty < y < +\infty \tag{21}$$

where t > 0 is the scale factor, and the Cauchy distribution is formulated as follows:

$$f(y) = \frac{1}{2} + \frac{1}{\pi} \arctan\left(\frac{y}{t}\right) \tag{22}$$

The Cauchy exhibits similarity with density function of Gaussian distribution entailing few notable differences. The Cauchy distribution manifests infinite variance promoting notable jumps for better transition among discrete environments, thereby avoiding the trap of local optima.

The proposed work employes a Cauchy distribution with y value as 0 and t as 0.5 as it promises optimal outcomes around these dimensions. Our approach overrides the pitfalls of Coot algorithm pertaining to local sub-optimal convergence and lowered accuracy by exploring a novel search position resulting from Cauchy distribution function over two randomly selected coot positions.

$$Opt = Cauchy\left(Y^N_{rand,1} - Y^N_{rand,2}\right) \tag{23}$$

Hereby the Cauchy mutation utilizes Cauchy distribution through Eq. (23).

$Y^N_{rand,1}$ and $Y^N_{rand,2}$ represent randomly selected coot positions from the population. The cauchy variational operator Opt encompasses the aforementioned components.

5. Results and discussion

Experimental setup and simulation

The proposed algorithm is evaluated for efficiency using the CloudSim simulation tool that better depicts our approach providing accurate results. The simulator succeeds in emulating the cloud environment components in an enhanced way, and also renders support to various well-versed scheduling strategies. Our work simulates cloud models based on a single data center, with infrastructure as service (IaaS).

Algorithm3: Enhanced coot optimization algorithm for task scheduling (ECOA-TS)

```
Input: Randomly initialize coot population as set of Tasks and VMs.
Output: Optimally mapped Tasks and VMs.

• Initialize the coot population by Eq (10), parameters Pr=0.5, $L_{cnt}$ (Number of Leaders), $Coot_{count}$ (Number of Coots).
• $Coot_{count}=Coot_{pop}-L_{cnt}$.
• Compute the fitness values of coots and leaders,Identify the best coot or leader as global optimum (Bestg).
• While End criteria not satisfied
    Compute G, H by using Eq. (14) and Eq. (20)
    If rand<Pr
        K, K1, and K3 are randomly selected dimensions.
    Else If
        K, K1 and K3 are a random number
    End If
• For x=1 to $Coot_{count}$
    Calculate parameter $E$ by Eq. (17).
    If rand >0.5
        Update coot location by Eq. (18).
    Else If rand <0.5~=1
        Update coot location by Eq. (16).
    Else
        Update coot location by Eq. (13)
        In Eq. (13), random number K2
        is computed using the Tent map as per proposed model.
    End If
• Compute Coot Fitness using Eq. (5)
    If Coot Fitness < Leader Fitness
        Swap (Coot, Leader).
    End If
  End For
• For Leaders count
    If K4 < 0.5
        Update Leader position by Eq. (19.1)
    Else
        Update Leader position by Eq. (19.2)
    End If
• If Leader Fitness< $Best_g$
    Swap (leader, $Best_g$) and revise global optimum.
  End If
 End For
• Perform Cauchy mutation for global search
  $H^t=H^t+1$
 End while.
```

Table 1. Configuration of host in datacenter

Host Parameters	Value
Processing element (PE)	2-10
Processing capacity	20000-35000 MIPS
RAM capacity	8Gb,16GB and 32GB

Table 2. Configuration of VMs

VM Parameters	Value
Pes in each VM	1
Processing capacity	500-4000 MIPS
RAM capacity	512-4196 MB

Table 3. Task parameters

Task Parameters	Value
Length of Task	15000-900000MI
Size of Task	60-3000KB

The system configurations supporting the simulation are Intel core i5 CPU with 1.80 GHz processor, RAM 8 GB, Windows 64-bit, and Windows 10 operating system. The configuration parameters of simulated cloud data are shown below.

5.1 Performance analysis

The metric that provides efficient monitoring of cloud components is performance, that ought to be quantifiable and compatible with objectives for the scheduling problem's performance. The ultimate goal of scheduling algorithms is to keep the makespan of the jobs to a minimum value for improved performance. Using varied numbers of VMs and jobs, three different experiments are carried out. The VM count is kept at values 20,30 and 50, with task count at intervals of 500. The resultant outcomes spotted with varied spikes in graphs for tasks and resources are depicted. Both the tasks and the resources in this situation display varied traits.

