International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Big Data Advanced Scheduling Integrating Q-Learning and Cuckoo Search

Nagina[1]*, Dr. Sunita Dhingra[2]

Submitted: 23/09/2023 **Revised**: 17/11/2023 **Accepted**: 29/11/2023

Abstract: As a well-known fact sensor nodes are collecting enormous amounts of data in real-time, task scheduling is a crucial problem in big data. It is essential in determining the effectiveness and performance of networks. The current methods only address the node angle that degrades the system's performance. This research presents two distinct frameworks to which a flexible Q-learning technique has been offered. The first scheduling architecture selects the suitable data node based on buffer latency using a Q-learning algorithm. In the second, feed forward neural networks are used to train the system while the reward mechanism is employed. To maintain the best records, the Advanced Cuckoo Search (A-CS) technique, a form of swarm intelligence, has been employed in conjunction with the K-means clustering algorithm to determine the centroid. Accuracy is 93.25%, True Positive Rate (TPR) is 0.93, and False Positive Rate (FPR) is 0.07. These performance metrics have been measured. The results indicate that the accuracy of the suggested strategy has increased by 6%, 5%, and 12%, respectively, when compared to the random forest, naïve bayes, and multi-class support vector machine.

Keywords: *Cloud Computing, Task Scheduling, Big Data, Machine Learning, Advanced Cuckoo search, Feed Forward Neural Network (FFNN).*

1. Introduction

The exponential growth in the use of internet technologies, Cloud Computing (CC) emerges as a scalable network which becomes very much useful for various applications in both industry and businesses. Cloud computing is a pool of various resources like storage units, processing unit, and other networking devices, etc. CC has the distinct features of grid computing, and utility computing but the concept is quite different from the resource virtualization. Apart from that, there are various important features of CC such as scalability, resource utilization, economic use, and its ubiquitous properties. However, users need the internet access to access the various facilities of the CC. The user can easily use the resource and pay when the service is fully utilized. Thus, massive volumes of data have been gathered by the sensor nodes in a real time environment. This data fall has been fallen under the category of Big Data that possesses volume, variety, and velocity of data transmission. The combination of these factors became difficult to handle such data [1]. Thus, innovative techniques and methods employed in the literature to handle the big data [2]. Scheduling is a technique used to allocate the tasks to the Virtual Machines (VM's) for processing[3]. In Big data, task scheduling is a difficult problem and can be optimized by applying several optimal solutions [4, 5]. Thus, scheduling of massive volume of data poses a lot of problems to the efficient computing that lead to longer make span time and large cost that hinders the system performance. The process requires cloud computing tools [6]. The costs incurred on both server Physical Machines (PM's) and users end make the data transmission costly and lengthy. But, efficient scheduling of tasks always needed for the efficient utilization of historical information and resources [7]. Thus, it does not only maintain the make span time but also manages the Service Level Agreement (SLA) to provide better service to the associated users by maintaining good Quality of Service (QoS).

Specifically, MapReduce programming also used to process the Big Data using the software such as Apache Spark or Hadoop. But there are certain problems associated when using this technique, make it unsuitable for the processing the Big Data in the cloud platform [8]. It is noted that fast response of the cloud is one of the key features that falls shortened by the MapReduce paradigm. Thus, it also falls to timely generate a response due to the key's generation in the process of map phase. MapReduce also fails to work with the complex classifiers such as Support Vector Machine (SVM) and other Machine Learning (ML) techniques due to longer compilation time. However, many algorithms were developed to tackle the problems as aforementioned. Ding proposed a Q-learning based two phase frameworks for efficient scheduling the tasks in the cloud environment. In the developed framework, the former phase used to assign the tasks using

[1] *Research Scholar, Department of Computer Science and Engineering, University Institute of Engineering and Technology, Maharshi Dayanand University, Rohtak, Haryana, India. nagina260@gmail.com*
[2] *Associate Professor, Department of Computer Science and Engineering, University Institute of Engineering and Technology, Maharshi Dayanand University, Rohtak, Haryana, India. sunitadhingra@rediffmail.com*
** Corresponding Author Email: nagina260@gmail.com*

the centralized dispatcher and then request has been pushed on a global queue. The later phase used at VM level in which Q-learning based scheduler has been used to process the various tasks. All the users' requests have been prioritized by applying incentives to the tasks based on different rewards. This also minimizes the response time and maximizes the utilization of CPU [9]. Thus, this also avoids the wastage of resources and makes span time. The advantages of Q-leaning based technique to schedule the task adapted to the environment and can easily benefit the system. But still system complexity increases and effectiveness of the system reduces due to the reward given to the agent actions in a complicated manner. The other deficiencies demonstrated in the previous research are: -

The past techniques focused only on the nodes perspectives as nodes only take actions to achieve the maximum rewards. The overall task scheduling is not ideal for this action as such a perspective not applicable but some approaches used to allocate the task considering a global view [10].

It is difficult to distinguish the nodes and thus same incentive strategy has been applied to all the nodes. Thus, there is a need to define the more operations in the Q-learning model.

Therefore, to avoid these issues, classification algorithms have been used in the literature to schedule the task in a big data. Vashishth used the classification techniques to surge the efficiency and system reliability while managing the Big Data in the cloud environment. The task scheduling has been done using the Machine Leaning and then VM's used for assignment purposes in the cloud environment. Further, Particle Swarm Optimization (PSO) also used effectively to reduce the overhead time and classify the task allocation technique [11].

The given Figure 1 illustrates that ML used as centered component interlinked with the four other components that includes Big Data, Customers or User, Domain, and linked system. The interconnection or communication between the devices is bidirectional. The learning engine is updated utilizing the current information that has been extracted from data. The engine processes the user query based on a repository that has been prepared during the training time.

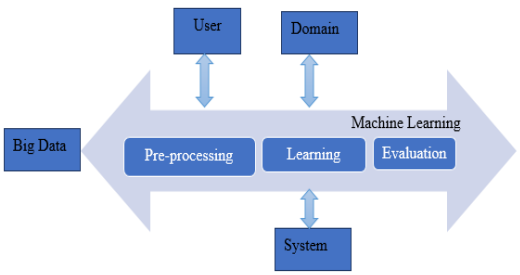

Fig. 1. A Framework to Manage the Big Data using the Machine Learning

Specifically, Swarm Intelligence (SI) techniques have been used in conjunction with the ML to improve the system reliability and reducing the execution time [12]. To guide the Cc so that to select the appropriate scheduling technique, a multi-criteria decision has been made to optimize the system performance. The scheduling has been done using the ABC and PSO technique to determine the makes pan time and managing the task in three different data centers. The outcomes are fruitful but unable to determine the CPU capacity, bandwidth utilization and usage of RAM. Moreover, system complexity also increases in simulating the different tasks as per user requirement. Jaber developed the modified Cuckoo Search technique to reduce the processing cost and solve the scheduling problems [13]. The outcomes are fruitful in reducing the makes pan time but limited to provide the desired results in terms of efficiency and accuracy. This paper aimed to solve these issues by reducing the computation complexity of the simulated big data. The proposed solution is divided into two parts namely the development of scheduling architecture using the Q-learning algorithm. The proposed algorithm is used for the selection of appropriate data node based on the buffer delay. The second part utilizes the reward mechanism of the Q-learning based technique and trains the system using Feed Forward Neural Networks for the training of the system.

The paper is organized as section 1 demonstrates the introduction part in which different task scheduling techniques in Big Data have been explained. This section also discusses the various issues related to previous Q-learning techniques and then SI techniques have been discussed. In section 2, the background in which previous techniques had been developed for task scheduling is explained. Section 3 highlights the proposed work in which Q learning technique and SI based Cuckoo Search technique applied to optimize the various tasks. Further, results have been discussed to compute the performance metrics in terms of TPR, FPR, and accuracy and, paper finally concluded in the last section.

2. Related Work

In the literature, different state of art techniques has been proposed to schedule the task in the big data. Specifically, technique that are based on Q-learning, and Machine learning gaining an attention due to the promising solutions with better performance [14, 15].

Jena 2015 proposed a multi-objective task scheduling framework using the nested PSO. The proposed solution was not only managing the processing time but also optimize the energy consumption. The simulated results had been developed using the Cloud Sim toolkit in which objective function computed and then algorithm was

implemented to compute the mutation operator. Further, experimental results have been compared to validate the proposed solution in terms of time, energy, and failed task. The outcomes are promising but limited to consider the other objectives such as Bandwidth and overall cost of the system. Moreover, there is also need to focus more on the robustness of the algorithm [16].

Tian et al. 2020 proposed a hybrid task scheduling algorithm considering the task clustering mechanism. The proposed solution integrates the tasks and then task duplication process was done to figure the execution cost of the process. The outcomes show that developed algorithm not only reduces the communication overhead but also manages the successor tasks. The experimental results further compared with the leading algorithms to evaluate the superiority of the task scheduling algorithm [17].

Alhubaishy & Aljuhani 2020 developed a model based on the preferences of the customers and predefined criteria. For this, the authors had been adopted the best-worst method to structure the task scheduling problems. The proposed study was provided reliable results by accommodating the customer preferences and thus, deals the task scheduling problem dynamically. The outcomes were consistent but still other problems such as cost and bandwidth still are the constraints that reduce the applicability of the algorithm [18].

Rjoub et al. 2020 proposed a big data task scheduling approach by understanding the problems of VM trust level and resource consumption issues. The authors developed a Big Trust Scheduling solution that worked in three different phases: Computing the VM trust level, prioritization of tasks, and scheduling the tasks based on trusts. The experiments were performed using the real-world dataset using the Google Cloud Platform and results show that proposed solution minimize the make span time by 59% in comparison to PSO and other existing techniques. The developed solution was promising and applicable to other task scheduling problems. Moreover, the trust model also extended to other environments such as IoT, Fog computing etc [19].

Ding et al. 2020 proposed a task assignment solution to avoid the energy consumption issue in an analytical and experimental manner. The Q-learning based solution had been worked in two different phases in which first phase used to design the centralized task dispatcher in combination with the queuing model. The second phase based on prioritizing the tasks to minimize the task life time and laxity. The shorter response time and other energy constraints were desirable that improves the system performance. Further, SLA constraints were considered using the dynamic task ordering strategy and thus enhance the cloud services quality. The proposed solution also designed the adaptive scheduler to reduce the response time and thus, maximize the CPU utilization. The proposed model is robust but increase in complexity makes the system inefficient [20].

Tong et al. 2020 had integrated the Q-learning with deep neural network to schedule the tasks in an effective manner. The proposed approach used to solve the problem of Directed Acyclic Graphs (DAG) in a cloud environment. The model was developed in Workflow Sim and performance was compared with the make span and load balancing in a real-life environment. The proposed solution is advantageous to provide the required scalability, containment, and learning ability [21].

Fu et al. 2021 proposed an effective scheduling technique using the PSO. In the first phase, dividing the particle swarm and then particles position was changed using the crossover mutation. The genetic algorithm further used to emphasize the search range of the space and avoids the probability to fall in the local optimal solution. Further, feedback mechanisms were used to experience the particles direction. The proposed solution is robust and outcomes show that empowering mechanisms used to avoid the space [8].

Sanaj & Prathap 2021 had been used the MAP reduce framework to schedule the tasks in an efficient manner. The authors initially extracted the task features and then reduce the features using the proposed algorithm. Further, massive tasks were separated into small tasks using the framework and scheduled using the optimization technique. The experimental results show that tasks were efficiently scheduled and proposed solution outperforms the other techniques [22].

Jalalian & Sharifi 2022 proposed a hierarchical task scheduling mechanism to enhance the efficiency of the resources. Further, K-means clustering algorithm was developed to optimize the clusters and then evolutionary algorithms were used to reduce the makes pan time. The proposed work was simulated in the Cloud Sim toolkit and experimental outcomes show that there is a reduction of makes pan time, and CPU utilization in comparison to Reinforcement Learning algorithms. The data transfer cost had been analyzed further to predict the overall cost of the distribution system. The results were prominent but limited to schedule the task in a real environment [23].

Arunadevi, & Thulasiraaman 2022 proposed an effective Cuckoo Search based task scheduling algorithm. The study includes the pre-processing of the data streams in which information was discretized and then consolidate the information using the Chi-square method. The authors also introduce the MapReduce framework utilizing the multi-objective CS to target the CPU time, memory utilization, and energy consumption. The proposed study shows robust

information but limited to analyse the operational parameters such as CPU utilization, cost and bandwidth [24].

Loheswaran et al. 2019 proposed a hybrid algorithm, called Hybrid Cuckoo Search with Iterative Local Search (HCS-ILS), for task scheduling in the cloud environment. The algorithm combines cuckoo search optimization with iterative local search to improve efficiency. Simulation results show that HCS-ILS outperforms the conventional cuckoo search algorithm and its variants in terms of minimizing makespan and increasing resource utilization [25].

García et al. 2018 utilized Big Data techniques to implement metaheuristic algorithms for industrial decision-making. They evaluated the algorithm's performance in terms of result quality and convergence times under various conditions. The proposed approach was a cuckoo search binary algorithm implemented in Apache Spark, tested on multiple crew scheduling instances. However, the experiments showed that achieving satisfactory results and iterations depended on specific problem characteristics and was not consistently guaranteed [26].

Agarwal et al. 2016 proposed a cuckoo search-based approach for task scheduling, aiming to efficiently distribute tasks among available virtual machines (VMs) while minimizing overall response time (QoS). This algorithm allocated tasks to VMs based on their processing power (measured in million instructions per second, MIPS) and task length. They compared the cuckoo search algorithm with first-in-first-out (FIFO) and greedy-based scheduling algorithms using the CloudSim simulator. The results demonstrated that cuckoo search outperformed the other algorithms [27].

3. The Proposed Solution

The proposed solution aims to reduce the computation complexity of the simulated big data. The proposed solution is divided in two segments namely the scheduling architecture which uses Q-learning algorithm for the selection of appropriate data node based on the buffer delay. The second section utilizes the reward mechanism and trains the system using Feed Forward Neural Networks for the training of the system against two ground truth labels namely "Acceptable"," Moderate- Acceptable" in order to further reduce the computation complexity of the exploration part of the Q-learning algorithm.

3.1. Section 1: The scheduling architecture using the Q-Learning

The scheduling architecture is now considered as the defined problem of the Q-learning algorithm. The learning algorithm is learning from its taken actions in the systems incorporating the state variables undertaken during the simulation and has been referred as Agd in earlier segments.

The State and the Variables(S_v) : The state variable is a set of finite attributes described S_v at a state time 't' can be defined as

$S_v = \{l1, l2, \ldots . ln\}$ where 'l' ={IIR, Buffer_Delay(B_D)}$\in$ Agd and B_D is the buffer delay of the data nodes under one data center. Each data node has multiple buffers based on the priority of the task which ranges from 1-8 viz each data node will have 8 buffers to handle. Each data center is found to be busy executing the buffer tasks and as per assumption, no buffer is empty.

The action set (As=$\{As_1, As_2 .. As_n\}$) where n is total number of users in Agd, is the set of possible actions that a learning agent would select in order to generate maximum reward. Any action will represent the allocation of the user job to data center.