For evaluation and comparison analysis, three existing approaches including particle swarm optimization (PSO), grey wolf optimization (GWO), and whale optimization algorithm (WOA), are

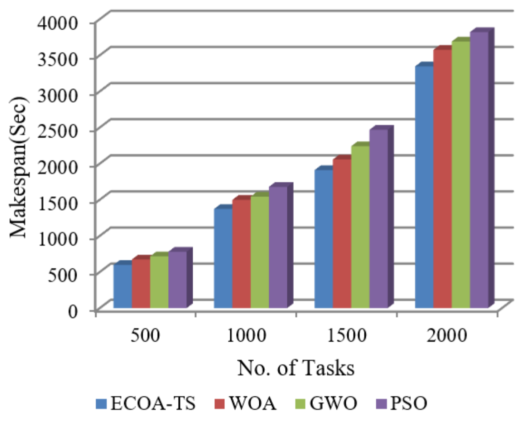

Figure. 3 No. of Tasks vs Makespan(sec) with 20 VMs

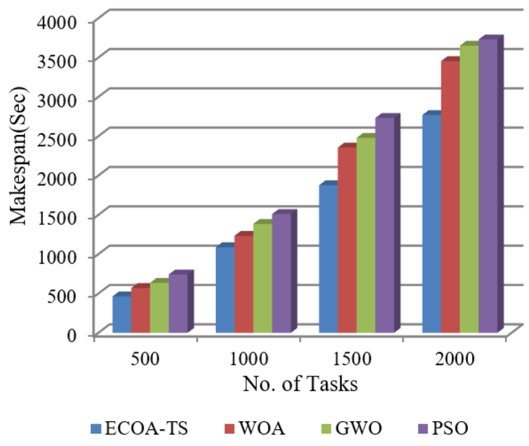

Figure. 4 No. of Tasks vs Makespan(sec) with 30 VMs

considered as peer methods for our work. The considered algorithms are individually executed over google clustered traces and its performance evaluations. In Fig. 3 ECOA-TS method has demonstrated a reduction MST (Makespan Time) for several counts of task ranges with 20 VMs. For instance, with 500 tasks, the ECOA-TS approach achieved a minimal MST of 598 but the PSO, GWO and WOA algorithms presented high values of MST of 780, 716, and 672 respectively. Continued with, 1000 tasks, the ECOA-TS approach presented reduced MST of 1372 forasmuch the PSO, GWO, and WOA algorithms have computed larger values of MST of 1678, 1543, and 1498 respectively. In addition to the aforementioned, when given 2000 tasks, the ECOA-TS algorithm produced a lower MST of 3346, but the PSO, GWO, and WOA techniques produced higher MSTs of 3820, 3690, and 3575, respectively.

Fig. 4 illustrates the performance comparison of proposed work over the existing approaches for several counts of task ranges with 30 VMs. For instance, with 500 tasks, the ECOA-TS approach

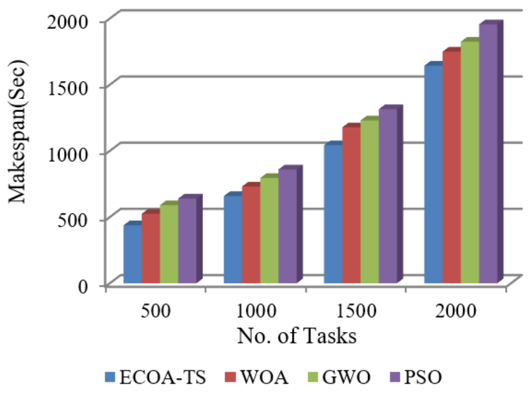

Figure. 5 No. of Tasks vs Makespan(sec) with 50 VMs

offered minimal MST of 463 but the PSO, GWO and WOA algorithms presented high values of MST of 742, 635, and 570 respectively. The PSO, GWO, and WOA algorithms computed higher values of MST of 1510, 1385 and 1234, respectively, whereas the ECOA-TS technique exhibited a lower MST of 1089 after 1000 tasks. Apart from above, with 2000 tasks, the proposed algorithm rendered lowered MST of 2769 where the PSO, GWO and WOA methods imaged higher MST of 3730, 3650, and 3455 respectively.

The results depicted in Fig. 5 proposed method has promised to offer reduced MST for several counts of task ranges with 50 VMs. For instance, with 500 tasks, the proposed method offered minimal MST of 438 but the PSO, GWO and WOA algorithms presented high values of MST of 640, 590, and 525 respectively. Continued with, 1500 tasks, the ECOA-TS approach presented reduced MST of 1043 sec forasmuch the PSO, GWO and WOA algorithms have computed larger values of MST of 1315, 1230 and 1178 respectively. Apart from above, with 2000 tasks, the ECOA-TS algorithm rendered lowered MST of 1644 where the PSO, GWO and WOA methods imaged higher MST of 1956,1825, and 1750 respectively.