The environment and the observations from environment

Consider the assumption that, user U_1, at the current state S_{v_1} is to be assigned to data node d_{n_1} and task has a set of attributes A_1. By taking the action A_1, U_1 will be allocated to d_{n_1} at the end of respective buffer queue ξ. As per the policies of Q-learning, the learning agent will come to know the situation of the new state as soon as A_1 has been performed. The next state will provide either a reward generated against the action A_1 or the agent will be penalized as the result of the action. In both the cases, the learner will update its information regarding its surrounding objects. The average B_D can be calculated as follows.

$$B_D = T_{ct} + \sum_{i=1}^{p} \frac{U_p}{Er} \quad (1)$$

Where T_{ct} is the task completion time, p is total number of users in the buffer, U_P is the p^{th} user, Er is the execution rate under 1 specific buffer. All the tasks are associated with some completion time and that is pre-defined. The reward for such a situation is defined as follows.

$$R = \begin{cases} 1 & \{\min\{U.B_{D_t}\} < \aleph + \varphi_1 \leq d_n.cpu_{util} \leq \varphi_2\} \\ 0 & otherwise \end{cases} \quad (2)$$

where φ_1 and φ_2 are upper and lower threshold of CPU utilization of the data node d_n for the User U with Buffer delay for job type t and $\aleph$ is the task deadline. There are total number of B buffers in the list. The reward uses the buffer delay time and the upper and lower threshold of CPU utilization against the given priority of the task. The data node is over-utilized if the CPU utilization of the current data node is higher than the upper threshold φ_2. The data node is under-utilized if the CPU utilization of

the data node is below lower threshold φ_1. If the CPU Utilization of the data node remains under the boundary of $\varphi_1 and \varphi_2$, the data node is considered as a suitable candidate for the allocation of the user. Though, it is not necessary that if the data node is one of the suitable candidates for allocation, it must be the best candidate for allocation and hence a Q-table is created for all the possible actions as allocations has been considered. It is to be noted that, the overall deadline of the task should not be exceeded by the Buffer Delay. The algorithmic architecture for the Q-learning is illustrated in Algorithm 1. For one Buffer, the total number of waiting tasks in the buffer is equal to q, and hence the Q-table will be of size q $\times \varpi$ where ϖ is the total number of waiting task in the buffer.

Algorithm 1: The Q-learning Algorithm

Input: Q ,S_v , B_q , t-list // where t-list is the total task list , B_q is the buffer queue

Q-L[1] **while** $t - list. \varpi > 0$

Q-L[2] **Wait** if $B_q = \varpi$ for all $1 \leq B$

Q-L[3] ϖ_l = Extract ϖ from p_r from $t - list$ // where p_r is the priority list

Q-L[4] $A_l = Select_{Action(Q,S_t)}$ // Use greedy method to select an action

Q-L[5] Perform A_t // Merge ϖ_l to B_q ,

Q-L[6] Evaluate R_t and S_{t+1}

Q-L[7] $Q(S_{v_t}, A_t) = (1-\propto)Q(S_{v_t}, A_t) + \propto [R_t + \gamma. maximum_{A_t} Q(S_{v_{t+1}}, A_t)]$ // where $\propto$ is the learning rate and γ is the discount.

Q-L[8]$S_{v_t} = S_{v_{t+1}}$

Q-L[9]**End**$_{\textbf{while}}$

The algorithm 1, takes the Q, S_t and B_q as buffer queue along with task list as t-list. With every simulation iteration, the Q-table which is hold by Q, is updated utilizing the Bellman's equation defined in Q-L[7] of algorithm 1. The process repeats itself until all the tasks in the buffer are not complete. At a given state value S_t, the scheduling architecture selects an action A_t by applying Greedy Search policy (ε -greedy-policy) in which the exploration part dominates the exploitation part for a while in order to understand the reward of selection.

$\overline{S_{V1}} = \langle l_1 + 1, l_2, \ldots\ldots\ldots, l_k, \ldots\ldots\ldots\ldots l_n \rangle,$

$\overline{S_{V2}} = \langle l_1, \quad l_2 + 1, \ldots\ldots\ldots, l_k, \ldots\ldots\ldots\ldots l_n \rangle$

…… … …… ….

…… … …… ….

$\overline{S_{Vk}} = \langle l_1, \ l_2, \ldots\ldots\ldots, l_k + 1, \ldots\ldots\ldots\ldots l_n \rangle,$

……. … … …

$\overline{S_n} = \langle l_1, \ l_2, \ldots\ldots\ldots, l_k, \ldots\ldots\ldots\ldots l_n + 1 \rangle$ (3)

When a system is forced to explore, viz. in case of ε -greedy-policy, the system picks random state and performs the action and a reward is generated. The process continues, till all the possible actions are not suppressed into actual implementation until the discount becomes too big to be adjusted. The Q-value is updated using the Bellman's equation as given by eqn 4.

$$Q(S_{v_t}, A_t) = (1-\propto)Q(S_{v_t}, A_t) + \propto [R_t + \gamma. maximum_{A_t} Q(S_{v_{t+1}}, A_t)] \quad (4)$$

Where $\propto$ is the learning rate and γ is the discount. The selected data node is supplied with the job and the data node performs the execution. It is to be noted that, Q-learning gets the input from mini-max architecture, which uses map-reduce separation architecture. Post execution of the job, the data is aggregated as Agd$\in$ $\{U, Dn, IIR, Th, Sd, Ext, CO_{2_e}, O_{ur}, U_{ur}, C_p\}$ attributes where U is user, Dn is the allocated data node, IIR is the instruction injection rate, This throughput, S_d is the storage demand, $CO_{2_e}$is the carbon emission, Ext is the execution time, O_{ur} is the overutilization ratio, U_{ur} is the underutilization, and C_p is the consumed power. The Agd is divided into three groups that are further labeled using statistical machine learning.

3.2. Section 2: The data-selection, training, and classification model

The data selection process incorporates a new search behavior based on the Levy flights generated to place the record to appropriate group. The group generation is performed utilizing centralized k-means algorithm. K-means algorithm faces convergence issue due to its random centroid selection process for the first centroid [28].

The updated k-means algorithm can be illustrated using Pseudo code centralized k-means

The segmented class label entries are passed for attribute set selection utilizing Advanced Cuckoo Search(A-CS) algorithm.

Pseudo Code 1 for Centralized k-means

Input: Count of Centroids (C); Set of task (T); centroids list assigned randomly (C_C).

Output: Task scheduled with their centroid

Begin

1. Repeat
2. For each task in T do
3. Calculate the distance between the tasks assigned to each node and the quality value using the equation 3.
4. Assign the tasks to the nearest centroid

5. End for
6. For each task in C_C do
7. Compute the new centroid task allocation based on quality using the equation 4.
8. End for
9. Until all the tasks assigned to the VMs or the iterations reached to maximum.

End

The K-means clustering algorithm has been used to determine the centroid of each task allocated to the VMs. K-Means has resolved different problems encountered during the task scheduling in the cloud environment to manage the Big Data. Initially, different tasks have been allocated by computing the centroid of each node. The node which is nearest to the centroid has been allocated and that are farther away from the centroid schedule the task accordingly.

Table 1: Notations and Abbreviations

Agd	Aggregated Data
A-CS	Advanced Cuckoo Search
Sd	Selected Data
$\mathbf{Total}_{eggs}$	Total aggregated data
Π	Policies
Q-L	Q-Learning
σ	Levy Flights
S_t	State Value
l	IIR Buffer Delay
IIR	Instruction Injection Rate
τ	Random Index of Feature
∂	Distribution Ratio of Egg
B_D	Buffer Delay
ξ	Buffer Queue
Sv	The State and the Variables
As	Action Set
n	Total Number of users in Agd
T_{ct}	Task Completion Time
p	Total Number of Users in the Buffer
Er	Execution Rate under 1 Specific Buffer
φ_1 and φ_2	Upper and Lower Threshold of CPU Utilization
J	Job Type
$\aleph$	Task Deadline
d_n	Data Node
t-list	Total Task List
B_q	Buffer Queue
p_r	Priority List
$\propto$	Learning Rate
γ	Discount
Th	Throughput
Sd	Storage Demand
CO_{2_e}	Carbon Emission
Ext	Execution Time
Our	Overutilization Ratio
Uur	Underutilization
Cp	Consumed Power
Cc	Centroid List

The aim of the A-CS algorithm is to keep most relevant data records belonging to one class in order to train the system at its most precise manner. Hence here in the proposed solution, one class elements are considered as eggs of one host. The Cuckoo bird in the proposed solution does not put its egg into other host's nest but, checks its own eggs in order to find any intruding egg from any other host's nest. To do so, the cuckoo bird performs a rigorous analysis by dividing its own eggs into multiple segments and performs a Levy-flight evaluation method in order to check the sustainability of the examined egg with other eggs in the nest. The CS algorithm takes the aggregated data along with the clusters that are identified by the proposed k-means algorithm. The cuckoo bird keeps a Levy-flight and shuffles the egg's attributes that are the simulation parameters evaluated under the policy.The policy contains three distribution ratio of {d1,d2,d3} as a set of two values. The first value represents the percentage of total number of eggs that would be taken from egg's inner native host and the second value represent the percentage of total eggs that would be taken from egg's inner non-native host. As for example, in the distribution policy $\sqcap_{d1}$, 70 % eggs will be considered to be paired from egg's inner native host and 30% eggs will be considered from egg's non-native node.

The proposed A-CS algorithm creates a hypothetical two nest system within a single host cuckoo. The aim of the algorithm is to keep the best records belonging to one single class. In order to do so, each record is referred as one cuckoo egg. Each egg must pass a 10 Levy flight organization in which each egg will be placed with random number of other eggs with each Levy flight. Levy flight is the duration of the time a cuckoo bird goes out of its nest. When the cuckoo comes back, it hypothetically shuffles the eggs one by one until the last egg is not considered. The proposed A-CS algorithm introduces an angularity-oriented fitness function which utilizes cosine similarity for the calculation of the fitness value. Each Levy flight contains a reward of 1 unit if the fitness of cuckoo egg to the satisfaction of cuckoo bird matches. To do so, the cuckoo bird divides the entire eggs in two separate inner nest and it calls them native nest/hive and non-native nest/hive. The cuckoo bird applies the policy of distribution $\sqcap_{distribution}$. The fitness function calculates the cosine similarity between the selected random population along with their hive and the cosine similarity between the selected random population containing the current egg as well.

Pseudo Code 2 for A-CS algorithm

A-CS1. Pseudo Code A − CS
A-CS2. Inputs: Agd,Output： Selected Data (Sd) // where Agd is the aggregated data set//

A-CS3. $Sd = [\,]$ // Initialize an empty index for selected data //

A-CS4. $Total_{classes} = 3$ // It is the total number of classes which is separated using centralized $k-means$ //

A-CS5. $For_{main}\ class = 1: Total_{classes}$

A-CS6. $dtp = \oint_{i=1}^{t} Agd.data_{values}$

A-CS7. // Extract the data values for the current class getting processed and put it to data to process (dtp)//

A-CS8. $Total_{eggs} = dtp$

//The total aggregated data against the supplied class will act as eggs of the cuckoo that the bird will aim to place in other hives //

A-CS9. $inner_{hive} = 2$ // As the cuckoo bird will have to perform analysis over its own egg,

A-CS10. the cuckoo bird divides the eggs in two different segments and validates the under − examination egg against both the classes //

A-CS11. $[Index_{inner}, Centroid_{innner}] = k-means(Total_{eggs}, inner_{hive})$//

Divide the data into two segments //

A-CS12. $\sigma = 10$ // Total Levy flights $= 10$ //

A-CS13. $Fitness_{reward} = 0^{s}\{1:\sigma\}$

Initialize a reward array for the Levy flights each containing a 0 value for each egg and will be updated as per the flight reward //

A-CS14. $For_1\ j = 1: Total_{eggs}.count$ // consider each egg

A-CS15. $C_g = Total_{egg}[j]$ // Take out the first egg //

A-CS16. $\emptyset = Total_{eggs}(\tau) \qquad \tau \epsilon\ At_{set}$

A-CS17. select random feature set, where τ is the random index of feature, At − set is the attribute set as $At_{set} \rightarrow$ $\int_{t=1}^{simitr}\{U, Dn, IIR, Th, Sd, Ext, CO_{2_e}, O_{ur}, U_{ur}, C_p\}\,dt$ where simitr is the total number of simulation records aggregated. In case of proposed work model, simitr $\epsilon\{i_1, i_2 \ldots . . i_l\}$ where 'i' is 100000//

A-CS18. $For_{1_1}\ jj = 1:\sigma$ // Loop for every Levy flight

A-CS19. define $\partial \epsilon \prod$ distribution $_{\{70-30\ ,60-40\ ,50-50\}}$

A-CS20. *where ∂ is the distribution ratio of egg made under the* $\prod$ *to accommodate eggs in the host nest* //

a. $total_{population} = Total_{egg}.population()$ // Total cuckoo eggs in the nest//

b. $deployed_{population} = total_{population}.random()$//Generate a random population //

c. $discounted_{population} = \{\prod \frac{\prod distribution\ .e1}{100}\ ,\ \frac{\prod distribution\ .e2}{100}\}$

d. //Create the discounted population following $\prod$ distribution law, as for example if $\partial == 1, e1 = 70\ , e2 = 30$ //

e. $pairing_{index} = discounted_{population}.index\ ()$//

Extract the indexes of the discounted population //

A-CS21. $paired_{index} = (pairing_{index} \cup Cg_j)$

Merge the current cuckoo egg index to all the indexes gathered in pairing index //

$CS1 = Avg - Cosine_{similarity}(pairing_{index}, pairing_{index}.Centroid)$//

The cuckoo bird calculates similarity of the pairing index to its relative centroid //

$CS2 = Avg - Jaccard_{similarity}(paired_{index}, pairing_{index}.Centroid)$

// The centroid for both the indexes will be same //

A-CS22. If $CS2 > CS1$

A-CS23. $Fitness_{reward}[j] = 1$

A-CS24. End If

A-CS25. End For_{1_1}

A-CS26. $Successful_{levys} = \sum find(Fitness_{reward} > 0)$

A-CS27. If $\text{Successful}_{\text{levys}} > \sigma \times \frac{50}{100}$
A-CS28. Keep Egg
A-CS29. Sd.append(j)
A-CS30. Else
A-CS31. Reject Egg
A-CS32. End
A-CS33. End For_1
A-CS34. $\text{End For}_{\text{main}}$
A-CS35. Return Sd

Cosine Similarity: The cosine similarity is a distance measure which uses eq (5) which is further simplified to eq (6).

$$Cosine_{similarity} = \frac{C1.C2}{||C1||\times||C2||} \quad (5)$$

$$Cosine_{similarity} = \frac{\sum_{i=1}^{n} C1_i.C2_i}{\sqrt{\sum_{i=1}^{n} C1_i^2}.\sqrt{\sum_{i=1}^{n} C2_i^2}} \quad (6)$$

where n is total number of attributes in the cluster

Being an angular similarity measure, the $90^0\ or\ 270^0$ is the maximum value when it comes to least similarity and arithmetically it is $Abs(Cos(\{90^0, 270^0\})) = 0$ and hence the aim is to maximize the co-relation value which can be flt on the current object i.e $0^0 or 180^0$.

Arithmetically, $Abs(Cos\{0^0, 180^0\}) = 1$. The fitness function of the A-CS algorithm evaluates CS1 and CS2 for the illustrated scenario by using eq (6).

$$f_{cuckoo} = \begin{matrix} 1 & if CS1 \leq CS2 \\ 0 & otherwise \end{matrix} \quad (7)$$

The selected data against each ground truth value is supplied to Feed Forward Neural Networks for the training.

3.3. Feed Forward Neural Networks (FFNN)

In this section, tasks are distributed considering the different ratios such as 70:30, 80:20, and 90:10. The propagation model namely Levenberg has been applied accordingly and then stopping criteria namely Gradient has been used to stop the scheduling process. The regression model has been used to select the different tasks and the developed FFNN model has been validated using the Mean Square Error. The ordinal measures are as follows.

Table 2: The Ordinal Measures of Used Feed Forward Neural Networks

Propagation Type	Feed Forward
Distribution	a)70-30 b)80-20 c)90-10
Model of propagation	Levenberg
Stopping Criteria	Gradient
Layers of Propagation	5-20
Selection type	Regression
Validation type	Mean Squared Error

4. Result and Elaborations

The learning rate of the proposed work model is illustrated in contrast to the number of steps considered in the architecture. A tabular representation is given as follows. The table demonstrates the actions taken by the agents in the proposed contrast. For the current instance, there are 3 users to be migrated from a list of 500 users. There are 30 users that are running on an overloaded data node. For the migration of 2 users, 6 users have been selected based on the lower utilization of the available resources. These users were propagated through learning steps of greedy search algorithm to form Q-table. The table can be illustrated as follows. The Q-value for 8th step for three of the users is same viz. 0.134752. It indicates that all the users were falling into similar structure of jitter at the data node and hence these two users are to be migrated from the current data node to another data node based on step-8.