Fitness analysis: The behavioral performance of our proposed algorithm is evaluated by plotting curves for fitness value against other comparison algorithms for a variable count of iterations. Fig. 6 provide the fitness function evaluation of PSO, GWO, WOA and proposed method for variable permutations. The profound outputs shows that ECOA-TS method has occupied the optimal outcome with optimal fitness value in all iterations. By way of illusion, when 100 iterations, the ECOA-TS offered a reduced fitness value of 187.46 with PSO, GWO and WOA offering a fewer improvement in fitness value of 200.2,198.56,196.38. Consecutively, with 500 iterations, the ECOA-TS approach reflected reduced

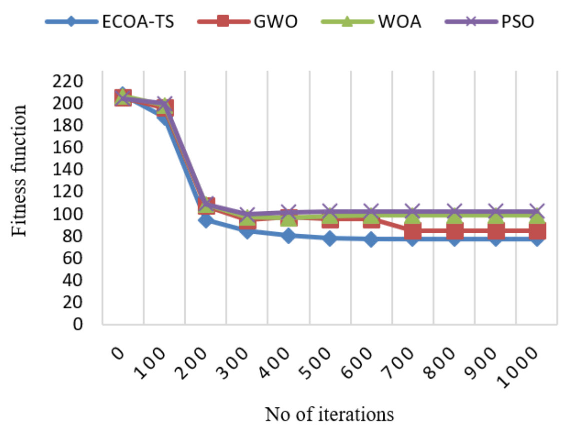

Figure. 6 Fitness function over iterations

fitness value of 78.38 and the PSO, GWO, WOA attained higher value of 102.78,98.56, 95.84. Similarly, with 1000 iterations, the ECOA-TS approach attained minimum fitness value of 77.66 in contrast to PSO, GWO, WOA with fitness value of 102.78,98.56, 95.84.

In a nut shell our Enhanced coot optimization algorithm for task scheduling (ECOA-TS) succeeds in improving the performance by imaging the experimental results in comparison to mentioned approaches generating better makespan time for datasets considered.

6. Conclusion

Our work explores a meta-heuristic framework for task scheduling on cloud platforms that guarantees optimal results. The realistic dataset considered in our work pertains to Google cluster traces. The framework originates with pre-processed data pipelined into the classification process using the improved density clustering method (IDCM) algorithm where user tasks are categorized related to resource needs and execution time. The resultant outcome of cluster groups flows into queues designated as low processing memory intensive (LPMI), high processing memory intensive (HPMI), processor intensive (PI), and memory intensive (MI) tasks. The tasks moving out of the queue are assigned the appropriate VMs on the respective physical machines using proposed approach. Experimental outcomes of the proposed ECOA-TS framework delectably override the compared approaches by lowering average makespan time by 15.33%,19.8%, 27.41%, over GWO, WOA and PSO algorithm respectively. The miniature of our work is materialized and simulated using CloudSim. Future Scope of the work can be accelerated by considering the dynamic nature of user tasks, where the tasks keep reshaping their resource needs either by leveraging their requirements or pulling down their needs dynamically.

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

The first author has done investigation, dataset collection, implementation, result analysis, and preparing original draft. The second author has done supervision, review of work and validation.

References

[1] A. H. A. Halim and A. I. Hajamydeen, "Task Scheduling Management for Load Balancing Using Task Grouping Based on Cloud Computing", *Asian Journal of Computer and Information Systems*, Vol. 9, No. 3, 2021.

[2] E. H. Houssein, A. G. Gad, Y. M. Wazery, and P. N. Suganthan, "Task Scheduling in Cloud Computing based on Meta-heuristics: Review, Taxonomy, Open Challenges, and Future Trends", *Swarm and Evolutionary Computation*, Vol. 62, p. 100841, 2021.

[3] S. Mohanty, P. K. Patra, M. Ray, and S. Mohapatra, "A Novel Meta-Heuristic Approach for Load Balancing in Cloud Computing", *Research Anthology on Architectures, Frameworks, and Integration Strategies for Distributed and Cloud Computing*, pp. 504–526, 2021.

[4] M. B. Gawali and S. K. Shinde, "Task scheduling and resource allocation in cloud computing using a heuristic approach", *Journal of Cloud Computing*, Vol. 7, No. 1, 2018.

[5] R. Kaur, V. Laxmi, and Balkrishan, "Performance evaluation of task scheduling algorithms in virtual cloud environment to minimize makespan", *International Journal of Information Technology*, Vol. 14, No.6, pp. 79–93, 2022.

[6] 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.

[7] S. Borah and S. K. Mishra, "Scheduling Tasks in Virtual Machines Using Ant Colony Optimization Technique", *AI in Manufacturing and Green Technology*, pp. 11–18, 2020.

[8] A. M. S. Kumar, K. Parthiban, and S. S. Shankar, "An efficient task scheduling in a cloud

computing environment using hybrid Genetic Algorithm - Particle Swarm Optimization (GA-PSO) algorithm", In: *Proc. of International Conf. on Intelligent Sustainable Systems (ICISS)*, pp. 29-34, 2019.