Table 3: Q-value

Users	1	2	3	4	5	6
Q-value Step 1	0.13703	0.13475	0.13564	0.13475	0.13475	0.07521
Q-value Step 2	0.13564	0.13656	0.13475	0.13564	0.13564	0.13475
Q-value Step 3	0.13703	0.13475	0.13564	0.13475	0.13475	0.13703
Q-value Step 4	0.13475	0.13564	0.13656	0.13564	0.13656	0.13564
Q-value Step 5	0.13475	0.13656	0.07512	0.13475	0.13475	0.13475
Q-value Step 6	0.13475	0.13475	0.07567	0.07521	0.13656	0.137(
Q-value Step 7	0.07423	0.07512	0.13564	0.13475	0.13656	0.1347
Q-value Step 8	0.13656	0.13475	0.13475	0.13475	0.07549	0.1365
Q-value Step 9	0.13475	0.13564	0.13475	0.13656	0.13656	0.1347
Q-value Step 10	0.13475	0.13475	0.13475	0.13475	0.13564	0.1347
Q-value Step 11	0.07521	0.13475	0.13475	0.13475	0.13475	0.13475
Q-value Step 12	0.13475	0.13564	0.13475	0.13475	0.13564	0.13475
Q-value Step 13	0.13475	0.13475	0.13564	0.13475	0.13475	0.13475

The results have been evaluated on the base of two aspects. The first aspect relates to the training and classification architecture which involves the usage of SI for the selection of the separated data. In order to evaluate and validate the performance, following attributes have been evaluated.

- True Positive Rate (TPR): It is the true detected samples from each class to the total number of samples of the respective class.
- False Positive Rate (FPR): It is the false detected samples from each class to the total number of samples in the respective class.
- Accuracy: It is the arithmetic mean of TPR.

To evaluate the performance of the proposed algorithm A-CS, a set of users have been deployed over a set of data nodes placed under the name node. The name node maps the request from the user and finds the possible solution to the requests of the users. Due to the application of the A-CS that is integrated with k-means algorithm as shown in Fig. 2, is supplied to feed-forward neural networks. The neural network runs regressive engine that follows the Levenberg principle of propagations in order to train the system.

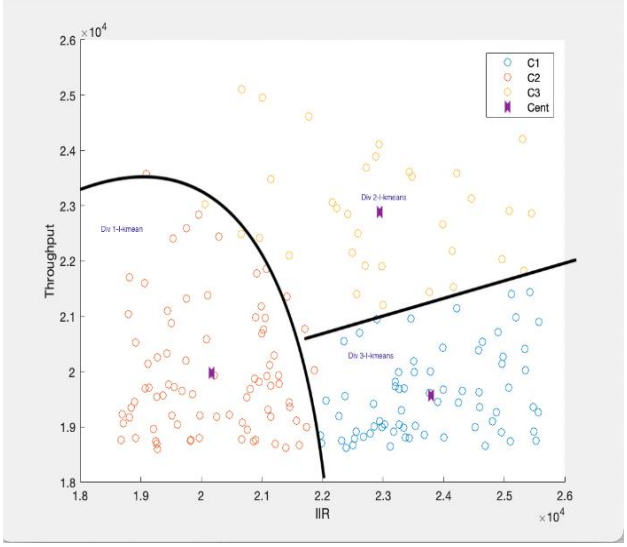

Fig. 2: Data Segmented by I-k-Means

A-CS selects the most relevant data from the list that are concise and more applicable to the list. To do so, a novel fitness function and a flight behavior is introduced in the proposed work section. A-CS reduces the population size as shown in Fig. 3.

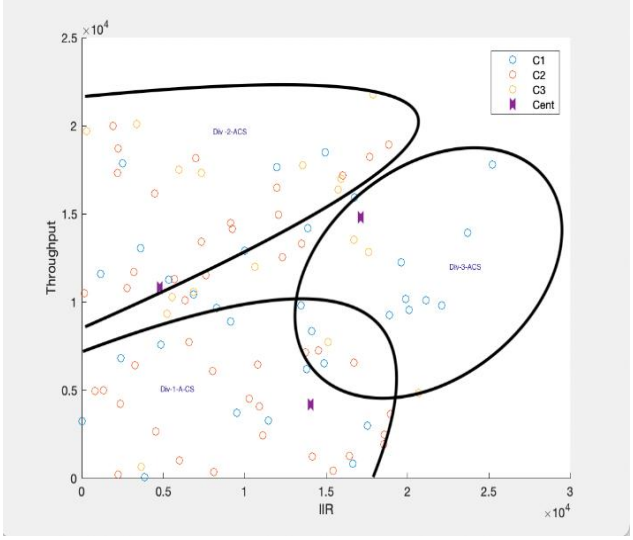

Fig. 3: The Selected Identities After the Application of A-CS

It is evident from Fig. 3 that, A-CS reduces the unused sample space or the sample space that is quite far from the original centroid of the cluster. Due to the application of A-CS algorithm, the centroids of the clusters also shift more towards the data rather than being towards the free space. To illustrate the working of A-CS, a small set of examples is provided with statistical values as follows. This provides conciseness to the training algorithm. To train the system, a total Starting from the first centroid, the following indexes have been identified to fall in the first centroid as shown in Table 4.

Table 4: Elements of First Centroid

2	**3**	**4**	**5**	**8**	**9**	**11**	**13**	
14	15	17	22	31	37	40	42	47

Each record value of this centroid set, is now being considered as one egg from Cuckoo bird.

10 random indexes as per policy 1 have been generated as follows. The first distribution states that, 70% of the population for the comparison should be generated from the existing set of data and rest 30% competitors will be selected from other two hives.

First of all, a random sample of feature set will be selected to evaluate each cuckoo egg. The selection of the features will be done on the base of these features for the first levy flight and for the first cuckoo egg. There is total 11 features that are supplied as input and hence the generated index would be 1-11.

Table 5: Egg Features for the Evaluation of First Cuckoo Levy Flight

5	**3**	**2**	**3**	**5**	**4**	**8**	**10**	**1**	**1**

The elements of Cluster 1 are going to be divided into two separate inner hives and based on the properties, two further distribution is listed in table 5.

Table 6: Egg Features for the Evaluation of First Cuckoo Levy Flight

1	**1**	**1**	**1**	**1**	**1**	**1**	**2**	**1**
1	1	1	1	1	1	1	1	

Based on the selected sample, five indexes have been selected for the evaluation of the current cuckoo bird. The indexes are {3 3 6 2 5 }. The current cuckoo bird will be collaborated with these indexes and the overall index will become {3 3 6 2 5 1} as the scheduler is on first bird right now.

As shown in Table 7, first 7 identities have been selected from the same hive whereas 3 identities viz. 10,1 and 1 belongs to other hive. The proposed work algorithm has not put any limit on duplicate identities and hence repetitive identities have also been observed. As for instance 3 and 5 repeats itself in 70% distribution and 1 repeats itself in 30% distribution. Now the cuckoo set combined value with the other eggs is passed to cuckoo fitness function which uses two similarity measures namely Cosine similarity and Jaccard similarity and is illustrated in the pseudo code as well.

The collaborative features will be indexes as follows.

Table 7: Indexes Evaluation for Current Cuckoo Bird using Selected Samples

23900	**19459.329**	**6.196**	**0.00000439**	**14.684**	**0.349**	**0.637**	**150.975**
23900	19459.329	6.196	0.00000439	14.684	0.349	0.637	150.975
24961	20146.855	9.898	0.00000614	16.950	0.565	0.410	139.129
19834	18765.123	9.703	0.00001123	18.495	0.352	0.589	131.174
21982	18706.155	3.085	0.00000663	14.374	0.552	0.415	133.429
22612	20700.321	1.455	0.00000142	15.005	0.341	0.622	138.431

The cuckoo search algorithm accepts two parameters as input, the first input is the collected egg values from the hives and the second value is the value of the centroids of the hive that contains more number of eggs in the list. An average of both the similarities have been evaluated and if the average similarity of the considered group is more than that of the similarity of the hive that contains a greater number of candidates, the cuckoo fitness return 1 in favor, else it return 0. For each flight, the proposed algorithm gets

0s and 1s in return and by the end of all flights, if number of 1s are more than that of number of 0s in the list, the egg is accepted and will remain in the save host value else it would be discarded from the existing group.

The selected dataset from each category will be passed to multi-layer propagation model which is supported by Levenberg's propagation architecture. The ordinal measures of the propagation network is provided in table 7.

In order to select the total number of layers, a regression analysis has been formed utilizing the Neural Network toolbox of MATLAB.

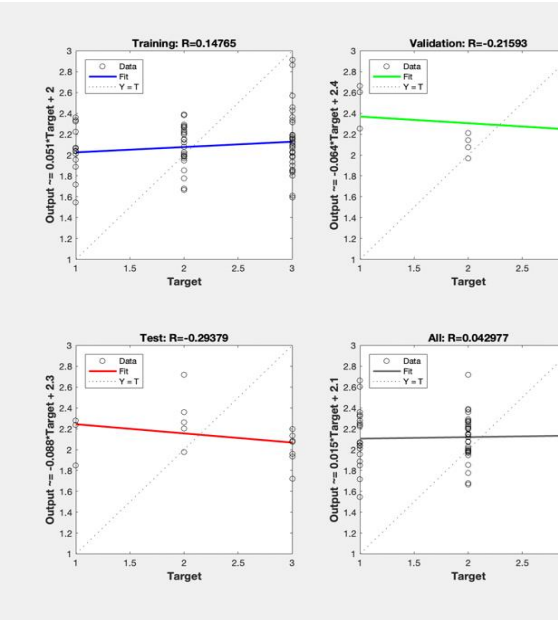

Fig. 4(a): Regression Analysis φ = 5 Rmax = .04

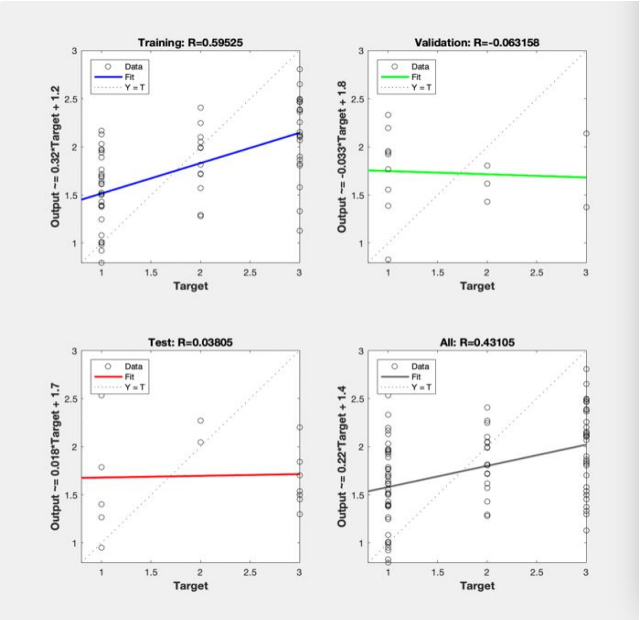

Fig. 4(b): Regression Analysisφ = 10 Rmax = .43105

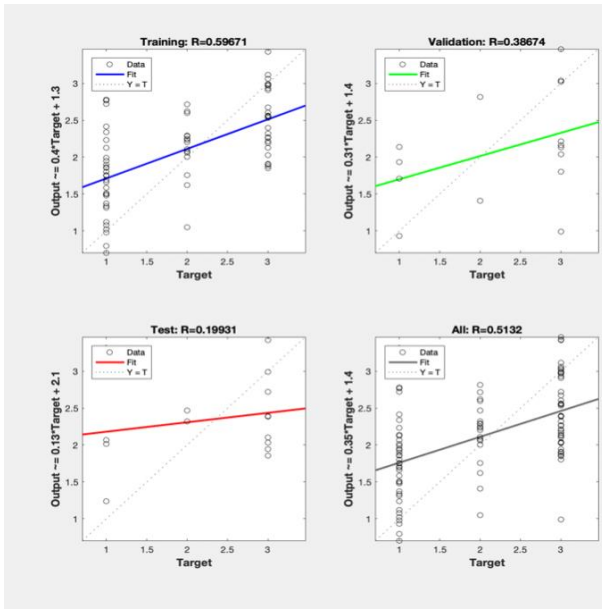

Fig. 4(c): Regression Analysisφ = 15 Rmax = .5132

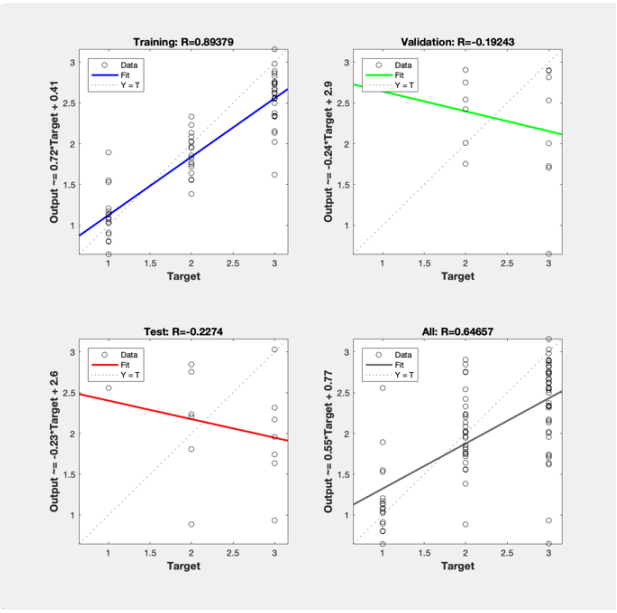

Fig. 4(d): Regression Analysis φ = 20 Rmax = .64575

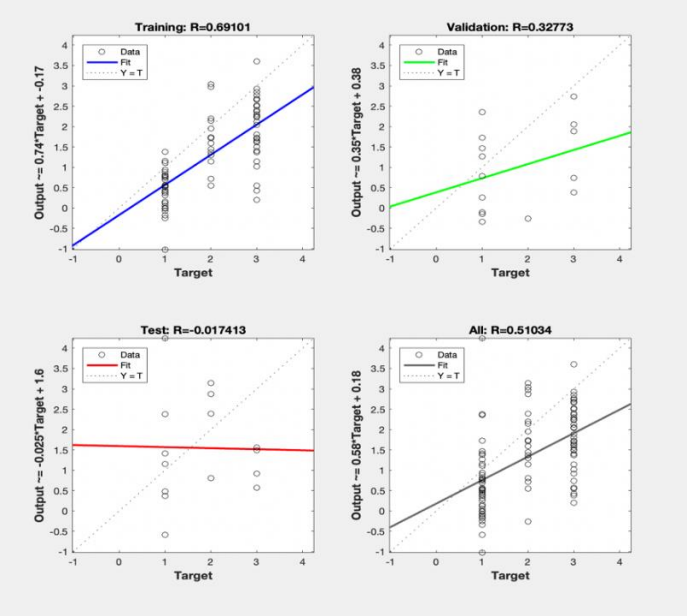

Fig.4(e): Regression Analysis $\varphi = 25$ Rmax $= .51034$

Table 8: Classification Results

Number of Test Samples	Tpr Proposed	Tpr Naïve Bayes	Tpr Random Forest	Tpr Multiclass-SVM	Fpr Proposed	Fpr Naïve Bayes	Fpr Random Forest	Fpr Multiclass-SVM
1000	0.8962	0.8259	0.8374	0.8011	0.1038	0.17415	0.1626	0.1989
2000	0.9004	0.8323	0.8473	0.8295	0.0996	0.16768	0.1527	0.1705
3000	0.9176	0.8489	0.8511	0.8243	0.0824	0.15107	0.1489	0.1757
4000	0.9279	0.8520	0.8607	0.8370	0.0721	0.14800	0.1393	0.1630
5000	0.9399	0.8602	0.8742	0.8309	0.0601	0.13980	0.1258	0.1691
10000	0.9413	0.9182	0.9212	0.8350	0.0587	0.08177	0.0788	0.1650
15000	0.9479	0.9150	0.9265	0.8486	0.0521	0.08496	0.0735	0.1514
20000	0.9545	0.9292	0.9300	0.8499	0.0455	0.07082	0.0700	0.1501
30000	0.9675	0.9372	0.9344	0.8684	0.0325	0.06282	0.0656	0.1316

The regression value at $\varphi = 25$ Rmax $= .51034$ is less as compared to the $\varphi = 20$ Rmax $= .64575$ which indicates that the most suitable propagation value is $\varphi = 20$ .The trained value is classified against the 30% separated data as classification test data. The classified results are also compared with other state of art techniques namely Naïve Bayes, Random Forest and multiclass SVM and the classification results are illustrated in the table 8.