[9] A. K. Maurya, "Resource and Task Clustering based Scheduling Algorithm for Workflow Applications in Cloud Computing Environment", In: *Proc. of International Conf. on Parallel, Distributed and Grid Computing (PDGC)*, pp. 566-570, 2020.

[10] W. Khallouli and J. Huang, "Cluster resource scheduling in cloud computing: literature review and research challenges", *The Journal of Supercomputing*, Vol. 78, No. 5, pp. 6898–6943, 2021.

[11] S. Kanwal, Z. Iqbal, F. A. Turjman, A. Irtaza, and M. A. Khan, "Multiphase fault tolerance genetic algorithm for vm and task scheduling in datacenter", *Information Processing & Management*, Vol. 58, No. 5, p. 102676, 2021.

[12] S. P. Praveen, H. Ghasempoor, N. Shahabi, and F. Izanloo, "A Hybrid Gravitational Emulation Local Search-Based Algorithm for Task Scheduling in Cloud Computing", *Mathematical Problems in Engineering*, Vol. 2023, pp. 1–9, 2023.

[13] 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.

[14] P. Pirozmand, H. Jalalinejad, Hosseinabadi, A. R. Asghar, M. Seyedsaeid, and Y. Li, "An improved particle swarm optimization algorithm for task scheduling in cloud computing", *Journal of Ambient Intelligence and Humanized Computing*, Vol. 14, pp. 4313–4327 ,2023.

[15] 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.

[16] A. M. Zadeh, M. Masdari, F. S. G. Chopogh, and A. Jafarian, "Improved chaotic binary grey wolf optimization algorithm for workflow scheduling in green cloud computing", *Evolutionary Intelligence*, Vol. 14, No. 4, pp. 1997–2025, 2020.

[17] H. Liu, "Research on cloud computing adaptive task scheduling based on ant colony algorithm", *Optik*, Vol. 258, p. 168677, 2022.

[18] B. M. H. Zade, N. Mansouri, and M. M. Javidi, "A two-stage scheduler based on New Caledonian Crow Learning Algorithm and reinforcement learning strategy for cloud environment", *Journal of Network and Computer Applications*, Vol. 202, p. 103385, 2022.

[19] S. Mangalampalli, S. K. Swain, and V. K. Mangalampalli, "Prioritized Energy Efficient Task Scheduling Algorithm in Cloud Computing Using Whale Optimization Algorithm", *Wireless Personal Communications*, Vol. 126, No. 3, pp. 2231–2247, 2021.

[20] L. Jia, K. Li, and X. Shi, "Cloud Computing Task Scheduling Model Based on Improved Whale Optimization Algorithm", *Wireless Communications and Mobile Computing*, Vol. 2021, pp. 1–13, 2021.

[21] J. Li, T. Ma, M. Tang, W. Shen, and Y. Jin, "Improved FIFO Scheduling Algorithm Based on Fuzzy Clustering in Cloud Computing", *Information*, Vol. 8, No. 1, p. 25, 2017.

[22] A. Jivrajani, D. Raghu, A. KH, H. L. Phalachandra, and D. Sitaram, "Workload Characterization and Green Scheduling on Heterogeneous Clusters", *22nd Annual In: Proc. of International Conf. On Advanced Computing and Communication (ADCOM)*, 2016.

[23] J. Li and Y. Han, "A hybrid multi-objective artificial bee colony algorithm for flexible task scheduling problems in cloud computing system", *Cluster Computing*, Vol. 23, No. 4, pp. 2483–2499, 2019.

[24] S. Mangalampalli, S. K. Swain, G. R. Karri, and S. Mishra, "SLA Aware Task-Scheduling Algorithm in Cloud Computing Using Whale Optimization Algorithm", *Scientific Programming*, Vol. 2023, pp. 1–11, 2023.

[25] Datasets:https://research.google/tools/datasets/google-cluster-workload-traces-2019/

[26] P. D. Kusuma and M. Kallista, "Stochastic Komodo Algorithm", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 4, 2022, doi: 10.22266/ijies2022.0831.15.

[27] P. D. Kusuma and M. Kallista, "Quad Tournament Optimizer: A Novel Metaheuristic Based on Tournament Among Four Strategies", *International Journal of Intelligent Engineering and Systems*, Vol. 16, No. 2, 2023, doi: 10.22266/ijies2023.0430.22.

[28] S. M. Menon and P. Rajarajeswari, "A Hybrid Machine Learning approach for Drug Repositioning", *International Journal of Computer Science and Network Security*, Vol. 20, No.12, pp. 217-223, 2020.

[29] I. Naruei and F. Keynia, "A new optimization method based on COOT bird natural life model", *Expert Systems with Applications*, Vol. 183, p. 115352, Nov. 2021.