Out of 100000 simulation records, 30000 simulation records were kept separate in order to perform the classification. The ground truth of the test samples was already available for the processing and hence three individual parameters have been evaluated. The samples were tested against their ground truth value and True Positive Rate(tpr) represents the total true identified samples to the total number of samples. False Positive Rate(fpr) is evaluated using 1-tpr values. As evident from the results, the tpr of proposed algorithm architecture has highest values in all records comparisons. A certain raise in the classification percentage is also observed as the number of records increases from 1000 to 30,000. This is due to the correct sampling of the data in the proposed algorithm A-CS. A-CS helped the neural engine to establish the co-relation between the ground truth values and hence as the record set increases, the co-related samples are categorized to their original ground truth value. As tpr $1/\propto$ to fpr, higher tpr will result into lower fpr values. The overall accuracy of proposed algorithm is also

high as compared to other state of art classification algorithms as shown in Fig. 5.

The overall algorithm architecture, results into improved overall QoS improvement and the statistical values are listed in table 9 to 11. The QoS evaluation is centric towards two parameters under three supplied load conditions. The load varies in a scenario {25000,30000,35000} packets/second. As the supplied load increases, the overall execution rate i.e the throughput also raises in each state of art technique along with the proposed algorithm. This is due to the fact that all the compared state of art techniques has worked on balancing the load and improving the overall execution rate of the system and hence it is not surprising to see that none of the algorithm whether it is the proposed algorithm or any other state of art technique fails when the load increases. Despite of the fact that none of the algorithm failed on increasing the load, the proposed algorithm outcasts the other techniques and are illustrated in table 8, 9 and 10.

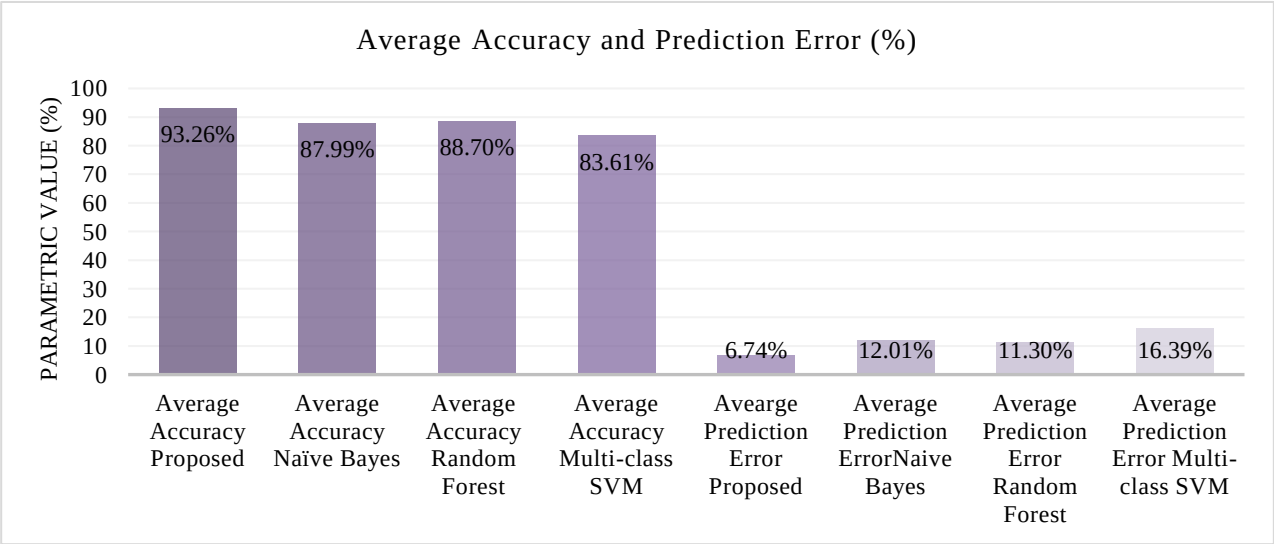

Fig. 5: Classification Accuracy and Prediction Error

Table 9: QoS Analysis, load =25000 MIPS

Total Number of Simulations	Average Throughput Proposed	Average Throughput Loheswaran et al. [25]	Average Throughput Garcia et al.[26]	Average Throughput Agarwal et al. [27]	CO2 Emission Proposed	CO2 Emission Loheshwaran et al. [25]	CO2 Emission Garcia et al. [26]	CO2 Emission Aggarwal et al. [27]
1000	22902.625	21091.243	18832.788	18406.704	13.348	18.264	19.397	19.891
2000	22727.966	20165.477	18521.188	18718.734	17.732	19.811	17.599	20.824
5000	22912.37	21277.621	18340.445	17804.048	12.89	16.798	14.967	19.764
10000	21996.265	21116.359	19387.12	19280.359	15.031	19.569	18.193	20.963
15000	22910.127	20642.1	19338.176	18902.131	13.518	16.55	16.039	18.041
20000	22311.409	21584.767	17866.551	19030.207	17.105	17.337	19.783	20.055
25000	23196.854	20243.367	18957.814	18869.331	15.931	21.277	16.159	17.393
30000	23256.834	21529.986	19294.957	18926.465	13.701	21.521	15.734	20.266
50000	22848.297	22213.864	18127.326	19242.971	12.695	21.083	18.399	19.016
60000	22855.363	22262.933	18065.593	19529.834	14.917	21.053	18.326	16.185
80000	24263.146	22669.707	19952.424	18620.571	14.646	16.979	14.426	19.583
100000	22957.014	21368.663	20129.65	20063.093	13.115	19.165	16.394	20.585

Table 10: Analysis, load=35,000 MIPS

Total Number of Simulations	Average Throughput Proposed	Average Throughput Loheswaran et al. [25]	Average Throughput Garcia et al.[26]	Average Throughput Agarwal et al. [27]	CO2 Emission Proposed	CO2 Emission Loheshwaran et al. [25]	CO2 Emission Garcia et al. [26]	CO2 Emission Aggarwal et al. [27]
1000	26637.293	23122.870	20741.833	19636.846	19.276	22.547	22.463	19.991
2000	26522.862	20983.829	19481.223	19391.589	15.058	19.612	20.261	18.280
5000	26059.158	22571.834	21207.023	19795.774	14.264	22.318	22.718	22.145
10000	26779.509	21702.818	19716.660	21170.620	18.180	21.494	18.055	19.238
15000	26723.542	21590.502	20460.468	19746.355	15.350	22.809	18.583	22.192
20000	26865.428	22342.254	20030.799	19823.284	18.219	24.521	21.810	20.135
25000	26797.583	22013.887	21232.056	21329.115	18.181	23.360	20.065	21.975
30000	26388.171	21585.021	20928.066	20255.314	15.357	24.346	21.584	18.465
50000	27248.131	22983.666	20313.241	21703.867	18.278	24.400	18.946	18.253
60000	27224.545	22056.537	21212.118	20783.430	15.562	20.738	22.024	20.494
80000	27100.835	22490.209	21196.748	21526.341	17.529	23.156	19.183	18.239
10000	27513.816	23161.359	21002.051	20712.199	17.633	23.739	20.851	21.419

5. Conclusion

In the cloud computing environment, it is vital to implement an effective task scheduling technique that affects the system performance in terms of carbon dioxide emission, energy consumption, and system load. The Q-learning technique guarantees the task scheduling of the Sensor node in an optimal manner. Therefore, this paper elucidates the Machine Learning based Q-learning technique to schedule the task by validating the Quality value considering the different loads. Further, the Q-value for the different number of nodes has been computed and it is clear that the Q-value for 8^{th} step for three of the users is same as 0.134752 which means that all the users were falling into similar structure and hence these users are to be migrated from the current data node to another data node based on step-8. Consequently, the average throughput value for the different number of nodes has been computed as 2.06E followed by illustrating the over utilization and underutilization of resources. Further, the average value for the overutilization and underutilization of resources is 0.448 and 0.51 for the different number of nodes. The performance metrics in terms of TPR, FPR, and Accuracy has been measured to validate the results. The average TPR obtained using the proposed technique is 0.93 and that of FPR is 0.07. The proposed technique further compared with the Naïve Bayes Multi-class SVM, and Random Forest technique to determine the superiority. The proposed technique has been improved by 5.6% and 4.5% in comparison to Naïve Bayes and Random Forest technique.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] F. Marozzo and D. Talia, "Perspectives on Big Data, Cloud-Based Data Analysis and Machine Learning Systems," *Big Data and Cognitive Computing 2023, Vol. 7, Page 104*, vol. 7, no. 2, p. 104, May 2023, doi: 10.3390/BDCC7020104.

[2] D. Soni, D. Srivastava, A. Bhatt, A. Aggarwal, S. Kumar, et.al., An Empirical Client Cloud Environment to Secure Data Communication with Alert Protocol. Mathematical Problems in Engineering, 2022.

[3] I. M. Ibrahim, et.al., "Task scheduling algorithms in cloud computing: A review," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 4, pp. 1041–1053, 2021.

[4] D. Paulraj, T. Sethukarasi, S. Neelakandan, M. Prakash Id, and E. Baburaj Id, "An Efficient Hybrid Job Scheduling Optimization (EHJSO) approach to enhance resource search using Cuckoo and Grey Wolf Job Optimization for cloud environment," *PLOS ONE*, vol. 18, no. 3, p. e0282600, Mar. 2023, doi: 10.1371/JOURNAL.PONE.0282600.

[5] A. Agarwal, N. Bora, and N. Arora. "Goodput enhanced digital image watermarking scheme based on DWT and SVD." International Journal of Application or Innovation in Engineering & Management vol. 2, no. 9, pp. 36-41, 2013.

[6] H. A. Alsayadi, N. Khodadadi, and S. Kumar. "Improving the Regression of Communities and Crime Using Ensemble of Machine Learning Models." Journal of Artificial Intelligence and Metaheuristics vol. 1, no. 1, pp. 27-34, 2022.

[7] V. K. Chandarapu and M. Kasa, "Balanced Prediction Based Dynamic Resource Allocation Model for Online Big Data Streams using Historical Data," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 10, no. 2s, pp. 81–87, Dec. 2022, Accessed: Nov. 17, 2023. [Online]. Available: https://www.ijisae.org/index.php/IJISAE/article/view/2366

[8] 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*, pp. 1–10, 2021.

[9] K. Kaur, S. Garg, G. Kaddoum, and N. Kumar, "Energy and SLA-driven MapReduce job scheduling framework for cloud-based cyber-physical systems," *ACM Transactions on Internet Technology (TOIT)*, vol. 21, no. 2, pp. 1–24, 2021.

[10] A. Aggarwal, S. Kumar, A. Bhatt, and M. A. Shah. "Solving user priority in cloud computing using enhanced optimization algorithm in workflow scheduling." Computational Intelligence and Neuroscience 2022.

[11] V. Vashishth, A. Chhabra, and A. Sood, "A predictive approach to task scheduling for Big Data in cloud environments using classification algorithms," in *2017 7th International Conference on Cloud Computing, Data Science \& Engineering-Confluence*, 2017, pp. 188–192.

[12] L. Zhou, S. Pan, J. Wang, and A. V Vasilakos, "Machine learning on big data: Opportunities and challenges," *Neurocomputing*, vol. 237, pp. 350–361, 2017.

[13] A. Aggarwal, P. Dimri, and A. Agarwal. "Survey on scheduling algorithms for multiple workflows in cloud computing environment." International Journal on Computer Science and Engineering, vol. 7, no. 6, pp. 565-570, 2019.

[14] [14] O. Y. Mohammed, H. I. Abed, and N. A. Sultan, "Design and Implementation of Machine Learning and Big Data Analytics models for Cloud Computing platforms," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 11, no. 6s, pp. 185–192, May 2023, Accessed: Nov. 17, 2023. [Online]. Available: https://www.ijisae.org/index.php/IJISAE/article/view/2840

[15] I. M. Alqahtani, E. Shadadi, and L. Alamer, "Big Data and Reality Mining in Healthcare Smart Prediction of Clinical Disease Using Decision Tree Classifier," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 10, no. 4, pp. 487–492, Dec. 2022, Accessed: Nov. 17, 2023. [Online]. Available: https://ijisae.org/index.php/IJISAE/article/view/2312

[16] R. K. Jena, "Multi objective task scheduling in cloud environment using nested PSO framework," *Procedia Computer Science*, vol. 57, pp. 1219–1227, 2015.

[17] Q. Tian *et al.*, "A hybrid task scheduling algorithm based on task clustering," *Mobile Networks and Applications*, vol. 25, no. 4, pp. 1518–1527, 2020.

[18] A. Alhubaishy and A. Aljuhani, "The best-worst method for resource allocation and task scheduling in cloud computing," in *2020 3rd International Conference on Computer Applications \& Information Security (ICCAIS)*, 2020, pp. 1–6.

[19] G. Rjoub, J. Bentahar, and O. A. Wahab, "BigTrustScheduling: Trust-aware big data task scheduling approach in cloud computing environments," *Future Generation Computer Systems*, vol. 110, pp. 1079–1097, 2020, doi: 10.1016/j.future.2019.11.019.

[20] D. Ding, X. Fan, Y. Zhao, K. Kang, Q. Yin, and J. Zeng, "Q-learning based dynamic task scheduling for energy-efficient cloud computing," *Future Generation Computer Systems*, vol. 108, pp. 361–371, 2020.

[21] Z. Tong, H. Chen, X. Deng, K. Li, and K. Li, "A scheduling scheme in the cloud computing environment using deep Q-learning," *Information Sciences*, vol. 512, pp. 1170–1191, 2020.

[22] M. S. Sanaj and P. M. J. Prathap, "An efficient approach to the map-reduce framework and genetic algorithm based whale optimization algorithm for task scheduling in cloud computing environment," *Materials Today: Proceedings*, vol. 37, pp. 3199–3208, 2021.

[23] Z. Jalalian and M. Sharifi, "A hierarchical multi-objective task scheduling approach for fast big data processing," *The Journal of Supercomputing*, vol. 78, no. 2, pp. 2307–2336, 2022.

[24] N. Arunadevi and V. Thulasiraaman, "Cuckoo Search Augmented MapReduce for Predictive Scheduling With Big Stream Data," *International Journal of Sociotechnology and Knowledge Development (IJSKD)*, vol. 14, no. 1, pp. 1–18, 2022.

[25] K. Loheswaran, T. Daniya, and K. Karthick, "Hybrid cuckoo search algorithm with iterative local search for optimized task scheduling on cloud computing environment," *Journal of Computational and*

Theoretical Nanoscience, vol. 16, no. 5–6, pp. 2065–2071, 2019.

[26] J. García, F. Altimiras, A. Peña, G. Astorga, and O. Peredo, "A binary cuckoo search big data algorithm applied to large-scale crew scheduling problems," *Complexity*, vol. 2018, 2018, doi: 10.1155/2018/8395193.

[27] M. Agarwal and G. M. S. Srivastava, "A Cuckoo Search Algorithm-Based Task Scheduling in Cloud Computing," *Advances in Intelligent Systems and Computing*, vol. 554, pp. 293–299, 2018, doi: 10.1007/978-981-10-3773-3_29.

[28] H. Pan, Y. Lei, and S. Yin, "K-means clustering algorithm for data distribution in cloud computing environment," *International Journal of Grid and Utility Computing*, vol. 12, no. 3, pp. 322–331, 2021.

[29] Dhabliya, D., Ugli, I.S.M., Murali, M.J., Abbas, A.H.R., Gulbahor, U. Computer Vision: Advances in Image and Video Analysis (2023) E3S Web of Conferences, 399, art. no. 04045, .